Pemanfaatan Graf untuk *Dungeon Generation* pada Permainan Dead Cells

Muhammad Furqon - 13519184
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13519184@std.stei.itb.ac.id

***Abstrak*—Permainan Dead Cells adalah sebuah permainan yang memanfaatkan *Procedural Content Generation* untuk menciptakan konten secara acak. Salah satu elemen yang diciptakan secara acak dalam permainan tersebut adalah *dungeon* dengan menggunakan *dungeon generation*. Metode tersebut memanfaatkan graf sebagai basis dalam pembuatan sebuah *dungeon*. Makalah ini bertujuan untuk menjelaskan proses pembuatan sebuah *dungeon* dalam permainan Dead Cell secara prosedural menggunakan bentukan graf.**

***Kata Kunci* —Dead Cells, *Dungeon Generation, Graf, Procedural Content Generation*.**

I. PENDAHULUAN

Dead Cells adalah sebuah permainan digital atau *game* yang dirilis pada tahun 2018 oleh *developer* Motion Twin. *Game* ini tersedia di berbagai perangkat baik *PC, console,* dan *mobile*.

Gambar 1 Tangkapan Layar *Game* Dead Cells (Sumber: https://www.mobygames.com/game/dead-cells/screenshots/gameShotId,916104/platformId,3/ diakses pada tanggal 7 Desember 2020)

Tujuan dari permainan ini adalah melawan musuh sampai mengalahkan musuh terakhir atau *final boss*. Pemain akan mengendalikan sebuah tubuh yang memiliki kepala berupa sel yang berukuran besar. Jika pada suatu saat karakter pemain mati, pemain akan mengulang dari awal.

Pemain memiliki senjata dan kemampuan khusus untuk melawan musuh. Hal tersebut didapat dengan bermain dan eksplorasi di dalam *game*.

Dead Cells dikategorikan ke dalam beberapa genre *game*, salah satunya yaitu *Rogue-Lite.* Genre ini terinspirasi dari *game* Rogue yang mempelopori penggunaan konten acak dalam *game.* Setiap kali bermain semua musuh, kemampuan, peta, dan senjata akan diacak. Metode pembuatan konten secara acak merupakan ciri khas utama *game* yang termasuk ke dalam genre ini. Genre ini juga memberikan pemain perkembangan setiap kali bermain selama permaian. Tujuan akhirnya adalah pemain mencapai akhir dari *game* tersebut[1].

Gambar 2 Peta dalam Dead Cells (Sumber: https://ondrejnepozitek.github.io/Edgar-Unity/docs/examples/dead-cells/ diakses pada tanggal 7 Desember 2020)

Pemain akan menelusuri beberapa *dungeon* dalam *game* tersebut yang dibuat secara acak oleh algoritma pada *game* tersebut (Gambar 2). Pembuatan secara acak diperoleh dengan memanfaatkan algoritma untuk membuat konten, yaitu *procedural content generation*.

Dalam makalah ini akan dijelaskan pemanfaatan graf dalam proses pembuatan *dungeon* dalam Dead Cell.

II. DASAR TEORI

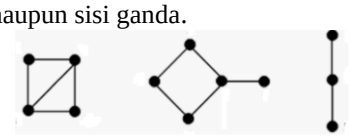
A. Graf

Graf adalah kumpulan titik dan garis yang menghubungkan sebagian (ada kemungkinan kosong). Titik-titik pada sebuah graf disebut sebagai *vertices graph*, istilah lainnya titik, simpul, atau *node*. Garis yang menghubungkan kedua *vertice* tadi adalah *edges*, istilah lainnya garis atau sisi[2].

Graf terbagi menjadi dua jenis, yaitu graf sederhana dan graf tidak sederhana. Graf tidak sederhana dibagi lagi menjadi dua jenis, yaitu graf ganda dan graf semu. Definisi dari masing-masing jenis graf [3], yaitu

1. Graf Sederhana adalah graf yang tidak mengandung

gelang maupun sisi ganda.

