Aplikasi *Binary Space Partitioning* dalam Desain Level Permainan Roguelike

Amar Fadil - 13520103[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13520103@std.stei.itb.ac.id

***Abstrak*—Desain level permainan roguelike membutuhkan algoritma *procedural generation* dalam pembuatannya, salah satu variannya menggunakan struktur data *Binary Space Partitioning (BSP) tree*. Pohon *BSP* menyusun level permainan dalam struktur data secara spasial dengan mempartisi ruang permainan menjadi beberapa daerah yang dapat ditentukan secara acak. Kelebihannya dalam struktur data spasial mengakibatkan mudahnya kontrol parameter pembuatan level sehingga dapat dimainkan, sesuai dengan kehendak pengembang, dan menyenangkan untuk dimainkan.**

Kata Kunci—binary space partitioning, game development, procedural generation, level design.

I. PENDAHULUAN

Roguelike merupakan genre permainan yang menugaskan para pemainnya untuk melakukan *dungeon crawling* dan mendapatkan item yang dapat digunakan sebagai progresi dari pemain. Permainan roguelike yang akhir-akhir ini sering dimainkan oleh para *gamers* yaitu Hades dan The Binding of Isaac. Genre ini biasanya dicirikan dengan *procedurally generated dungeon* beserta musuh dan *loot* yang bisa pemain dapatkan secara acak. Karena desain levelnya yang acak dibuat secara procedural, hal ini membutuhkan algoritma pembuatan level permainan dengan batasan tertentu sehingga layak untuk dimainkan.

Dalam membuat level permainan, dibutuhkan beberapa konsiderasi untuk dapat dimainkan oleh pemain. Level permainan yang mustahil untuk dilakukan, misalnya pembagian ruang yang tidak rata atau tidak terhubung, tentu saja menjadi masalah. Selain itu, level yang terlalu banyak atau terlalu sedikit ruang akan cukup mudah membosankan pemain. Oleh karena itu, dibutuhkan sebuah struktur data yang dapat membagi rata ruang sehingga level permainan disusun dengan baik. Salah satu implementasi struktur data ini adalah *Binary Space Partitioning tree*.

Binary Space Partitioning tree, juga sering disingkat sebagai BSP tree, merupakan salah satu aplikasi dari *binary tree*. BSP tree bekerja dengan membagi ruang menjadi dua bagian berbeda secara rekursif hingga menjadi ruang terkecil yang memenuhi kriteria. BSP tree pada awalnya dikembangkan untuk keperluan grafika komputer. Teknik *rendering* geometri statis dengan BSP tree terbukti sangat efisien untuk menyelesaikan masalah *visible surface determination* [1]. Hal ini yang kemudian mengantarkan pengembang di Id Software dengan keterbatasan *hardware* saat itu untuk menggunakan BSP tree pada implementasi *rendering engine* dalam membuat dunia virtual untuk permainan Doom (1993) [2]. Melihat potensi yang besar, BSP tree kemudian berkembang dan digunakan dalam berbagai aplikasi spesifik yang membutuhkan data spasial seperti *raytracing*, *constructive solid geometry*, *map generation*, dan *collision detection*.

Pada makalah ini, akan dibuat sebuah algoritma pembuatan level roguelike secara prosedural dengan memakai pohon BSP. Untuk memudahkan implementasi, implementasi ini akan menggunakan *game engine* Unity.

II. TEORI DASAR

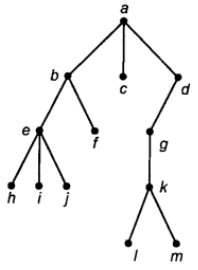
A. Pohon

Pohon merupakan graf sederhana, yakni graf yang tidak memiliki sisi gelang dan sisi ganda, yang tak-berarah. Teori pohon ini pertama kali dikemukakan oleh matematikawan berkebangsaan Inggris, Arthur Cayley, pada tahun 1857 yang menggunakan pohon untuk menghitung berbagai macam tipe bahan kimia [3]. Salah satu contoh pohon yang paling sering dipakai adalah pohon keluarga. Hingga sekarang, teori ini memiliki banyak aplikasi terutama dalam bidang *computer science*.

