Kemunculan Monster pada Game Online Menggunakan Algoritma Divide and Conquer

Georgius Rinaldo Winata / 13509030
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
Georgius.Rinaldo@students.itb.ac.id

***Abstrak* — Kemunculan atau *Spawning* monster adalah salah satu unsur penting yang ada dalam permainan berjenis MMORPG atau Massively Multiplayer Online Role Playing Game. Bayangkan apa yang terjadi jika ketika monster dibunuh dan tidak muncul kembali mengakibatkan pemain harus mencari-cari tempat baru setiap kali membunuh monster. Hal ini tentu akan menghabiskan banyak waktu dan membuat pemain lelah dalam menaikkan levelnya.**

Pada karya tulis kali ini user akan mencoba membahas tentang para pembuat game online dalam menggunakan metode *spawning monster* dan kaitannya dengan algoritma *Divide and Conquer*. Algoritma divide and conquer yang bersangkutan adalah pada octree dan quadtree. Algoritma ini biasa digunakan ketika monster terbunuh dalam suatu area map.

***Index Terms*—Brute Force, Divide and Conquer, Collision Detection, Quadtree, Octree**

I. PENDAHULUAN

Di dunia ini banyak sekali jenis permainan (*game)* yang dimainkan dan salah satu yang populer adalah permainan dengan jenis MMORPG atau massively multiplayer online gaming. Dari namanya bisa kita lihat bahwa permainan ini dimainkan secara online dengan menggunakan bantuan internet. Pada permainan ini setiap orang dapat berinteraksi dengan orang lain baik berbincang-bincang, bertarung, berjualan, dll. Sama seperti permainan RPG offline, pada *game* ini pemain akan memainkan sebuah karakter dengan kemampuan dan peran tertentu. Perbedaannya adalah pada MMORPG jalan ceritanya sangatlah panjang dan seolah-olah tidak berakhir dan bisa birentraksi dengan orang lain.

Pada MMORPG pemain akan dituntut untuk menjadi lebih kuat dari waktu ke waktu untuk menyelesaikan misi dan bertarung baik melawan monster ataupun melawan guild (kelompok yang dibentuk dari banyak pemain) lain. Salah satu cara untuk menjadi kuat adalah dengan menaikkan level dari pemain dengan berburu monster. Dengan ini pengalaman pemain akan bertambah dan pemain juga dapat mempelajari kemampuan baru atau meningkatkan kemampuan yang sudah ada. Ketika monster berhasil dikalahkan, monster akan muncul kembali dan tidak akan habis. Hal inilah yang disebut dengan *spawning monster*.

Spawing Monster atau Kemunculan Monster adalah salah satu elemen *game* yang mengatasi pembagian monster dalam tiap area dalam peta. Teknik ini bisa mencakup area dua dimensi dan tiga dimensi tergantung pada *game*-nya. Dengan menggunakan teknik divide and conquer, kemunculan monster pada area di tiap peta menjadi merata dan teratur. Dengan pertimbangan seperti ini, para pemain bisa bermain di daerah-daerah yang telah disediakan di peta tanpa perlu saling berebut tempat bermain. *Spawning monster* banyak sekali digunakan dalam pembuatan game terutama pada game dengan jenis MMROPG atau massively multiplayer online role playing game.

Teknik s*pawning monster* ini juga bermacam-macam. Ada yang dilakukan secara acak, ada juga yang dilakukan secara teratur. Untuk monster biasa, umumnya akan dilakukan spawning secara teratur, sedangkan spawning monster secara random atau acak dilakukan untuk boss monster atau monster-monster yang khusus. Game yang menggunakan algoritma divide and conquer untuk membantu proses kemunculan monster biasa MMORPG yang memiliki tipe lokasi monster tersebar di world map. Ada pula game MMORPG yang sudah disetting supaya monter tepat muncul di area tersebut tanpa perlu divide and conquer karena ketika monster dikalahkan monster tidak akan muncul kembali seperti pada game Dragon Nest dimana pemain akan masuk ke peta dungeon untuk melawan monster yang sudah ditetapkan posisinya.