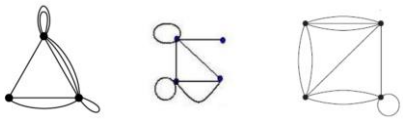


Gambar 3 Graf sederhana (Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf diakses pada tanggal 7 Desember 2020)

2. Graf Tidak Sederhana adalah graf yang mengandung gelang maupun sisi ganda.
   a. Graf Ganda (*multi-graph*) adalah graf yang mengandung sisi ganda.

Gambar 4 Graf Ganda (Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf diakses pada tanggal 7 Desember 2020)

   b. Graf Semu (*pseudo-graph*) adalah graf yang mengandung gelang.



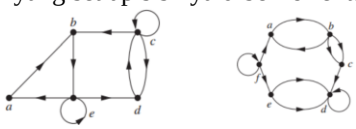
Gambar 5 Graf Semu (Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf diakses pada tanggal 7 Desember 2020)

Graf terbagi menjadi dua jenis berdasarkan orientasinya, yaitu graf tak berarah dan graf berarah. Definisi dari masing-masing jenis graf berdasarkan orientasi [3], yaitu

1. Graf Tidak-Berarah (*undirected graph*) adalah graf yang sisinya tidak memiliki orientasi atau arah.

Gambar 6 Graf Tidak Berarah (Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf diakses pada tanggal 7 Desember 2020)

2. Graf Berarah (*directed graph* atau *digraph*) adalah graf yang setiap sisinya diberi orientasi atau arah.



Gambar 7 Graf Berarah (Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf diakses pada tanggal 7 Desember 2020)

Terminologi atau kosakata yang perlu diketahui mengenai sebuah graf adalah sebagai berikut [3],

1. Ketetanggan(*adjacent*)
   Ketetanggaan adalah dua simpul yang terhubung secara langsung.
2. Bersisian(*incidency*)
   Bersisian adalah untuk sembarang sisi yang menghubungkan dua simpul. Sisi tersebut bersisian dengan kedua simpul tersebut.
3. Derajat(*degree*)
   Derajat sebuah simpul adalah jumlah sisi yang bersisian dengan simpul tersebut.
4. Lintasan(*path*)
   Lintasan yang memiliki panjang n dari simpul awal ke simpul tujuan di dalam graf G ialah barisan berselang-seling simpul-simpul dan sisi-sisi mulai dari simpul awal, lalu sisi yang menghubungkan ke simpul yang menuju simpul tujuan, berulang sampai simpul tujuan. Panjang lintasan adalah banyaknya sisi dalam lintasan tersebut.
5. Sirkuit(*circuit*)
   Sirkuit adalah lintasan yang berawal dan berakhir dengan simpul yang sama. Panjang sirkuit adalah banyaknya sisi dalam lintasan tersebut.
6. Keterhubungan(*connected*)
   Dua simpul dikatakan saling terhubung jika kedua simpul tersebut dihubungkan oleh sebuah lintasan.
7. Upagraf
   Upagraf dari graf G adalah graf yang seluruh elemen simpul dan sisinya adalah sub himpunan dari graf G.

B. Procedural Content Generation (PCG)

Procedural Content Generation(PCG) adalah metode untuk menciptakan konten apapun secara acak[4]. Konten secara otomatis dapat dihasilkan sesuai dengan algoritma yang digunakan. Konten yang diciptakan dapat berupa musik, seni, dan *video game.*

Procedural Content Generation digunakan pada *game* untuk menghasilkan elemen tertentu dari *game* tersebut. Biasanya tidak semua elemen menggunakan metode ini agar menghasilkan hasil yang konsisten.

Ada beberapa kelebihan dengan penggunaan metode ini. Pertama metode ini dapat menghasilkan konten yang banyak secara cepat dan memenuhi desain yang diinginkan. Kedua, variasi hasil yang dihasilkan beragam dan unik sehigga dapat membuat *game* itu lebih baik saat dimainkan berulang kali. Ketiga, menghemat biaya dan waktu pembuatnya dibandingkan mendesain dan membuat setiap aspek dari *game* secara manual. Keempat, basis dari *procedural content generation* dapat digunakan untuk beradaptasi menyesuaikan dengan pemain.