Sifat-sifat yang dimiliki oleh pohon G = (V, E) dengan jumlah simpul n sebagai berikut: [4]

- Lintasan tunggal menghubungkan setiap pasang simpul pada G.
- Semua sisi pada G adalah jembatan
- G saling terhubung
- G memiliki m = n – 1 buah sisi
- G tidak memiliki sirkuit
- Penambahan satu sisi saja pada graf G akan mengakibatkan G hanya memiliki satu sirkuit.

B. Pohon Berakar



Gambar 2.1 Contoh pohon berakar
Sumber: R. Munir, *Matematika Diskrit*, 3rd ed., pp. 459

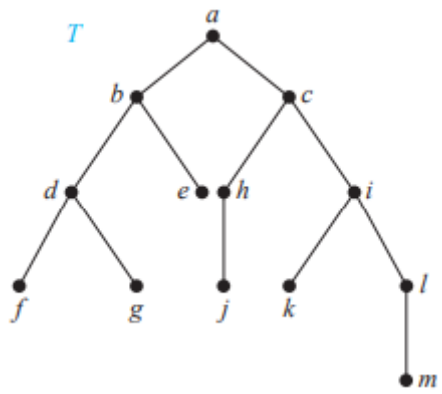
Terdapat salah satu jenis dari pohon yakni pohon berakar yang satu buah simpulnya diperlakukan sebagai akar dengan sisi-sisinya diberi arah sehingga menjadi graf berarah. Jika diketahui sebuah pohon berakar T, maka pohon ini memiliki beberapa terminologi yang perlu diketahui [4]:

1) Anak (*Child*) dan Orangtua (*Parent*)
   Dua buah simpul pada T, misalnya A dan O, A dikatakan anak dari O dan O dikatakan orangtua dari A jika ada sisi pada simpul A ke simpul O. Pada gambar 2.1, diketahui bahwa k memiliki anak j dan m, j memiliki orangtua k, b merupakan orangtua dari e dan f, dan seterusnya.
2) Lintasan (*Path*)
   Lintasan dibentuk dari simpul-simpul $v_1, v_2, \ldots, v_j$ pada T sedemikian rupa sehingga v_{i+1} merupakan *children* dari v_i (dan v_i merupakan *parent* dari v_{i+1}) untuk semua $1 \leq i < k$. Lintasan ini memiliki panjang yakni total sisi yang dilewati pada lintasan. Pada gambar 2.1, dapat diketahui lintasan dari d ke m yakni d, g, k, dan m.
3) Keturunan (*Descendant*) dan Leluhur (*Ancestor*)
   Misalkan L dan K merupakan simpul pada T, simpul L dikatakan sebagai leluhur dari simpul K dan K merupakan keturunan dari L jika ada lintasan dari L menuju K. Pada gambar 2.1, pohon tersebut memiliki contoh a merupakan leluhur dari j dan j merupakan keturuan dari a.
4) Saudara kandung (*Sibling*)
   Misalkan X dan Y merupakan simpul pada T, simpul X merupakan saudara kandung dari Y dan sebaliknya jika X dan Y memiliki orangtua yang sama. Pada gambar 2.1, pohon tersebut memiliki simpul f yang merupakan saudara kandung dari e dan begitu sebaliknya, namun f bukan saudara kandng dari g karena f dan g memiliki orangtua yang berbeda.
5) Upapohon (*Subtree*)
   Misalkan A merupakan simpul pada T, upapohon adalah upagraf $T' = (V', E')$ dengan A sebagai akarnya dan V' merupakan semua keturunan dari A beserta A itu sendiri serta E' merupakan sisi-sisi pada semua lintasan yang berasal dari A. Pada gambar 2.1, terdapat upapohon $T' = (\{g, k, l, m\}, \{(g, k), (k, l), (l, m)\})$ yang berakar dari g.
6) Derajat (*Degree*)
   Jumlah anak atau upapohon pada sebuah simpul x pada pohon berakar T disebut sebagai derajat dari x. Terdapat juga istilah derajat pohon, yakin derajat simpul maksimum dari semua simpul pohon. Pada contoh gambar 2.1 diketahui bahwa derajat m dan l adalah 0, derajat g adalah 1, dan derajat T adalah 3 karena derajat simpul maksimum pada T adalah derajat a atau derajat e yaitu 3.
7) Daun (*Leaf*)
   Daun merupakan semua simpul pada T yang derajatnya 0. Dengan kata lain, daun merupakan semua simpul pada T yang tidak memiliki anak. Pada contoh gambar 2.1, semua simpul yang merupakan daun adalah h, i, j, f, c, l, dan m.
8) Simpul dalam (*Internal nodes*)
   Semua simpul yang memiliki anak atau berderajat lebih dari 0 disebut sebagai simpul dalam. Pada gambar 2.1, semua simpul yang merupakan simpul dalam adalah a, b, e, d, g, dan k.
9) Aras/Tingkat (*Level*)
   Aras atau tingkat pada simpul di dalam T didefinisikan sebagai 0 jika simpul merupakan akar. Jika tidak, aras simpul tersebut adalah panjang lintasan dari akar sampai ke simpul tersebut. Pada contoh gambar 2.1, aras atau tingkat dari simpul e, f, dan g adalah 2.
10) Tinggi (*Height*) atau Kedalaman (*Depth*)
    Tinggi atau kedalaman dari T adalah panjang lintasan maksimum dari akar ke daun. Tinggi atau kedalaman dari T juga dapat didefinisikan sebagai aras atau tingkatan maksimum dari pohon tersebut. Contoh pada gambar 2.1, pohon berakar tersebut memiliki tinggi 4.

