Pembuatan *Games Landscape* dengan *Matrix Tiling*

Rido Ramadan, 13509049
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13509049@std.stei.itb.ac.id

Abstrak*—*Lanskap* (landscape) *merupakan suatu area dalam permainan* (video games) *di mana pemain dapat melakukan seluruh interaksi yang dimungkinkan oleh permainan tersebut. Lanskap bisa berupa bentuk sederhana seperti level, stage, dll, atau bentuk kompleks seperti perkotaan, pedesaan, pegunungan,* dungeon*, dll. Lanskap sederhana dapat dibangun dengan memanfaatkan matrix dan* array*, di mana setiap elemen menandakan suatu sub-lanskap yang akan digunakan.

***Kata kunci*—landscape, video games, matrix, array.**

I. PENDAHULUAN

Perkembangan dunia digital sangatlah pesat, di mana hampir seluruh perangkat yang bersifat analog mulai tergantikan dengan perangkat digital. Oleh karena sifat benda digital yang menyimpan hasil dalam bentuk data digital, maka data tersebut sangatlah *portable* dan *easily-transferable*. Intinya, data sangatlah *user-friendly* dan *device-friendly*.

Yang menerima imbas dari perkembangan dunia digital ini sangatlah banyak, salah satunya adalah permainan (*games*). Dahulu, permainan sifatnya fisik, seperti kartu, bola, dll. Semakin maju zaman, permainan menambah haluannya ke *dedicated device* seperti konsol permainan. Semakin maju lagi, komputer pun mengalami fungsional baru dari sekedar alat perkantoran menjadi perangkat multimedia. Dari perkembangan-perkembangan yang terjadi, *video games* mulai menyebar.

Video games saat ini sudah memiliki banyak varian dan *genre*. Sebagian besar genre video games sekarang menggunakan latar belakang yang hidup, seperti pemandangan alam atau sebagainya. Pemandangan/latar belakang inilah yang disebut lanskap. Lanskap bisa berbentuk lanskap 3 dimensi atau lanskap 2 dimensi. Lanskap 3 dimensi banyak diterapkan pada permainan yang berbasis 2 dimensi, untuk faktor estetika.

Lanskap bisa dibuat dengan berbagai metode. Salah satu dari sekian metode untuk membuat lanskap, metode yang sederhana adalah dengan metode *matrix tiling*, di mana matrix menjadi prosedur pemanggil area yang diinginkan.

II. DASAR TEORI

A. Array Data Structure

Array data structure, biasa dikenal dengan istilah array, merupakan salah satu tipe primitif yang ada pada berbagai bahasa pemrograman. Array merupakan suatu kumpulan data dengan tipe yang sama yang menempati suatu alokasi memori dalam komputer. Menurut Dr. Ir. Inggriani Liem dalam bukunya *Diktat Algoritma dan Struktur Data* menyatakan bahwa "Array adalah sekumpulan koleksi objek yang diorganisasi secara kontigu, artinya memori yang dialokasi antara satu elemen dengan elemen lainnya mempunyai address yang berututan." Array biasa juga disebut dengan istilah larik.

Array ditandai dengan kesoliteran data tiap ruang, dan tiap ruang data dikenal dengan menggunakan indeks ruang array. Dalam satu array, tidak dimungkinkan adanya perbedaan tipe. Berikut adalah contoh deklarasi array dalam notasi algoritmik dan bahasa Pascal: `bilangan : array [1..10] of integer`. Dari syntax berikut langsung dapat dipastikan bahwa alokasi memori yang tersedia disediakan untuk elemen bertipe integer (bilangan bulat).

10	12	3	5	3	2	1			
1	2	3	4	5	6	7	8	9	10

Ilustrasi sebuah array of integer alokasi 10 memori

Sebuah elemen pada array diakses dengan memanggil nama variabel array dan indeks terkaitnya, seperti berikut: `A[1]` akan memanggil elemen array A di ruang indeks 1, dan seterusnya. Pengisian array boleh saja tidak kontigu, tetapi alokasi memori yang tersedia pada array pastilah kontigu.

Array pada umumnya bersifat statik, di mana jika sudah diberikan alokasi memori di dalam sebuah proses, total alokasi itu tidak bisa dikurangi atau ditambah. Biasanya, untuk mencegah kurangnya alokasi memori pada program, programmer membuat alokasi memori array yang berlebihan. Makanya, array biasanya tidak digunakan dalam pembuatan program yang alokasinya sangat fluktuatif, karena bisa terjadi pemborosan memori atau defisit memori saat dijalankan.

