Pembuatan Peta Permainan dengan BSP

Muhammad Hilman Beyri - 13509047
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13509047@std.stei.itb.ac.id

***Abstract*—Makalah ini membahas mengenai salah satu teknik dalam pembuatan peta permainan secara dinamik yaitu BSP(Binary Space Partitioning). Pembahasan akan dimulai dengan dasar teori mengenai apa itu binary space partitioning dan kemudian dilanjutkan dengan bagaimana menggunakan binary space partitioning ini untuk menghasilkan peta dinamik. Pembahasan dilengkapi dengan implementasi algoritma dalam bahasa Java dan juga gambar pohon dan gambar daerah partisi. Peta permainan yang dibuat dengan BSP ini nantinya akan berisi daerah-daerah yang terpartisi. Daerah partisi tersebut bisa dilihat sebagai ruangan dalam peta, ataupun juga bisa dilihat sebagai dinding yang tidak bisa dilalui. Pada bagian akhir makalah, disajikan hasil pengujian dengan waktu yang dibutuhkan dan juga kemampuan program dalam menangani kasus yang sangat besar(ukuran peta melebihi 10000)**

***Kata Kunci*—Peta Permainan dinamik, BSP, Binary Space Partitioning**

I. PENDAHULUAN

Salah satu unsur paling penting dalam permainan adalah desain level/peta. Desain peta permainan yang bagus adalah salah satu kunci dalam menarik pemain untuk memainkan game. Biasanya terdapat satu orang khusus untuk merancang suatu peta.

Akan tetapi, desain peta permainan yang dinamik, yaitu peta pemain yang satu berbeda dengan pemain yang lain menawarkan ketertarikan tersendiri. Dengan peta permainan yang dinamik, terdapat banyak replay value karena peta tersebut bisa sangat bervariasi ketika dihasilkan dan juga menawarkan tantangan berbeda dan keunikan tersendiri. Salah satu game terkenal yang menggunakan peta permainan dinamik adalah Minecraft dan infiniminer.

Salah satu strategi yang digunakan untuk menghasilkan peta dinamik adalah dengan menggunakan *binary space partitioning*. Pada makalah ini, pembuatan peta dinamik akan menghasilkan daerah-daerah kotak(partisi) yang bisa digunakan oleh programmer sebagai ruangan dalam gua, ataupun dinding. Pembuatan peta dinamik masih bisa ditingkatkan lagi dengan cara menambahkan koridor atau jalan antar ruangan dan juga penambahan ruangan bukan persegi panjang. Tenik binary space partition yang digunakan pada makalah ini hanyalah pengembangan DFS dalam mencari ruang solusi.

II. DASAR TEORI

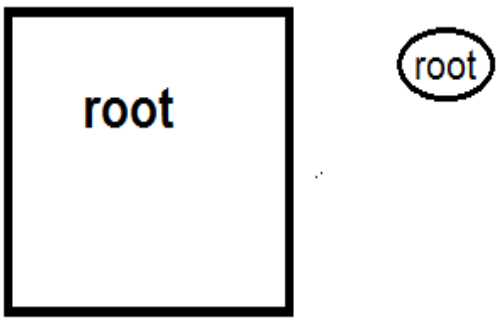
Algoritma Binary Space Partition pada makalah ini sederhananya adalah suatu metode yang secara rekursif membagi suatu ruangan menjadi dua ruangan secara rekursif . Binary Space Partition mempartisi menjadi dua bagian secara rekursif sampai proses partisi tersebut memenuhi suatu persyaratan. Binary Space Partition ini akan menghasilkan tree yang bisa ditelusuri tergantung tujuan penggunaannya. Pada awalnya, binary space partitioning ini diusulkan untuk digunakan pada grafik 3D komputer untuk meningkatkan efisiensi rendering dengan cara menghitung dulu (precomputing) pohon BSP sebelum melakukan operasi low-level rendering. Penggunaan algoritma ini lainnya adalah pada operasi geometric bangun (constructive solid geometry) pada CAD, deteksi tumbukan pada robotika, dan pada game komputer 3D, dan pada bidang lainnya yang terkait dengan penanganan ruangan yang kompleks.

III. PEMBUATAN PETA DINAMIK

A.Algoritma Dasar

Berikut adalah algoritma pembuatan peta dinamik.