Salah satu penggunaan *procedural content generation* adalah pada aspek pembangunan sebuah *dungeon*. *Dungeon* merupakan aspek penting dalam sebuah *game* yang bergenre *adventure* atau *RPG(Role Playing Game)* [4].

C. Dungeon Generation

Definisi *dungeon* itu sendiri pada sebuah *game* adalah sebuah lingkungan yang bersifat seperti labirin, terdiri dari tantangan

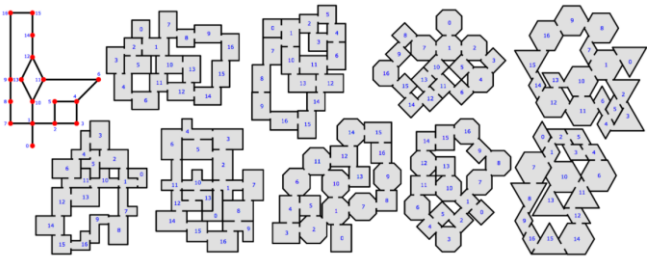
yang saling berhubungan satu sama lain, *reward*, dan *puzzle* yang saling terikat secara erat dalam waktu dan ruang untuk menghasilkan perkembangan secara sangat terstruktur pada saat bermain[4].

Dungeon generation memiliki arti pembuatan sebuah *dungeon* tersebut dengan *procedural content generation*. Elemen yang akan dihasilkan secara acak pada suatu *dungeon* biasanya terdiri dari karakter selain player (*non playable characters*), dekorasi atau tampilan, dan objek.

Metode pada sebuah *dungeon generator* dibagi menjadi tiga bagian [4].

1. Model representasi, sebuah bentuk abstrak yang merupakan representasi paling sederhana dari versi lengkap *dungeon* tersebut.
2. Metode untuk membangun model representasi tersebut.
3. Metode membuat bentuk geometri nyata dari *dungeon* tersebut sesuai dengan representasi model.

Salah satu metode yang digunakan adalah pengunaan metode graf dalam menciptakan sebuah *dungeon*. Dengan menggunakan *procedural content generation*, graf dibuat terlebih dahulu sebagai sebuah model representasi, setelah itu barulah setiap simpul diubah menjadi ruangan yang sesuai untuk menghasilkan bentuk yang sesuai dengan graf tersebut.



Gambar 8 *Dungeon Layouts* dari Satu Graf (Sumber: http://chongyangma.com/publications/gl/index.html diakses pada tanggal 7 Desember 2020)

Parameter dari suatu ruangan dapat diatur sesuai kebutuhan dan kemauan dari pembuatnya. Sebuah *dungeon* biasanya memiliki keunikannya masing-masing dengan sebuah tema tertentu.

Gambar di atas merupakan beberapa *layout* atau peta yang dihasilkan menggunakan satu graf yang sama. Setiap simpul yang berupa titik merah telah diganti dengan ruangan yang memenuhi parameter[5].

III. DUNGEON GENERATION DENGAN GRAPH PADA PERMAINAN DEAD CELLS

Pada sebuah *game* yang menggunakan *dungeon generation* dengan graf, peta tersebut dapat direpresentasikan dalam sebuah graf. Pada graf tersebut simpul atau *node* merupakan sebuah ruangan atau *room* pada *dungeon* tersebut. Hubungan antar simpul, yaitu sisi atau *edge* adalah pintu atau koridor yang menghubungkan ruangan tersebut[6].