Sebuah pohon berakar, misalnya T, dikatakan sebagai pohon m-ary jika semua simpul dalam pada T tidak memiliki lebih dari m anak. T juga dapat disebut sebagai *full m-ary tree* (pohon m-ary penuh atau teratur) jika semua simpul dalam pada T memiliki tepat m anak. Pohon berakar yang setiap anak dari *internal nodes* memiliki keterurutan tertentu juga disebut sebagai *ordered rooted tree* atau pohon berakar terurut [3].

C. Pohon Biner

Pohon biner merupakan pohon berakar m-ary dengan m = 2 atau 2-ary. Pohon binary merupakan pohon berakar yang mempunyai banyak aplikasi dalam bidang sains komputer. Beberapa aplikasi *binary tree* yang dapat digunakan adalah *prefix code*, kompresi Huffman, pohon keputusan, *parsing tree*, pohon ekspresi, pohon pencarian (*binary search tree/BST*) dan *binary space partitioning tree*.



Gambar 2.2 Contoh pohon biner
Sumber: K. H. Rosen, *Discrete Mathematics and Its Applications*, 7th ed., pp. 749

Pohon biner merupakan salah satu pohon berakar terurut karena urutan anaknya yang berbeda. Sebuah *binary tree*, misalnya T, memiliki beberapa istilah yang juga perlu diketahui [4]:

1) Anak kiri (*left child*) dan anak kanan (*right child*)
   Setiap simpul dalam pada T memiliki paling banyak dua anak yang berbeda, yakni anak kiri dan anak kanan.
2) Upapohon kiri (*left subtree*) dan upapohon kanan (*right subtree*)
   Setiap simpul dalam pada T memiliki dua upapohon, yakni upapohon kiri jika akarnya adalah anak kiri, dan upapohon kanan jika akarnya adalah anak kanan.

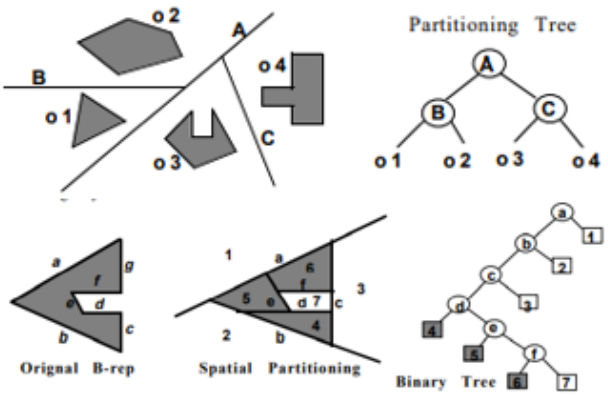
Selain itu, sebuah binary tree, misalnya T, juga dapat

dikatergorikan menjadi beberapa jenis pohon biner [4]:

1) Pohon condong (*skewed tree*)
   Pohon biner T disebut sebagai pohon condong jika semua simpulnya berada pada bagian kiri saja (pohon condong-kiri/*left skewed tree*) atau pada bagian kanan saja (pohon condong-kanan/*right skewed tree*). Dengan kata lain semua simpulnya merupakan anak kiri saja atau anak kanan saja.
2) Pohon biner penuh (*full/complete binary tree*)
   Jika setiap simpul dalam pada pohon biner T memiliki tepat dua buah anak dan simpul tersebut tidak berada pada tingkat paling bawah, maka T dapat disebut sebagai pohon biner penuh. Pohon biner jenis ini memiliki properti khusus yakni pohon biner penuh dengan tinggi h memiliki total daun 2^h.
3) Pohon biner seimbang (*balanced binary tree*)
   Untuk dapat dikatakan sebagai pohon seimbang, pohon biner T harus memiliki perbedaan tinggi antara kedua upapohon paling banyak 1. Dengan kata lain, setiap daun pada sebuah pohon biner seimbang dengan kedalaman h harus berada pada tingkat h atau $h - 1$.

D. Binary Space Partitioning Tree

Pohon BSP meupakan struktur data yang merepresentasikan pemotongan dari sebuah objek geometri atau ruang berisikan objek-objek geometri. Struktur ini sekaligus juga memberikan informasi untuk pencarian dan representasi setiap objek. Pohon BSP dapat dikatakan sebagai generalisasi dari pohon pencarian (*binary search tree*) pada dimensi lebih besar dari 1, sehingga representasi ruang pada BSP biasanya memberikan sebuah keterurutan spasial pada ruang. Pohon BSP dapat digunakan untuk partisi objek geometri (pada objek) maupun partisi ruang (antar objek) [5].



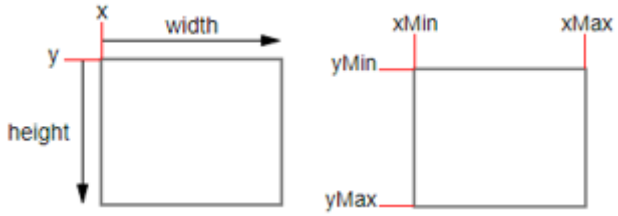
Gambar 2.3 Contoh partisi ruang (antar-objek) dan partisi objek (pada objek) dengan pohon BSP
Sumber: B. F. Naylor, *A tutorial on binary space partitioning trees*, Comput. Games Dev. Conf. Proc., no. August, pp. 433.

Pohon BSP ini akan terus valid selama objek tidak bergerak. Alasan inilah yang mengakibatkan pohon akan dibuat hanya sekali dan kemudian disimpan sebagai struktur pada ruang penyimpanan. Hal ini tentunya membuat pohon BSP hanya efisien pada objek atau ruang yang statis, seperti latar belakang dari sebuah permainan video.

Untuk sebuah pohon BSP, misalnya S, pohon ini akan menyimpan objek-objek pada ruang (atau daerah-daerah partisi dari objek) pada setiap daun dari pohon. Pohon S ini juga akan menyimpan informasi mengenai segmentasi ruang (atau sisi perpotongan objek) pada setiap simpul dalamnya. Akar pohon S sendiri merupakan ruang awal sebelum dipartisi (atau objek awal sebelum dipotong). Pohon S yang disimpan kemudian dapat digunakan untuk berbagai macam kebutuhan, salah satunya pembuatan level pada permainan video.

E. Rect Structure

Untuk mendefinisikan ruang pada pohon BSP, implementasi akan menggunakan struktur Rect bawaan dari Unity yang merupakan representasi dari persegi panjang (*rectangle*) pada ruang dimensi dua. Persegi panjang ini didefinisikan dengan posisinya pada absis (X) dan ordinat (Y) serta lebar (*width*) dan panjang (*height*) dari persegi tersebut. Properti lain yang dimiliki struktur ini adalah xMin dan yMin yang merupakan posisi ujung terkecil masing-masing koordinat serta xMax dan yMax yang merupakan posisi ujung terbesar masing-masing koordinat [6].



Gambar 2.4 Definisi persegi panjang pada struktur Rect
Sumber: Unity Technologies, *Unity - Scripting API: Rect (v2020.3)*. https://docs.unity3d.com/ScriptReference/Rect.html (accessed Dec. 11, 2021).