1. Inisialisasi daerah yang ingin dipartisi dan panjang dan lebar minimum partisi yang diinginkan.

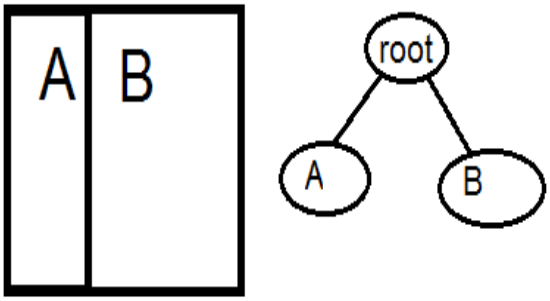


Gambar 1 Daerah yang ingin dibuat peta dinamik dijadikan simpul akar

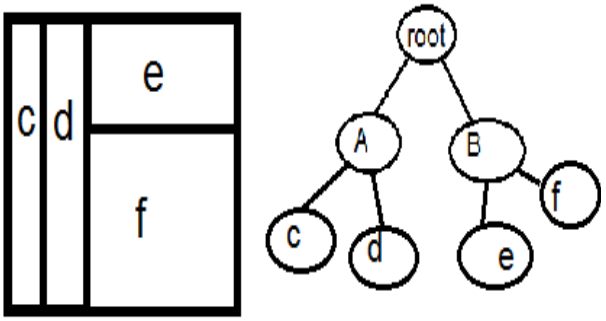
2. Cek apakah daerah tersebut masih bisa dipartisi(dua daerah hasil partisi mempunyai panjang dan lebar melebih panjang dan lebar minimum yang ditetapkan di langkah 1). Jika

bisa, lanjut ke langkah 3, jika tidak lompat ke langkah 4.

3. Partisi daerah tersebut menjadi dua bagian seperti diilustrasikan oleh gambar 2. Partisi bisa dilakukan secara horizontal ataupun vertical. Daerah hasil partisi harus memenuhi panjang dan lebar minimum(menghindari ruangan yang terlalu kecil). Jika partisi sukses, ulangi langkah dua untuk mempartisi kembali kedua daerah hasil tersebut. Begitu kembali dari rekursif(proses rekursif partisi dua daerah tersebut sudah selesai), tambahkan kedua daerah hasil partisi tersebut sebagai daerah anak dari daerah ini.

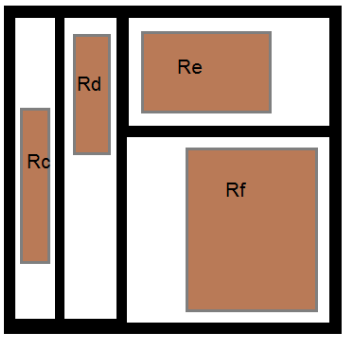


Gambar 2 Partisi pertama(vertikal)



Gambar 3 Partisi kedua(vertikal untuk kiri dan horizontal untuk kanan)

4. Jika daerah tersebut tidak bisa dipartisi lagi. Buat ruangan dari partisi(ruangan bisa sama dengan daerah partisi atau bisa lebih kecil dari partisi. Berguna untuk memberi ruangan kosong dipeta sehingga peta tidak penuh dengan ruangan yang saling berhimpit) tersebut dan tambahkan ke ruang solusi. Daerah partisi ini berarti adalah daun dari pohon BSP.



Gambar 4 Hasil peta dinamik

Didapatkan peta dinamik seperti pada gambar 3.

B. Implementasi prosedur partisi dalam Java

```
void
createChildPartition(LinkedList<Partiti
on> qp) {
        //arah pemotongan(horizontal
atau vertikal)
        boolean dir =
random.nextBoolean();
        if(((dir ? width:height)-
MIN_PART_SIZE) < MIN_PART_SIZE) {
            createRoom();
            qp.push(this);
            return;//terlalu kecil
untuk di partisi
             }
        Partition par1;
        Partition par2;
        int val;

        if(dir==true) {//vertikal
            val =
randomize(left+1,left+width-1);
            val = ((val-
left>=MIN_PART_SIZE && left+width-
val>=MIN_PART_SIZE) ? val :
left+MIN_PART_SIZE );
            par1 = new
Partition(top,left,height,val-left);
            par2 = new Partition(top,
val, height, left+width-val);
        }else{//horizontal
            val =
randomize(top+1,top+height-1);
            val = ((val-
top>=MIN_PART_SIZE  && top+height-
val>=MIN_PART_SIZE) ? val :
top+MIN_PART_SIZE );
            par1 = new
Partition(top,left,val-top,width);
            par2 = new
Partition(val,left, top+height-val,
width);
        }
        par1.createChildPartition(qp);
        par2.createChildPartition(qp);
    }
```

Pada kode sumber diatas, kelas Partition memuat informasi:

1. top. Kordinat pojok atas daerah
2. left. Kordinat pojok kiri daerah
3. height. Panjang daerah
4. width. Lebar daerah
5. room. Ruangan di dalam partisi tersebut. Variabel ini hanya terisi jika partisi ini adalah salah satu daun dari pohon BSP. Pada satu partisi, hanya terdapat satu ruangan.