Contoh MMORPG yang menggunakan *spawning monster*

a. Ragnarok Online
b. Seal Online
c. World of Warcraft
d. Perferct World Online

Gambar 1. Pertarungan dengan monster pada Seal

Gambar 2. Pertarungan melawan boss monster pada Ragnarok Online yang muncul secara random

Spawning Monster ini sendiri berkaitan dengan pengaturan *collision detection* yaitu quadtree pada game dua dimensi, dan octree pada game tiga dimensi. *Collision detection* digunakan untuk mengetahui ketika player telah mengalahkan yang juga menggunakan algoritma divide and conquer. Apabila monster telah dikalahkan, maka perlu dimunculkan monster yang baru untuk diserang pada interval waktu tertentu sehingga area tidak terus kosong. Supaya bisa dilakukan dengan mangkus maka diperlukan algoritma yang baik salah satunya divide and conquer.

Pada makalah ini akan difokuskan mengenai *spawning monster* daripada *collision detection*.

II. LANDASAN TEORI

II.1. DIVIDE AND CONQUER

Divide and Conquer adalah metode pemecahan masalah yang bekerja dengan membagi masalah (*problem*) menjadi beberapa upa-masalah (*subproblem*) yang lebih kecil, kemudian menyelesaikan masing-masing upa-masalah secara independen, dan akhirnya menggabung solusi masing-masing upa-masalah sehingga menjadi solusi masalah semula. Obyek masalah yang akan dibagi bisa bermacam-macam. Pembagian upa-masalah harus memiliki karakteristik yang sama dengan masalah asal, sehingga metode *Divide and Conquer* lebih natural diungkapkan dalam skema rekursif. Tetapi tidak semua fungsi rekursif adalah algoritma *Divide and Conquer*.

Metode *Divide and Conquer* memberikan dua keuntungan. Pertama, metode ini menyediakan pendekatan yang sederhana untuk menyelesaikan masalah yang secara konseptual sulit, seperti permainan Menara Hanoi, dengan cara mereduksi ukuran masalah secara rekursif sampai ia menjadi kasus basis yang mudah diselesaikan secara trivial.

Kedua, bahkan jika solusi masalah sudah diketahui secara pasti (seperti masalah pengurutan, masalah perpangkatan bilangan secara *brute force*), *Divide and Conquer* dapat secara substansial mengurangi biaya (cost) komputasi (kompleksitas). Sebagai contoh, jika masalah semula membutuhkan waktu sebesar $O(n^2)$ untuk masalah yang berukuran n, dan langkah penggabungan solusi dari dua buah upa-masalah membutuhkan waktu $O(n)$, maka algoritma *Divide and Conquer* mengurangi kompleksitas waktu menjadi $O(n \log n)$.

II.2. BRUTE FORCE

Brute Force adalah metode pemecahan masalah yang lempang (*straightforward*, biasanya didasarkan pada pernyataan masalah (*problem statement*) dan definisi konsep yang dilibatkan. Metode ini adalah yang paling sederhana dan paling mudah. Algoritma *brute force* umumnya tidak "cerdas" dan tidak mangkus, karena ia membutuhkan jumlah langkah yang besar dalam penyelesaiannya, terutama bila masalah yang dipecahkan berukuran besar. Kadang algoritma brute force juga disebut algoritma **naïf** (*naïve algorithm*).

Untuk masalah kecil, kesedrahanaan *brute force* lebih dipertombangkan daripada ketidakmangkusannya. Algoritma *brute force* sering digunakan sebagai basis bila membandingkan beberapa alternative algoritma yang mangkus. Dengan membandingkan kompleksitas waktunya bisa dilihat kemangkusan algoritma yang dibandingkan.

Pada masalah yang akan dibahas dalam makalah ini, brute force bukanlah pilihan yang baik. Karena pengolahannya yang lempang dengan memeriksa daerah satu per satu, algoritma *brute force* akan menghabiskan banyak biaya yang mengakibatkan lambatnya proses. Apabila hal ini terjadi terutama pada *online game*, tentunya *game* yang dibuat akan sangatlah lama dan ada kemungkinan terjadi *crash* pada server karena besarnya proses.