III. Implementasi Pohon BSP

A. Desain Level Permainan

Level permainan pada roguelike memiliki beberapa kriteria, tergantung dari yang diberikan oleh pengembang permainan. Kriteria ini dapat bervariasi tergantung *gameplay* yang diinginkan. Pada makalah ini, permainan akan mengikuti *game roguelike* dengan *dungeon crawling* pada dasarnya, yakni pemain akan dihadapi oleh level permainan dengan ukuran tetap yang dibuat dengan prosedur tertentu (*procedurally generated*) secara acak.

Setiap level pada awalnya akan memiliki ukuran sebesar 100 x 100. Ruangan memiliki ukuran dalam rentang 25 x 25 hingga 50 x 50. Setiap ruangan akan berisi 3-10 monster dan 1-3 chest untuk di-*loot* tergantung ukuran ruangan. Karena keterbatasan waktu, makalah ini hanya memfokuskan pada aspek pembangkitan level permainan. Prosedur untuk membuat level dapat diacak dengan pembangkit acak semu (*pseudorandom number generator*/PRNG). Setiap level dapat memiliki konfigurasi unik berupa umpan dari pembangkit acak semu, sehingga jika level menggunakan umpan yang sama, konfigurasi level yang dibuat akan sama. Hal ini mempermudah untuk *debugging* dan berbagi konfigurasi level untuk dimainkan oleh

pemain lainnya.

Untuk membantu implementasi terhadap partisi dan pembuatan ruang, makalah ini akan menggunakan struktur Rect sebagai penyimpanan representasi ruang untuk setiap Level. Kelas Level sendiri akan menjadi basis dari struktur data untuk level permainan dengan pohon BSP. Kelas Level ini memiliki properti seperti pohon biner biasa, yakni referensi Level kiri dan kanan (anak-anaknya), serta properti tambahan seperti *room* (ruang level permainan) dan *plane* (daerah level partisi) yang bertipe Rect dan id untuk penomoran daerah. Konstruksi kelas Level membutuhkan *plane* (daerah partisi) bertipe Rect. Terdapat pula kelas LevelManager yang menampilkan level permainan dan mengelola *root* 'akar' yakni ruang awal mula permainan.

B. Partisi Ruang Permainan

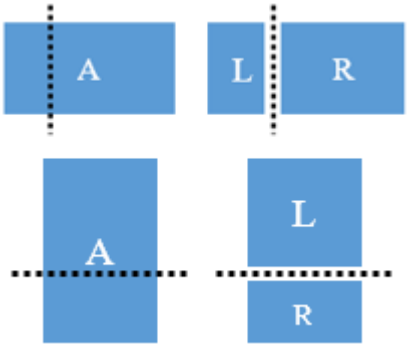
Untuk membagi ruang permainan awal menjadi beberapa partisi menggunakan pohon BSP, terdapat dua tahap yang dijalani pada implementasi, yakni tahap generasi level permainan dan tahap partisi daerah level.

Pada tahap generasi level permainan, akan dilakukan langkah berikut (subrutin GenerateLevel pada LevelManager):

1) Basis 1: Jika level bukan merupakan daun, tidak melakukan apa-apa (keluar dari taha ini).
2) Basis 2: Jika daerah level terlalu kecil (dibandingkan ukuran minimum ruangan yang dispesifikasi) untuk dipartisi, tidak melakukan apa-apa (keluar dari tahap ini).
3) Rekuren: Lakukan partisi pada daerah level (tahap partisi daerah level). Jika berhasil, maka lakukan pemanggilan tahap 1 kembali untuk left dan right dari level.

Saat tahap partisi daerah level, akan dilakukan langkah berikut (subrutin Split pada Level):