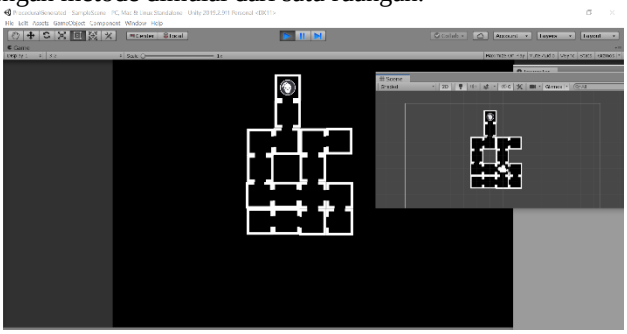
A. Dungeon Generation: Graph Based Approach (One Node)

Procedural Content Generation digunakan dengan metode graf dapat menghasilkan hasil secara acak sesuai dengan algoritma yang digunakan.

Cara paling sederhana dan mudah untuk menghasilkan hasil tersebut adalah menggunakan satu simpul sebagai awal dari *dungeon* tersebut (*one node*).

Unit terkecil dalam pembangkitannya(generator) adalah simpul yang berupa ruangan atau *room*. Ruangan pada dungeon generation tidak diacak, tetapi menggunakan himpunan ruangan yang telah disediakan oleh pembuatnya.

Contoh di bawah ini menggunakan Unity dengan bahasa C# dengan metode dimulai dari satu ruangan.



Gambar 9 Dungeon Generation (Sumber: Dokumen Pribadi)

Simpul awal yang digunakan dalam proses tersebut adalah ruangan yang memiliki empat pintu atau arah. Dari ruangan awal itu akan dihasilkan ruangan-ruangan lainnya sampai dihasilkan ruangan yang semua pintunya telah terhubung atau tertutup. Pada lintasan terjauh dari lokasi awal atau ruangan yang paling dihasilkan paling akhir, dibuat ruangan *boss* yang ditandai dengan simbol sebagai penanda.

Pada contoh di atas dibentuk sebuah *dungeon* dengan 16 ruangan. Ruangan *boss* yang menandakan bagian akhir berjarak sejauh tujuh simpul dari simpul awal dengan lintasan terpanjang sebanyak delapan simpul.

```
void Spawn()
{
    if (spawned == false)
    {
        if (openingDirection == 1) //1 --> butuh spawn pintu bottom (bawah)
        {
            rand = Random.Range(0, templates.bottomRooms.Length);
            Instantiate(templates.bottomRooms[rand], transform.position, templates.bottomRooms[rand].transform.rotation);
        }
        else if (openingDirection == 2) //2 --> butuh spawn pintu top (atas)
        {
            rand = Random.Range(0, templates.topRooms.Length);
            Instantiate(templates.topRooms[rand], transform.position, templates.topRooms[rand].transform.rotation);
        }
        else if (openingDirection == 3) //3 --> butuh spawn pintu left (kiri)
        {
            rand = Random.Range(0, templates.leftRooms.Length);
            Instantiate(templates.leftRooms[rand], transform.position, templates.leftRooms[rand].transform.rotation);
        }
        else if (openingDirection == 4) //4 --> butuh spawn pintu right (kanan)
        {
            rand = Random.Range(0, templates.rightRooms.Length);
            Instantiate(templates.rightRooms[rand], transform.position, templates.rightRooms[rand].transform.rotation);
        }
        spawned = true; //akan stop
    }
}
```

Gambar 10 Potongan Kode Dungeon Generation (Sumber: Dokumen Pribadi)

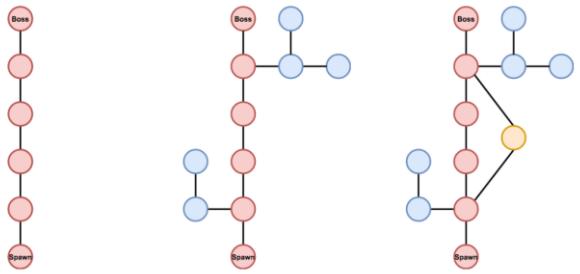
Penggunaan konstrain untuk menghasilkan ruangan baru adalah pintu yang digunakan pada ruangan tersebut sehingga akan terhubung dengan ruangan sebelumnya.