Contoh kode untuk memeriksa map dua dimensi pada setiap koordinat:

```
for (int x = 0; x <= map.sizex; x++)
{
        for   (int y = 0; y <= map.sizey; y++)
       {
                do_something();
        }
}
```

Potongan kode di atas sangatlah tidak mangkus. Selain karena harus memeriksa per titik pada koordinat secara lempang, perlu juga ada tindakan dan pertimbangan lebih lanjut yang juga menggunakan metode yang sama. Jika input sangatlah besar, maka akan terjadi pembesaran biaya dan kompleksitas yang signifikan.

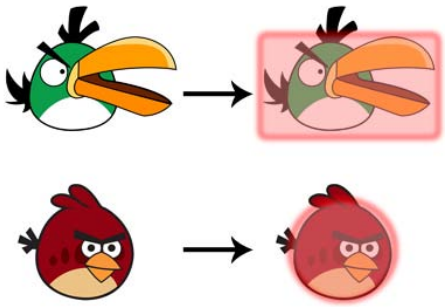
Walaupun dapat dioptimasi, algoritma *brute force* tetaplah bukan pilihan yang baik. Hal ini terjadi karena dalam pembuatan *game* online pasti akan terjadi penambahan atau *upgrade* di setiap elemennya yang membuat objek semakin banyak. Sebagai akibatnya, biaya dan kompleksitas algoritma akan meroket seiring

dengan bertambahnya objek. Terutama untuk metode spawning monster yang harus menggunakan pemeriksaan di setiap area, tentunya akan memakan biaya yang sangat besar.

II. 3. COLLISION DETECTION

Collision detection pada bidang *game programming* adalah suatu cara untuk mendeteksi masalah perpotongan atau persentuhan antar dua objek dalam suatu bidang koordinat tertentu. *Collision detection* ini biasanya akan memberikan informasi tentang waktu kejadian (*time of impact* atau TOI) dan tempat kejadian dari sentuhan antara dua objek. Sebagai akibat dari sentuhan atau *collision* yang terjadi, berdasarkan informasi yang diterima akan diberikan tanggapan terhadap *collision* tersebut.

Pada *collision detection*, dua benda yang saling bersentuhan akan dimodelkan terlebih dahulu. Untuk game-game yang masih dua dimensi dan sederhana, umumnya masih direpresentasikan secara lojik dengan bentuk primitif seperti lingkaran atau segi empat. Bahkan game yang populer dan terkenal pun memungkinkan untuk masih menggunakan permodelan primitif ini seperti misalnya Angry Bird. Apabila game sudah mencapai tiga dimensi, maka akan diperlukan *collision detection* pada bidang tiga dimensi dan menggunakan representasi yang paling mudah digunakan adalah *bounding box* yang setara dengan representasi primitif pada dua dimensi.



Gambar 3. Permodelan *collision detection* dengan bentuk primitif pada game Angry Bird

Dengan menggunakan presentasi primitif seperti segiempat atau lingkaran, tentulah hasilnya kurang maksimal. Hal ini terjadi karena akan ada area dari objek yang sebenarnya bisa bertumbukan jadi terlewatkan atau sebaliknya sebagai akibat dari objek yang akan direpresentasikan tidak mempunyai bentuk yang sama persis dengan bentuk primitive atau hanya mendekati. Hal ini yang biasa membuat para pemain heran atau kesal karena sebenarnya mereka bisa memenangkan suatu permainan yang sedang dimainkan, tetapi menjadi gagal.

Dua objek dikatakan bersentuhan apabila antara dua bentuk yang direpresentasikan saling bersentuhan. Maksud saling bersentuhan adalah apabila jarak antara dua objek sama dengan nol atau tepat bersentuhan. Untuk game yang lebih kompleks, diperlukan perhitungan yang lebih matang karena bisa saja dengan pergantian posisi, hasil pengurangan jarak antara dua objek bisa kurang dari nol atau lebih dari nol, tetapi secara kasat mata sudah bersentuhan. Contoh game yang membutuhkan *collision detection* yang lebih kompleks adalah game fighting. Untuk membuat game dengan *collision detection* yang baik dibutuhkan ***pixel-perfect collision detection,*** yang tidak dibahas pada makalah ini.