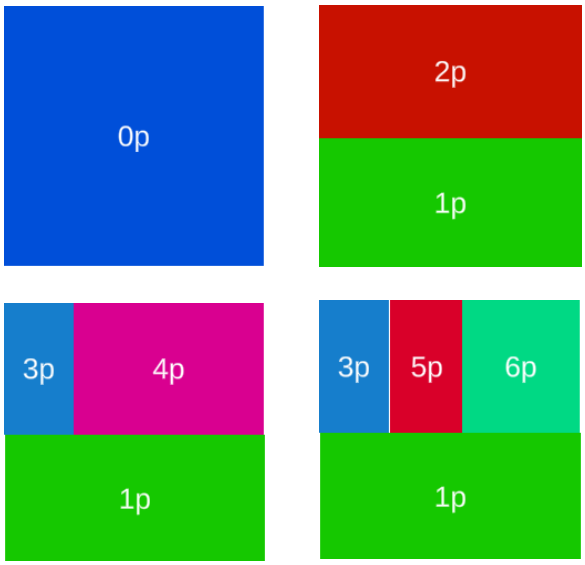
1) Jika level bukan merupakan daun, kembalikan *false* (gagal mempartisi level karena telah dipartisi/bukan merupakan daun).
2) Jika daerah level lebih kecil daripada ukuran minimum yang dispesifikasi, kembalikan *false* (gagal mempartisi level)
3) Lakukan pengambilan arah pemotongan untuk partisi level. Arah pemotongan ditentukan berdasarkan rasio ukuran $\frac{lebar}{panjang}$ atau $\frac{panjang}{lebar}$ dengan *threshold* T tertentu. Jika $\frac{lebar}{panjang} \geq T$, maka pemotongan akan dilakukan secara vertikal. Jika $\frac{panjang}{lebar} \geq T$, maka pemotongan akan dilakukan secara horizontal. Pengambilan nilai *threshold* T diambil lebih besar dari 1 sehingga sebisa mungkin tidak ada ruang yang sangat tipis. Jika semua kondisi tidak memenuhi, yakni $panjang < T \times lebar < T^2 \times panjang$, maka arah pengambilan akan diambil secara acak.
4) Partisi level sesuai arah pemotongan yang sudah ditentukan. Setiap partisi akan menjadi level baru yang di-*assign* untuk left dan right dari level ini. *Offset* pemotongan akan diambil secara random dengan ukuran paling kecil ukuran minimum ruangan yang terdefinisi.



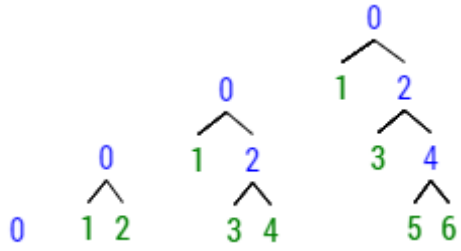
Gambar 3.1 Ilustrasi pemotongan level sesuai arah partisi horizontal dan vertikal
Sumber: Dokumentasi Pribadi

5) Kembalikan nilai *true* (berhasil mempartisi level).

Contoh tahap-tahap partisi ini dapat dilihat pada gambar berikut:



Gambar 3.2 Ilustrasi dua tahap partisi ruang permainan.
Sumber: Dokumentasi Pribadi



Gambar 3.3 Representasi pohon BSP yang terbentuk dari setiap tahap pada contoh gambar 3.2.
Sumber: Dokumentasi Pribadi dengan bantuan tree generator. Miles Shang, *Syntax Tree Generator*. http://mshang.ca/syntree/ (accessed Dec. 11, 2021).

Pada contoh tersebut, ruang permainan dimulai pada level 0. Tahap pertama generasi memungkinkan untuk tahap kedua partisi level 0 menjadi dua level 1 dan 2. Tahap pertama selanjutnya berhenti untuk mempartisi level 1 karena sudah

terlalu kecil, namun akan memproses level 2 dengan tahap kedua partisi menjadi level 3 dan 4. Tahap pertama selanjutnya berhenti untuk mempartisi level 3 karena sudah terlalu kecil, namun akan mempartisi level 4 dengan tahap kedua partisi menjadi level 5 dan 6. Karena pada tahap pertama selanjutnya level 5 dan 6 sudah terlalu kecil untuk dipartisi, maka proses partisi ruang permainan selesai.

Representasi pohon BSP pada gambar 3.3 ini dapat memudahkan pembagian daerah dan pembuatan ruang sehingga dijamin tak ada yang berhimpitan. Representasi inilah yang membuat pembangkitan level secara acak lebih mudah dikontrol sehingga tidak menghasilkan level yang tidak dapat dimainkan. Selain itu, representasi ini juga dapat diubah sedemikian rupa sehingga memiliki kontrol secara artistik dan *game design* yang bagus. Pohon BSP ini juga dapat dikembangkan dengan menerapkan restriksi banyaknya daun, maksimum partisi ruang, dan *fast collision detection* karena keterurutan dari ruang yang dipartisi.