Kelemahan dari metode ini adalah hasilnya terlalu acak. Kasus terburuk adalah hasilnya terlalu pendek atau terlalu panjang. Hal ini dikarenakan kurangnya pengaturan yang dapat membuat hasil yang konsisten.

B. Dungeon Generation: Graph Based Approach (Main Path)

Penggunaan graf pada *dungeon generation* dengan metode ini dapat memperbaiki masalah dari *one node*. Masalah ukuran dan hasil yang terlalu acak dapat dipecahkan dengan menggunakan metode ini.

Setiap simpul melambangkan ruangan dan setiap sisi melambangkan koridor atau penghubung antar ruangan.



Gambar 11 *Dungeon Graph* (Sumber: https://ondra.nepozitek.cz/blog/42/dungeon-generator-part-1-node-based-approach/ diakses pada tanggal 7 Desember 2020)

Merah melambangkan konten utama yang selalu ada di dalam peta tersebut. Simpul utama ini merupakan jalan utama yang harus dilewati oleh pemain. *Optional path* berwarna biru, simpul ini merupakan simpul opsional yang bertujuan sebagai konten tambahan yang tidak perlu diselesaikan. Oranye adalah koridor yang merupakan jalan pintas yang menghubungkan graf[6].

Cara *generate* sebuah *dungeon* dengan menggunakan metode ini adalah dengan membuat graf dengan simpul utama atau merah terlebih dahulu. Setelah simpul merah terbentuk barulah secara acak sesuai dengan aturan yang diinginkan, ditambahkan simpul biru sebagai tetangga dari simpul merah. Barulah terakhir bisa ditambahkan simpul oranye yang menghubungkan dua simpul di tingkatan yang berbeda secara acak.

Alasan digunakan metode ini adalah bisa diatur dengan lebih mudah seberapa banyak konten yang harus dilakukan dan dapat dengan mudah menambah konten sebagai tambahan.

Pada permainan Dead Cells digunakan metode ini dengan eksepsi tidak menggunakan simpul bertipe oranye yang menjadi jalan pintas.

C. Dead Cells Room Template

Dead Cells menggunakan procedural generation untuk membuat konten secara random. Mulai dari musuh, *dungeon*, dan barang yang bisa diperoleh pemain. Akan tetapi, semua ruangan dibuat dengan *template*(basis) sehingga tidak akan menyebabkan hasil yang terlalu acak.

Semua ruangan dalam permainan ini dibuat secara manual satu per satu sebagai *template* yang nantinya akan digunakan pada proses pembuatan *dungeon*.

Gambar 12 Contoh *Room* Template (Sumber: https://www.gamasutra.com/blogs/SebastienBENARD/20170329/294642/Building_the_Level_Design_of_a_procedurally_generated_Metroidvania_a_hybrid_approach.php diakses pada tanggal 7 Desember 2020)

Alasan dibuat basis adalah karena kelemahan terbesar dari *procedural generation* adalah jika hasil yang dihasilkan terlalu acak dapat menyebabkan suatu *dungeon* tidak dapat diselesaikan oleh pemain karena kesalahan algoritma.

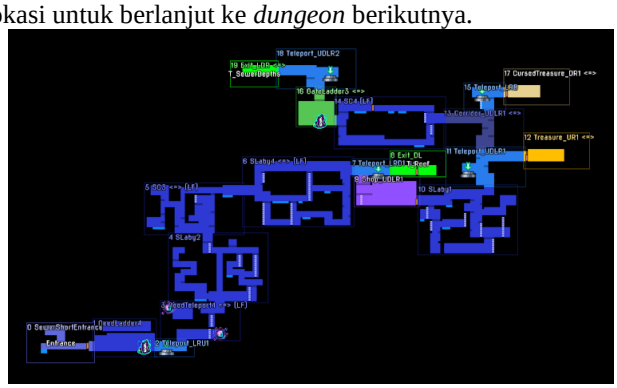
Setiap ruangan memiliki tujuan tertentu dalam desainnya. Terdapat ruangan untuk melawan musuh, menyimpan senjata, dan tempat membeli barang. Setiap ruangan tersebut akan memiliki isi yang berbeda sesuai dengan fungsinya. Pembagian ruangan selain dari kegunaannya juga dibagi sesuai dengan letak pintu pada ruangan tersebut agar bisa tersambung dengan baik.