Konstruktor kelas partition menerima top, left, height, dan width berurutan.

Variabe Qp yang bertipe LinkedList<Partition> adalah variabel yang memuat solusi atau daerah partisi yang menjadi daun di pohon BSP(ditandai dengan tidak punya daerah partisi anak). Tipe data LinkedList digunakan karena operasi penambahan lebih banyak dibandingkan dengan operasi akses elemen tertentu pada list sehingga bisa meningkatkan performa program.

Fungsi Randomize pada kode sumber diatas menghasilkan bilangan acak diantara parameter satu(inklusif) dan parameter kedua(inklusif).

Prosedur createRoom hanya dipanggil jika daerah tersebut sudah tidak bisa dipartisi lagi(terlalu kecil).

Pada kode sumber diatas juga ditunjukkan penentuan panjang dan lebar daerah partisi dengan menggunakan bilangan acak val yang berisikan kordinat dimana partisi dilakukan. Jika partisi dengan menggunakan bilangan val tersebut menghasilkan panjang/lebar salah satu daerah partisi dibawah bilangan minimum yang sudah ditentukan ssbelum, maka panjang/lebar (tergantung dari arah partisi) salah satu daerah partisi otomatis menjadi sama dengan bilangan minimum tersebut.

C. Implementasi prosedur membuat ruangan dalam Java

Berikut adalah implementasi prosedur membuat ruangan dalam Java.

```
void createRoom() {
        int rwidth =
randomize(MIN_PART_SIZE,width);
        int rheight =
randomize(MIN_PART_SIZE,height);
        int rleft =
randomize(left,left+width-rwidth);
        int rtop =
randomize(top,top+height-rheight);
        this.room = new
Partition(rtop,rleft,rheight,rwidt
h);
    }
```

Kode sumber diatas akan menghasilkan ruangan di dalam partisi yang mungkin luasnya sama dengan daerah partisi. Untuk mencegah kemungkinan ruangan langsung berdempetan dengan ruangan lainny, kode sumber diatas dapat diperbaiki menjadi seperti berikut.

```
void createRoom() {
        int rwidth =
randomize(MIN_PART_SIZE,width)-1;
        int rheight =
randomize(MIN_PART_SIZE,height)-1;
        int rleft =
randomize(left+1,left+width-
rwidth);
        int rtop =
randomize(top+1,top+height-
rheight);
        this.room = new
Partition(rtop,rleft,rheight,rwidt
h);
    }
```

D. Implementasi Program Utama dalam Java

```
public static void main(String[]
args) {
      byte[][] bitmap = new
byte[30][30];
      Partition root = new
Partition(0,0,30,30);
      LinkedList<Partition> leafp =
new LinkedList<Partition>();
      root.createChildPartition(leafp
);
      for(int i=0;i<leafp.size();i++)
{
            for(int
j=0;j<leafp.get(i).room.height;j++) {
                  for(int
k=0;k<leafp.get(i).room.width;k++) {

      bitmap[j+leafp.get(i).room.top]
[k+leafp.get(i).room.left] = (byte)
(i+1);
                  }
            }
      }

      for(int i=0;i<30;i++) {
            for( int j = 0; j < 30;
j++ ) {

      if(bitmap[i][j]==0 ) {

      System.out.print('.');
                  }else{
      System.out.print(bitmap[i][j]);
                  }
            }
            System.out.println();
      }
}
```

Kode sumber diatas adalah implementasi program utama dalam bahasa Java. Untuk merubah ukuran peta dinamik, cukup ubah angka 30 dengan angka yang diinginkan

Baris root.createChildPartition adalah dimana algoritma BSP mulai dijalankan. Setelah algoritma tersebut dijalankan, Setiap ruangan yang berhasil diciptakan dipetakan ke bitmap agar dapat dilihat hasil nya dalam bentuk peta 2D. Kemudian iterasi bitmap untuk mencetak bitmap ke layar.

IV. HASIL

Untuk melihat peta dinamik yang dihasilkan oleh BSP, cukup dengan menelusuri ruang solusi dan menyimpannya dalam bentuk array dua dimensi berukuran sebesar daerah yang dipartisi pertama kali seperti ditunjukkan pada kode sumber implementasi program utama diatas.