Gambar 4. Dua objek saling bersentuhan (collision)

C*ollision detection* ini hampir selalu ada di setiap game karena merupakan elemen yang penting. Tanpa ada *collision detection* mungkin game yang dibuat akan menjadi aneh seperti bisa menembus dinding. Oleh karena itu, perlu adanya *collision detection* dalam pembuatan game yang interaktif. *Collision detection* ini akan menggunakan algoritma divide and conquer khususnya pada quadtree pada dua dimensi dan octree pada tiga dimensi.

II. 4. QUADTREE DAN OCTREE

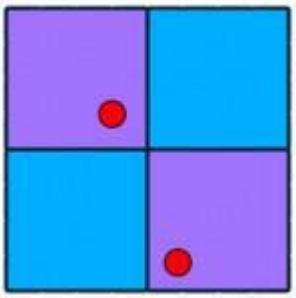
Quadtree dan octree adalah struktur data pohon yang masing-masing memiliki tepat empat dan delapan anak. Pada dasarnya quadtree dan octree menggunakan algoritma *divide and conquer* karena dari tiap daun akan dibagi menjadi beberapa anak dan kemudian dikerjakan sesuai dengan cara yang telah dibuat. Metode ini dipakai untuk pembagian wilayah pada game untuk mempermudah pencarian yang menggantikan algoritma *brute force*. Algortima *brute force* tidak masalah jika jumlah objek yang akan diperiksa sedikit. Akan tetapi, karena pemeriksaan yang dilakukan berskala besar dan game menjadi *real-time*, maka diperlukan algoritma yang lebih baik dalam pemeriksaan objek dalam game yaitu octree dan quadtree yang berbasis *divide and conquer*.

II. 4.1. QUADTREE

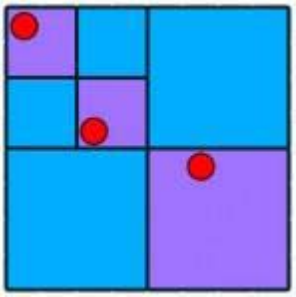
Seperti yang telah dijelaskan sebelumnya, quadtree akan membagi wilayah pengerjaan menjadi empat bagian. Pada game programming, quad tree ini digunakan pada game yang memiliki grafik dua dimensi. Hal ini dikarenakan pada game dua dimensi hanya menggunakan koordinat X dan Y. Oleh karena itu, dengan membaginya menjadi empat bagian sudahlah cukup untuk pemeriksaan objek dalam petanya.

Ketika pembagian dilakukan, pembagian akan dilakukan secara merata horizontal dan vertikal sama besar. Hasil pembagian ini yang akan menjadi anaknya dan secara rekursif dikerjakan dengan algoritma *divide and conquer*. Saat simpul dibagi menjadi empat, akan terbentuk empat simpul baru yang berisi objek yang

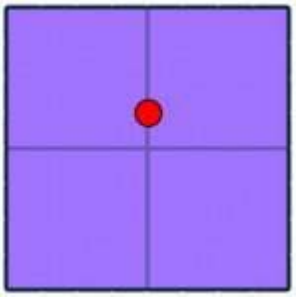
terdapat pada wilayah baru tersebut. Sedangkan untuk objek yang tepat berada pada perpotongan atau sisi akan tetap berada pada simpul induk. Cara inilah yang digunakan untuk pembagian daerah (*region*) untuk objek-objek dalam quadtree. Berikut adalah ilustrasi dari pembagian pada quadtree.