Room pada *game* Dead Cells dibagi sesuai dengan *biome*. Setiap *biome* juga memiliki aturan yang berbeda mengenai tata letak ruangan[7].

D. Dead Cells Dungeon Layout

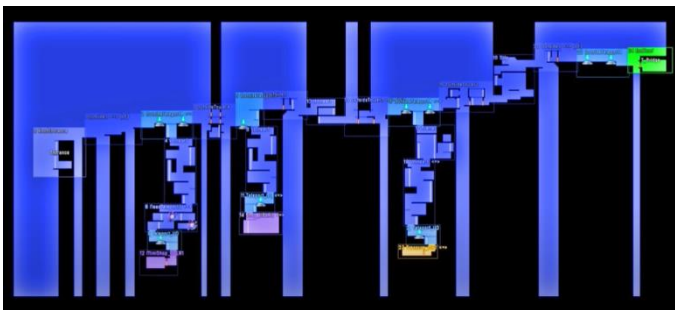
Setiap *biome* pada permainan ini memiliki *constrain* atau aturan yang unik dan berbeda-beda. Setiap *biome* memiliki tata letak atau *layout* yang akan mempengaruhi proses pembuatan *dungeon* sesuai dengan aturan tersebut.

Sebagai contohnya pada *biome* "Prisoners' Quarters" yang terletak di bawah tanah, pintu keluarnya akan terletak di dua lokasi untuk berlanjut ke *dungeon* berikutnya.



Gambar 13 *Layout Underground Map* (Sumber: https://www.gamasutra.com/blogs/SebastienBENARD/20170329/294642/Building_the_Level_Design_of_a_procedurally_generated_Metroidvania_a_hybrid_approach.php diakses pada tanggal 7 Desember 2020)

Gambar di atas merupakan salah satu contoh dari peta dengan *biome* yang terletak di bawah tanah. Hal ini menyebabkan ruangannya tersusun seperti labirin dan sempit. Pembuat *game* ingin pemain memiliki ruang gerak yang lebih terbatas dalam ruangan-ruangan ini[7].

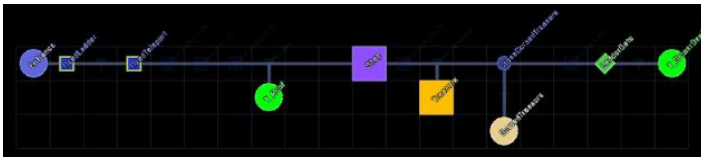


Gambar 14 *Layout Roof Map* (Sumber: https://ondrejnepozitek.github.io/Edgar-Unity/docs/examples/dead-cells/ diakses pada tanggal 7 Desember 2020)

Gambar di atas merupakan contoh lain dari *biome* yang terletak di atap dan terpisah menjadi beberapa bangunan berbentuk menara. Jenis ruangan terbagi menjadi ruangan di tempat terbuka dan ruangan yang menjadi interior atau isi dari setiap menara tersebut. Pemain dapat lebih leluasa bergerak antar menara dan desainnya jauh lebih terbuka dibandingkan tempat yang terletak di bawah tanah[8].

Dengan menggunakan tata letak yang berbeda pada setiap *biome*, pembuat *game* dapat memberikan keunikan dan tantangan yang berbeda.

E. Dead Cells Procedural Dungeon Generation



Gambar 15 Graf Skema *Dungeon* Utama (Sumber: https://www.gamasutra.com/blogs/SebastienBENARD/20170329/294642/Building_the_Level_Design_of_a_procedurally_generated_Metroidvania_a_hybrid_approach.php diakses pada tanggal 7 Desember 2020)

Graf pada gambar di atas merupakan graf dari biome "Prisoner's Quarter" dengan kode "T_Sewer" yang merupakan peta pertama pada *game* ini dengan tata letak berada di bawah tanah. Graf ini merupakan hasil dari algoritma *dungeon generator* pada game tersebut.