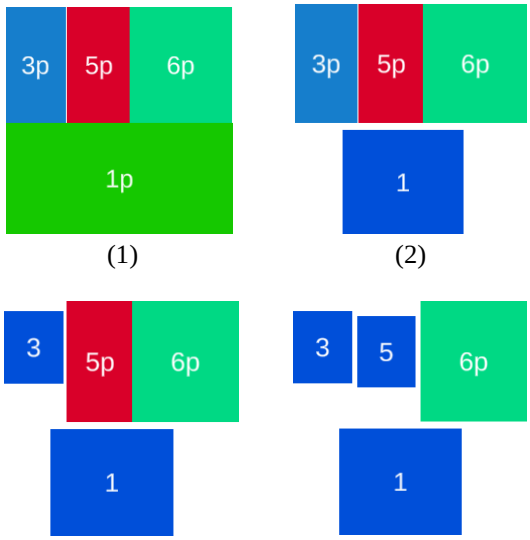
Setelah partisi ruang permainan selesai dilakukan, akan terbentuk daun-daun yang dapat dibuat menjadi ruangan dalam permainan. Pada contoh gambar 3.2 dan 3.3, daun-daun pada ruang permainan ditunjukkan pada nomor 1, 3, 5 dan 6. Daun-daun ini yang kemudian dapat dibuat sebagai ruang permainan.

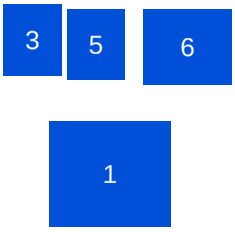
C. Pembuatan Ruang

Setelah mendapatkan daun-daun berisi ruangan yang dapat ditempati, setiap daerah level akan dibuat ruang untuk permainan itu sendiri. Pembuatan ruang permainan dilakukan dengan traversal ke seluruh simpul pada pohon secara rekursif. Langkah-langkah yang dilakukan sebagai berikut:

1) Basis: Jika level merupakan daun, buatlah ruang pada level ini. Pembuatan ruang dilakukan secara acak beserta posisinya dengan ukuran minimum adalah ukuran ruang terkecil yang telah terdefinisi sebelumnya dan maksimum sebesar daerah level itu sendiri.
2) Rekuren: Jika level bukan merupakan daun, lakukan pembuatan ruang secara rekursif pada anak kiri dan anak kanan dari level sekarang.

Contoh pembuatan ruang dapat dilihat pada gambar berikut:





(5)

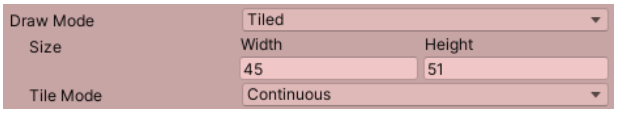
Gambar 3.4 Tahap pembuatan ruang pada partisi gambar 3.2.
Sumber: Dokumentasi Pribadi.

Pada contoh pembuatan ruang pada gambar 3.4, akan dilakukan traversal pada pohon hingga berada pada simpul daun. Untuk setiap simpul daun, akan dilakukan pembuatan ruang secara random, yakni membangkitkan angka acak yang menntukan ukuran dari ruang (minimum ukuran terkecil yang dispesifikasi sebelumnya dan maksimum ukuran dari daerah level itu sendiri) serta posisi dari ruang. Hal ini dapat dilihat pada gambar 3.4 (2), (3), (4), dan (5) untuk masing-masing daun 1, 3, 5, dan 6.

Ketika semua daun telah dibuat masing-masing ruangnya, maka prosedur yang harus dilakukan selanjutnya adalah menampilkan dan menginstansi objek level sesuai dengan pohon BSP yang telah dibuat.

D. Menampilkan Level Permainan

Untuk menampilkan level permainan, terdapat *tile* lantai berupa *tiled sprite* yang bisa ditentukan ukuran *tiling* berdasarkan ukuran masing-masing daun dari pohon. Sprite terlebih dahulu di-*import* degnan ukuran 1 unit (misalnya ukuran gambar adalah 16x16, maka Pixel per Unit adalah 16). Pada Unity, terdapat fitur Draw Mode tile pada Sprite Renderer yang dapat menentukan ukuran tiling.



Gambar 3.5 Ukuran tiling pada bagian Draw Mode di Sprite Renderer.
Sumber: Dokumentasi Pribadi.