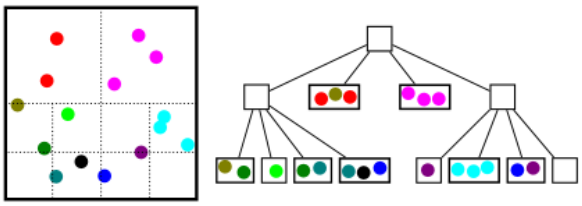
Gambar 5. Objek yang berada dalam region akan dimasukkan ke dalam region (simpul) baru



Gambar 6. Pembagian kembali quadtree ke quadtree yang lebih kecil agar lebih mudah diproses



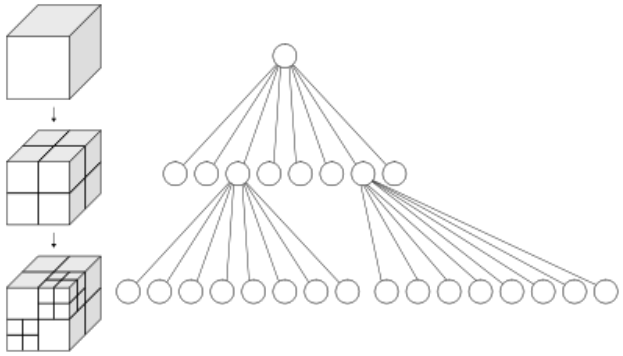
Gambar 7. Objek yang tepat berada pada perpotongan sisi akan tetap berada di simpul induknya



Gambar 8. Hasil pembagian pada quadtree

II. 4.2. OCTREE

Sama halnya dengan quadtree, octree juga akan membagi wilayah pengerjaannya. Hanya saja wilayah pengerjaan (*region)* pada octree akan dibagi menjadi delapan bagian. Octree ini biasa digunakan pada game-game yang memakai grafik tiga dimensi karena grafik tiga dimensi melibatkan tiga koordinat yaitu X, Y, dan Z. Jika daerah hasil pembagian pada quadtree berbentuk persegi, hasil pembagian pada octree akan berbentuk bidang kubus. Dengan pembagian seperti ini, metode ini tentu akan berhadapan dengan banyak perpotongan. Untuk pembagian region pada octree mempunyai aturan yang sama dengan quadtree.



Gambar 9. Pembagian pada octree

III. IMPLEMENTASI PADA GAME ONLINE

Pada makalah ini akan diambil contoh sebuah game online yang spawning monsternya berdasarkan daerah tertentu pada peta. Salah satu game yang akan diambil menjadi contoh adalah Seal Online karena memiliki metode *spawning monster* yang cukup teratur dan merata. Selain itu pada petanya juga mudah diketahui letak-letak monster yang akan dikalahkan.

Gambar 10. Contoh peta pada Seal Online

Dari gambar di atas dapat dilihat pembagian monster pada game Seal Online berdasarkan area (*spot*) tertentu. Biasa dalam permainan ini akan ditempatkan beberapa monster dalam suatu *spot.* Satu spot bisa mencakup sewilayah peta dengan rentang koordinat tertentu yang biasanya terdiri dari empat sampai lima monster. Sama

seperti algoritma *divide and conquer*, akan dibagi wilayah-wilayah (*region*) terlebih dahulu sebelum proses dikerjakan atau diproses.

III. 1. PENGGUNAAN DIVIDE AND CONQUER

Seperti yang telah dijelaskan sebelumnya, metode *divide and conquer* yang digunakan adalah octree sesuai dengan contoh *game* yang diambil yaitu Seal Online. Ketika *game* dimulai, maka pusat pengatur *online game* (*server*) akan mulai melakukan setting terhadap game yang akan mulai dan salah satunya adalah *spawning* monster-monster yang akan dikalahkan. Pembagian wilayah (*region*) akan dimulai dengan batasan-batasan dan modifikasi yang ada. Ketika pembagian wilayah telah selesai, akan diperiksa apakah wilayah tersebut kosong atau layak ditempati monster. Maksud kondisi layak ditempati monster adalah apakah jumlah monster di suatu wilayah telah mencapai batas maksimal atau belum. Jika sudah, maka tidak akan dilakukan penambahan dan pemeriksaan akan diteruskan ke wilayah lain.