Penjelasan setiap elemen pada graf tersebut adalah sebagai berikut, Entrance(lingkaran biru) =>needLadder(kotak biru)=> needTeleport(kotak biru) =>T_Roof(lingkaran hijau)=> Shop(kotak ungu) =>Treasure(kotak emas) => crossCursedTreasure(lingkaran biru) =>CursedTreasure(lingkaran coklat) =>LadderGate(belah ketupat hijau) => T_SewerDepths(lingkaran hijau).

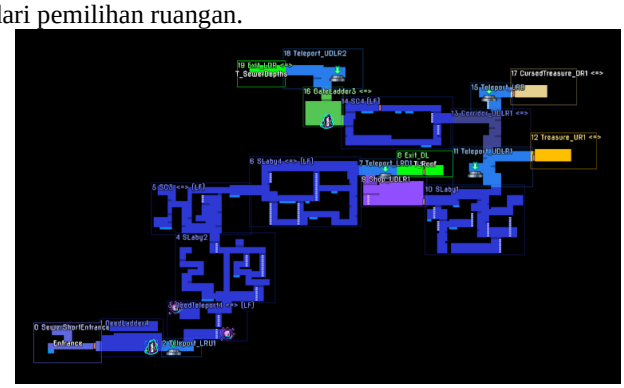
Lingkaran biru adalah ruang yang sudah ditentukan diambil dari *template* yang sudah ada. Kotak biru dengan garis luar kuning merupakan konstrain atau aturan untuk menentukan aturan untuk ruangan yang nantinya akan ditambahkan. Kotak biru dan segitiga biru merupakan ruangan tambahan sesuai dengan konstrain tadi. Kotak ungu merupakan toko atau tempat membeli barang dengan menukarkan uang saat bermain. Kotak emas adalah ruang dengan barang sebagai *reward* untuk pemain. Lingkaran coklat adalah ruang dimana pemain dapat mendapat barang yang kualitas lebih bagus, tetapi mendapatkan kutukan. Lingkaran hijau adalah jalan keluar dari *dungeon* tersebut, minimal pada setiap *dungeon* pasti memiliki satu jalan keluar. Untuk salah satu pintu keluar (T_SewerDepths) terdapat belah ketupat hijau sebelumnya yang memerlukan barang tertentu agar bisa menggunakan jalan tersebut.

Graf yang digunakan memiliki arah dari kiri ke kanan. Graf di-*generate* mulai dari sebelah kiri ke sebelah kanan .Simpul yang berada di sebelah kiri memiliki tingkatan yang lebih atas Graf tidak bisa membuat simpul selanjutnya ke tingkatan yang lebih atas. Graf dibuat sesuai aturan dan tata letak yang sudah ditentukan untuk *biome* tersebut[7].

Gambar 16 Graf Skema *Dungeon* Lengkap (Sumber: https://www.gamasutra.com/blogs/SebastienBENARD/20170329/294642/Building_the_Level_Design_of_a_procedurally_generated_Metroidvania_a_hybrid_approach.php diakses pada tanggal 7 Desember 2020)

Setelah jalur utama telah dibuat, barulah akan ditambahkan konten atau ruangan tambahan ke dalam graf. Gambar di atas merepresentasikan graf yang telah ditambahkan beberapa simpul. Segitiga biru dan kotak biru tersebut adalah simpul yang ditambahkan sesuai dengan syarat dari kotak dengan garis luar kuning. Berarti ruangan yang ditambahkan memenuhi syarat membutuhkan tangga dan *teleporter*.

Konten akan ditambahkan sesuai aturan yang sudah ditentukan. Graf di atas adalah graf yang akan menjadi basis dari pemilihan ruangan.