Langkah-langkah pada tahap menampilkan level permainan dapat dilakukan dengan traversal dimulai dari level akar sebagai berikut:

1) Basis: Jika level merupakan daun, tambahkan tile baru untuk menggambarkan ruang daun ini. Posisi tile akan diatur pada $(x + \frac{width}{2}, y + \frac{height}{2})$ sesuai dengan representasi Rect dari ruang. Ukuran dari sprite tile diatur menjadi *width* dan *height* dari ruang daun.
2) Rekuren: Jika level bukan merupakan daun, lakukan penambahan tile secara rekursif pada anak kiri dan anak kanan dari level sekarang.

Setelah level permainan semuanya ditampilkan, maka level permainan siap untuk dipakai.

IV. KESIMPULAN

Pembuatan level permainan roguelike dengan menggunakan BSP tree memudahkan untuk membuat level yang *playable*. Dengan menggunakan BSP tree, parameter dapat lebih dikontrol dan banyak hal yang bisa dimanfaatkan dengan efisien. Tiap ruang dijamin pasti tidak ada yang berhimpit dan ruangan dapat diacak ukuran serta partisinya hingga dikondisikan sekecil ukuran yang diinginkan dengan kriteria tertentu. Setiap ruangan terkecil juga dapat dihubungkan dengan mudah secara terpisah. Kuantitas ruang juga dapat dibatasi dengan nilai minimum dan maksimum. Oleh karena itu, pendekatan inilah yang membuat BSP tree lebih menarik dibandingkan pendekatan naif dengan mencoba semua kemungkinan membuat ruangan secara random.

Penulis tentunya menyadari dengan waktu yang cukup singkat ditambah dengan beban untuk dapat mengerjakan UAS dengan baik sehingga banyak hal yang bisa ditambahkan namun tidak sempat untuk dimasukkan. Penambahan fitur pada pohon BSP dapat berupa:

1) Penyambungan antar-ruang dengan membuat koridor untuk setiap daerah pada simpul dalam.
2) Pembuatan restriksi jumlah ruang (daun) yang dapat dibuat dengan nilai minimum dan maksimum.
3) Efisiensi dengan membuat pohon BSPnya menjadi *balanced binary tree*.
4) Pembuatan dinding tepi pada setiap ruangan.
5) *Fast collision detection* hanya dengan mengetahui posisi yang ingin dicari dan *searching* layaknya *binary search tree* ke daun (ruangan) yang diinginkan. Keterurutan pohon BSP membuanya cukup efisien dalam melakukan *searching*.

V. UCAPAN TERIMA KASIH

Dengan selesainya makalah ini, saya mengucapkan terima kasih yang sebesar-besarnya kepada Allah SWT terhadap karunia-Nya sehingga saya bisa membuat makalah ini. Saya juga berterima kasih kepada orang tua dan teman-teman yang telah membantu saya dalam bantuan mental sehingga membuat saya semangat dalam menyelesaikan makalah ini. Terakhir, saya berterima kasih kepada ibu Dra. Harlili, M.Sc. dan bapak Dr. Ir. Rinaldi Munir, MT. selaku dosen yang telah membimbing saya dalam mata kuliah IF2120 Matematika Diskrit.

REFERENSI

[1] H. Fuchs, Z. M. Kedem, and B. F. Naylor, "On visible surface generation by a priori tree structures," *Proc. 7th Annu. Conf. Comput. Graph. Interact. Tech. SIGGRAPH 1980*, no. December, pp. 124–133, 1980, doi: 10.1145/800250.807481.

[2] D. Kushner, *Masters of Doom: How Two Guys Created an Empire and Transformed Pop Culture*. Random House, 2003.

[3] K. H. Rosen, *Discrete Mathematics and Its Applications*, 7th ed. New York: McGraw-Hill, 2012.

[4] R. Munir, *Matematika Disktrit*, 3rd ed. Bandung: Informatika Bandung, 2010.

[5] B. F. Naylor, "A tutorial on binary space partitioning trees," *Comput. Games Dev. Conf. Proc.*, no. August, pp. 433--457, 1998.

[6] Unity Technologies, "Unity - Scripting API: Rect (v2020.3)," 2021. https://docs.unity3d.com/ScriptReference/Rect.html (accessed Dec. 11, 2021).

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, 3 Desember 2021

Amar Fadil
13520103