Kasus yang dijelaskan sebelumnya adalah kasus pada saat *game* pertama kali mulai, sedangkan pada saat *game* sudah mulai akan memiliki proses yang berbeda. Hal ini terjadi karena pemain sudah mulai berinteraksi satu sama lain dan mulai mengalahkan monster-monster yang ada. Selain itu metode yang digunakan pemain berbeda-beda dalam mengalahkan monster. Metode yang cenderung bermasalah adalah ketika pemain mengumpulkan monster dari wilayah lain untuk dikalahkan secara bersamaan (*mobbing*). Hal ini bermasalah karena bisa menimbulkan kesalahan informasi pada saat pengiriman hasil *collision detection* untuk aksi yang telah dilakukan pemain ketika monster dikalahkan terutama informasi tentang lokasi. Untuk mencegah hal ini, maka digunakan algoritma *divide and conquer* untuk memunculkan kembali monster secara merata dan teratur khususnya menggunakan octree.

Ketika permainan sudah berjalan, pemeriksaan monster akan dilakukan setiap periode waktu terutama ketika ada aksi dari pemain. Pemeriksaan ini bertujuan untuk mengetahui apakah monster telah dikalahkan dan hilang dari peta atau belum. Ketika monster dikalahkan, hanya perlu mengirim informasi untuk mengurangi jumlah monster pada suatu wilayah tanpa perlu informasi yang berhubungan dengan lokasi lainnya. Hal ini terjadi karena dengan algoritma *divide and conquer* kita langsung mengetahui wilayah mana yang kekurangan monster dan perlu dimunculkan kembali.. Tentu hal ini mempermudah penempatan kembali monster. Pada area tertentu akan ada pembatasan jumlah monster yang informasi mengenai total monster dalam suatu wilayah sudah disimpan pada saat memunculkan monster sehingga tidak terjadi *flooding* pada suatu wilayah.

Dalam memunculkan kembali (*respawning*) monster perlu dipertimbangkan beberapa hal penting. Hal yang pertama adalah apakah wilayah yang diperiksa memiliki jumlah monster yang cukup. Apabila belum, bisa ditambahkan monster sampai batas yang diizinkan.Yang kedua adalah *terrain* dari peta. Apakah pada tempat tersebut bisa diakses dalam arti dilalui atau dimainkan (*playable area*). Jika tidak bisa, maka monster tidak akan dimunculkan disana, tetapi di tempat lain. Dalam kasus ini digunakan pemeriksaan dengan octree.

III. 2. ILUSTRASI RESPAWN MONSTER

Pada bagian ini akan diilustrasikan bagaimana penggunaan algoritma *divide and conquer* dengan lebih jelas dalam *spawning monster*. Akan diberikan sebuah peta dalam permainan yang sudah dibagi dengan menggunakan divide and conquer. Selain itu dalam wilayah-wilayah yang telah dibagi juga sudah ditetapkan jumlah monster yang diperbolehkan. Berikut adalah gambar yang dijadikan sebagai contoh. Gambar monster pada peta hanya menunjukkan jenis monster yang berada di peta tersebut.

Gambar 10. Pembagian daerah pada peta dengan koordinat X mendatar ke kanan dan Y membujur

Contoh kasus yang akan diambil adalah pemain yang melakukkan metode *mobbing* untuk mengalahkan monster. Untuk mempermudah akan diberikan tabel mengenai monster yang diambil dari suatu daerah dan jumlah monster maksimum yang diizinkan. Dimisalkan posisi pemain pada daerah (3,3).

Daerah	Jumlah Mob	JMD
2,3	1	2
3,2	2	4
3,4	1	2
4,2	1	3
4,4	1	2

*__Keterangan__:

- Daerah dinyatakan dengan (x,y)
- Jumlah mob adalah jumlah monster yang ditarik oleh pemain dari suatu daerah
- JMD adalah jumlah maksimum monster pada suatu daerah

Ketika monster dikalahkan kondisi tabel akan berubah dan informasi akan dikirimkan untuk memunculkan kembali monster. Berikut adalah tabel ketika monster dikalahkan.