Berikut adalah hasil peta dinamik untuk peta berukuran 30x30. Perhatikan bahwa ruangan disimbolkan dengan angka.

```
5555555444444444....3333333...
5555555444444444....3333333...
5555555444444444....3333333...
5555555444444444....3333333...
5555555444444444....3333333...
5555555444444444....3333333...
5555555444444444....3333333...
.222222222222222222222222.....
.222222222222222222222222.....
.222222222222222222222222.....
.222222222222222222222222.....
.222222222222222222222222.....
.222222222222222222222222.....
.222222222222222222222222.....
.222222222222222222222222.....
.222222222222222222222222.....
.222222222222222222222222.....
.222222222222222222222222.....
.222222222222222222222222.....
.222222222222222222222222.....
..............................
..............................
1111111111111111111111111.....
1111111111111111111111111.....
1111111111111111111111111.....
1111111111111111111111111.....
1111111111111111111111111.....
1111111111111111111111111.....
1111111111111111111111111.....
1111111111111111111111111.....
```

Gambar 5 Hasil peta dinamik berukuran 30x30

Berikut adalah hasil peta dinamik untuk peta berukuran 30x30 dengan ruangan yang tidak saling berdempetan.

```
..............................
...............666666.55555555
...............666666.55555555
.999999.777777.666666.55555555
.999999.777777.666666.55555555
.999999.777777.666666.55555555
.999999.777777.666666.55555555
.999999.777777........55555555
.999999.777777................
.999999.777777................
.999999.777777................
........777777.444444.22222222
.888888.777777.444444.22222222
.888888.777777.444444.22222222
.888888.777777.444444.22222222
.888888.777777.444444.22222222
.888888.777777.444444.22222222
.888888.777777........22222222
.888888.777777.333333.22222222
.888888.777777.333333.22222222
.888888.777777.333333.22222222
.888888.777777.333333.22222222
.888888.777777.333333.........
.888888.777777.333333.11111111
.888888.777777.333333.11111111
.888888.777777.333333.11111111
.888888.777777........11111111
.888888...............11111111
.888888...............11111111
..............................
```

Gambar 6 Hasil peta dinamik berukuran 30x30 tanpa ruangan yang berdempetan

V. SIMPULAN DAN SARAN

Waktu untuk menyelesaikan algoritma pembentukan BSP tree dengan peta ukuran 1000x1000 membutuhkan rata-rata 9 milisekon dengan komputer penulis(Core2Duo). Sedangkan untuk peta ukuran 10000x10000 membutuhkan waktu sekitar 250 milisekon. Namun, akibat sifat rekursif algoritma ini, yang tiap rekursif menyimpan keadaan fungsi pemanggilnya didalam memori heap, menyebabkan apabila ukuran peta yang di-*generate* sangat besar maka Java akan mengeluarkan runtime error karena kehabisan memory heap(java.lang.OutOfMemoryError). Pada komputer penulis, program masih bisa berjalan dengan ukuran peta 13000x13000 namun ketika menyentuh angka diatasnya, program mengalami error akibat kehabisan memory heap

Peta dinamik yang sama bisa dihasilkan kembali jika *seed* yang sama diberikan kepada fungsi *random* bawaan dari Java sehingga pemain yang merasa tertarik dengan suatu peta dinamik tertentu dapat menyimpan *seed* tersebut jika ingin meng-*generate* peta dinamik yang sama..

Pembuatan peta dinamik masih dapat digali lagi lebih dalam terutama dalam hal pembuatan koridor atau jalan

antara ruangan, ruangan yang tidak selalu kotak.

REFERENCES

[1] Munir, Rinaldi. 2009. Diktat Kuliah IF3051 Strategi Algoritma. Program Studi Teknik Informatika, Sekolah Tinggi Teknik Elektro dan Informatika, Institut Teknologi Bandung.

[2] http://en.wikipedia.org/wiki/Binary_space_partitioning. Waktu akses: 08-12-11

[3] http://doryen.eptalys.net/articles/bsp-dungeon-generation/. Waktu akses: 08-12-11

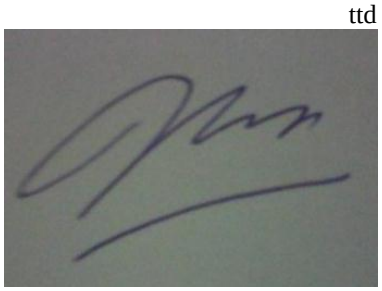
[4] http://stackoverflow.com/questions/4997642/simple-example-of-bsp-dungeon-generation. Waktu akses : 08-12-11

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, 8 Desember 2011

ttd



Muhammad Hilman Beyri
13509047