Gambar 17 Hasil Akhir *Dungeon* (Sumber: https://www.gamasutra.com/blogs/SebastienBENARD/20170329/294642/Building_the_Level_Design_of_a_procedurally_generated_Metroidvania_a_hybrid_approach.php diakses pada tanggal 7 Desember 2020)

Gambar di atas merupakan hasil dari graf setelah setiap simpul diganti dengan menggunakan ruangan yang terdapat pada daftar ruangan. Susunan setiap ruangan tidak akan saling timpa-menimpa dan akan saling terhubung sesuai dengan pintu yang terdapat di ruangan.

Setelah berhasil membentuk peta dengan setiap ruangannya, barulah barang, musuh, dan objek lainnya yang ada di dalam ruangan di-*generate* secara acak sesuai dengan komponen yang

ada di ruangan tersebut. Hasil setelah semua objek dan musuh telah ditambahkan ke dalam setiap ruangan adalah hasil akhir yang akan digunakan dalam permainan.

IV. KESIMPULAN

Teori graf memiliki banyak kegunaan di bidang *video game*. Salah satu penggunaan dari graf adalah dalam pembuatan *dungeon* secara prosedural. Hal ini dimanfaatkan dalam *game* Dead Cells.

Graf merupakan salah satu alternatif untuk menghasilkan model representasi yang baik sebagai basis untuk *dungeon generation* yang dapat menghasilkan hasil yang baik dan acak secara teratur sesuai dengan algoritma dan konstrain yang telah didesain.

V. UCAPAN TERIMA KASIH

Segala puji dan syukur penulis panjatkan ke hadirat Allah SWT karena berkat rahmat, hidayah, dan karunia-Nya, penulis dapat menyelesaikan makalah ini tepat waktu dengan baik. Terima kasih kepada kedua orang tua yang telah memberi dukungan kepada penulis.

Terima kasih yang sebesar-besarnya kepada semua pihak yang telah membantu penulis dalam menyelesaikan makalah ini. Terima kasih kepada Ibu Nur Ulfa Maulidevi selaku dosen pengampu mata kuliah Matematika Diskrit IF2120 kelas K4, dan juga kepada seluruh tim pengajar mata kuliah IF2120 yang telah mengajarkan ilmunya kepada penulis sehingga penulis mampu menyelesaikan makalah ini.

Penulis juga menyampaikan terima kasih kepada semua teman penulis yang selalu memberikan motivasi, masukan, dan bantuan dalam pengerjaan tugas makalah ini.

DAFTAR REFERENSI

[1] Bycer, Josh, https://www.gamasutra.com/blogs/JoshBycer/20191125/354673/The_Roguelike_Debate__Roguelikes_vs_Roguelites.php, diakses tanggal 7 Desember 2020.

[2] Weisstein, Eric. Wolfram Research. https://mathworld.wolfram.com/Graph.html, diakses tanggal 7 Desember 2020.

[3] Munir, Rinaldi, https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf, diakses tanggal 7 Desember 2020.

[4] R. van der Linden, R. Lopes, dan R. Bidarra, 2014. Procedural Generation of Dungeons. IEEE Transactions on Computational Intelligence and AI in Games, vol. 6, no. 1, pp. 78-89, doi: 10.1109/TCIAIG.2013.2290371.

[5] Ma, Chongyang & Vining, Nicholas & Lefebvre, Sylvain & Sheffer, Alla. 2014. Game Level Layout from Design Specification. Computer Graphics Forum. 33. 10.1111/cgf.12314.

[6] Nepoticzek, Ondra. https://ondra.nepozitek.cz/blog/42/dungeon-generator-part-1-node-based-approach/ diakses tanggal 7 Desember 2020.

[7] Benard, Sebastien, https://www.gamasutra.com/blogs/SebastienBENARD/20170329/294642/Building_the_Level_Design_of_a_procedurally_generated_Metroidvania_a_hybrid_approach.php diakses tanggal 7 Desember 2020.

[8] Nepoticzek, Ondra. https://ondrejnepozitek.github.io/Edgar-Unity/docs/examples/dead-cells/ diakses tanggal 7 Desember 2020.

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, 7 Desember 2020

Muhammad Furqon
13519184