Daerah	JMS	JMD
2,3	1	2
3,2	2	4
3,4	1	2
4,2	2	3
4,4	1	2

*__Keterangan__:

- Daerah dinyatakan dengan (x,y)
- JMS adalah jumlah monster yang ada sekarang
- JMD adalah jumlah maksimum monster pada suatu daerah

Setelah monster dikalahkan, akan diperiksa apakah jumlah monster yang ada telah mencukupi jumlah monster maksimal dalam suatu daerah atau belum. Seperti yang telah dijelaskan sebelumnya, maka akan dimunculkan monster sampai jumlah maksimum dalam suatu daerah. Untuk tempat memunculkan monster ini dilakukan secara acak selama masih dalam suatu daerah dimana monster berada seharusnya. Hal ini tentu lebih menghemat memori jika dibandingkan dengan posisi *respawn* monster yang statis karena perlu ada penyimpanan koordinat sedangkan peta dalam *game* sangatlah banyak.

III. 3. KEMUNCULAN MONSTER KHUSUS

Untuk kasus kemunculan monster khsusus mempunyai perlakuan yang berbeda dari monster biasa. Teknik yang telah dijelaskan sebelumnya adalah teknik untuk memunculkan monster secara acak namun teratur dalam suatu daerah. Kemunculan monster khusus (*boss monster* atau *MVP*) umumnya menggunakan metode yang lain dari monster biasa karena bertujuan untuk mengejutkan dan membuat permainan menjadi menarik. Untuk kasus tertentu *boss monster* memiliki tempat yang statis, akan tetapi hanya sedikit yang menggunakan metode ini. Kemunculan monster khusus ini salah satunya ada pada permainan Ragnarok Online.

Memunculkan monster khusus pun menggunakan algoritma *divide and conquer* sama seperti monster biasa. Cara yang dipraktikkan adalah dengan mencari tempat atau daerah yang strategis dan *playable* dimana hanya ada sedikit player atau bahkan tidak ada sama sekali. Oleh karena itu, untuk mengoptimalkan pencarian, digunakan algoritma *divide and conquer* lalu *solve* yang kemudian dibandingkan hasil pencariannya. Setelah didapatkan hasilnya, maka akan dimunculkan boss monster tersebut secara acak selama masih dalam daerah yang didapat dari hasil pencarian tadi.

Contoh kasus:
Langkah yang dilakukan pertama sama seperti yang sebelumnya, yaitu pembagian peta menjadi beberapa daerah. Setelah peta dibagi-bagi, akan diperiksa di setiap daerah tersebut apakah ada objek dengan tipe pemain. Hasil dari pencarian akan dimasukkan ke dalam senarai atau matriks (bisa dua dimensi atau tiga dimensi tergantung kebutuhan) untuk kemudian dibandingkan.

Misalkan didapatkan hasil dalam tabel yang merepresentasikan sebuah matriks sebagai berikut dimana isi tabel adalah jumlah pemain:

M	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	3	2	2	1	0	0	2	6	0	0
1	0	1	0	0	1	0	0	0	0	0
2	0	0	2	0	0	0	0	0	1	0
3	4	1	0	1	2	0	2	0	0	0
4	0	6	1	0	1	0	0	0	0	0
5	0	2	0	0	0	0	0	1	0	0
6	2	1	0	1	0	0	0	0	0	0
7	0	1	1	2	0	0	0	0	3	0
8	2	0	0	5	3	0	1	0	0	0
9	0	0	0	1	5	0	0	0	2	1

Berdasarkan tabel yang telah dibuat, akan menjadi kebijakan staff game online terutama Game Master (GM) untuk memunculkan boss monster dimana. Ada beberapa pertimbangan yang dipakai, yang pertama adalah tempat yang pertama kali ditemukan kosong tanpa pada tabel. Yang kedua adalah tempat yang paling sepi. Cara ini dilakukan dengan memeriksa daerah sekitar tempat yang diperiksa, misalkan pada koordinat (6,6) yang di daerah sekitarnya tidak ada pemain sama sekali, maka akan dimunculkan boss monster. Pertimbangan ketiga adalah tempat kosong yang strategis yang biasanya berada di paling dekat dengan wilayah tengah map atau pojok map yang masih merupakan daerah yang dapat dijangkau pemain atau *playable area*. Pertimbangan ini cukup boros karena perlu pencarian yang lebih detail dengan algoritma yang lebih kompleks tergantung pada awal pencarian. Dan masih banyak lagi pertimbangan lain. Walaupun begitu, langkah pembuka pertimbangan ini semua adalah algoritma *divide and conquer* untuk pemeriksaan yang optimal.

Gambar 11. Tempat yang sering dimunculkan boss monster pada Seal Online ditandai dengan X

IV. Analisis Hasil penerapan

Berdasarkan hasil pengamatan dari penerapannya pada beberapa game online, algoritma divide and conquer sangat membantu dalam penyelesaian masalah kemunculan monster. Hal ini bisa dilihat dari saat kita bermain langsung. Dengan penggunaan algoritma ini masih bisa menyediakan permainan yang *real-time* tanpa keterlambatan atau error. Selain itu bisa dilihat dengan kemunculan monster yang rapi, teratur, dan merata di seluruh peta. Algoritma ini cukup baik diaplikasikan dan cukup mangkus untuk menunjang permainan yang memiliki banyak sekali objek.

Algoritma *divide and conquer* dipilih karena dianggap jauh lebih mangkus dibandingkan dengan algoritma lain yang tersedia. Selain itu dengan algoritma *divide and conquer* dapat menjaga performa permainan jika dibandingkan dengan algoritma lain. Salah satu yang biasanya dipakai yaitu *brute force* merupakan pilihany ang buruk. Hal ini terjadi karena akan digunakan dalam input dengan skala besar. Seperti yang kita ketahui mengenai algoritma *brute force*, semakin banyak objek atau besar input maka semakin besar pula biaya (*cost*) dan kompleksitas waktunya. Jadi, Algoritma *divide and conquer* bisa dikatakan sudah cukup tepat untuk digunakan dalam kasus *spawning monster* pada game online.

V. KESIMPULAN

Setelah sekian banyak membahas mengenai algoritma *divide and conquer* dan aplikasi serta contoh-contohnya, dapat disimpulkan beberapa hal sebagai berikut:

1. Algoritma *divide and conquer* dapat digunakan, bahkan menjadi elemen yang cukup penting dalam pembuatan game online
2. Untuk penyelesaian hal yang besar seperti masalah kemunculan monster pada game online, dibandingkan dengan metode *brute force*, algoritma *divide and conquer* lebih mangkus dan hemat biaya karena hanya sedikit informasi yang dibutuhkan untuk dicatat dan dikirimkan jika dibandingkan dengan algoritma lain yang perlu banyak penyimpanan variable untuk informasi
3. Algoritma *divide and conquer* yang dipakai dalam pembuatan game tidak hanya mengenai *collision detection*, tetapi juga digunakan untuk *spawning monster* yang rapi, teratur, dan merata serta tanpa mengganggu performa game

Referensi

http://en.wikipedia.org/wiki/Octree diakses pada 12/5/2012

http://en.wikipedia.org/wiki/Quadtree diakses pada 12/5/2012

http://www.euclideanspace.com/threed/animation/collisiondetect/index.html diakses pada 12/5/2012

http://www.kyleschouviller.com/wsuxna/quadtree-source-included/ diakses pada 12/5/2012

http://www.toymaker.info/Games/html/collisions.html diakses pada 12/5/2011

http://www.videotutorialsrock.com/opengl_tutorial/collision_detection/text.php diakses pada 12/5/2011

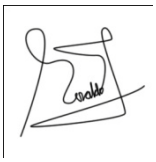
Munir, Rinaldi. 2009. *Diktat Kuliah Strategi Algoritmik IF3051*. Departemen Teknik Informatika ITB

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 2011

ttd



Georgius Rinaldo Winata / 13509030