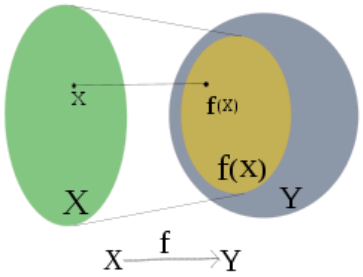
Pada bahasa pemrograman tertentu, alokasi memori array mungkin dilakukan pada saat runtime. Alokasi memori array yang seperti ini disebut array dinamik.

Array dikenali dengan adanya dimensi array. Dimensi yang dimaksud adalah berapa banyaknya sub-array dalam suatu array yang bersangkutan.

B. Fungsi

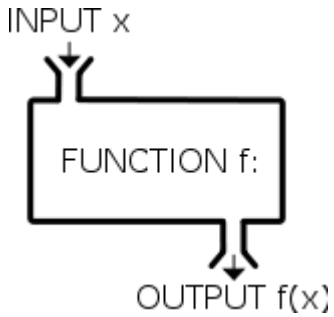
Fungsi adalah suatu ekspresi matematika yang berguna memetakan suatu nilai kepada nilai lainnya. Daerah nilai yang dipetakan disebut domain, daerah hasil pemetaan disebut dengan range. Karena sifatnya yang memetakan nilai, tidak mungkin satu elemen dalam domain memiliki dua atau lebih bayangan, sedangkan satu bayangan boleh memiliki satu atau lebih elemen di domain.

Fungsi bisa direpresentasikan dalam berbagai bentuk, seperti himpunan relasi, diagram, koordinat kartesian, koordinat polar, atau matriks.



Domain dihubungkan melalui fungsi ke range dalam diagram

Dalam pemrograman, fungsi didefinisikan sebagai sekumpulan proses yang memetakan nilai input menjadi nilai output. Jadi, adalah suatu hal yang jelas bahwa fungsi harus memberikan nilai balikan dari suatu parameter input tertentu. Berbeda dengan prosedur yang tidak wajib memberikan nilai balikan, prosedur bisa dilakukan tanpa harus ada parameter input. Fungsi dalam pemrograman layaknya sebuah mesin penghasil output dari input.



Ilustrasi kerja fungsi dalam pemrograman

C. Matriks

Matriks dalam matematika adalah sekumpulan bilangan tertentu yang diurutkan berdasarkan array 2 dimensi. Matriks selalu diidentifikasi dengan besar dimensinya, biasanya matriks selalu disebut matriks berukuran *m* x *n*.

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

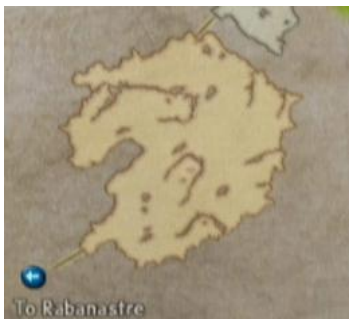
Matriks bisa digunakan sebagai alat untuk menunjukan hasil dari pemetaan suatu fungsi, di mana sekumpulan himpunan relasi dijadikan sebagai koordinat indeks dari matriks yang bersangkutan. Setiap anggota himpunan pasangan relasi yang tersedia diberi kode 1 dan yang bukan anggota himpunan diberi kode 0.

D. Landscape

Lanskap (Landscape) yang dimaksud di sini adalah lanskap dalam sebuah permainan. Lanskap merupakan suatu daerah di mana ada sekumpulan objek yang menjadi latar belakang dari suatu permainan. Di dalam lanskap, avatar (istilah untuk objek yang dimainkan oleh pemain/*user*) bisa bergerak dan berinteraksi baik dengan objek lainnya maupun dengan lanskap itu sendiri.

Lanskap bisa berbentuk macam-macam, dari yang sederhana sampai sangat kompleks. Lanskap sederhana contohnya adalah stage dan level, sedangkan lanskap kompleks contohnya adalah sebuah perkotaan atau pedesaan, lengkap dengan gedung, pepohonan, NPC (*Non-playable Character*), dan objek-objek lainnya.

Lanskap haruslah memiliki batasan ruang. Pemain hanya bisa berinteraksi dengan objek-objek tertentu yang berada dalam daerah tertentu di dalam lanskap. Jadi, tidak mungkin bagi pemain untuk menembus batas lanskap. Fungsi dari batasan ruang dalam lanskap adalah untuk memberikan kesan nyata akan suatu lanskap kepada pemain.



Contoh peta batasan ruang gerak dalam sebuah *games*

Kesan yang ditimbulkan dari batasan ruang akan memberikan impresi yang kuat kepada pemain, di mana lanskap yang tidak interaktif dan kurang realistis akan menimbulkan persepsi yang sangat buruk di mata permain. Sebagai contoh, jika pemain memainkan suatu tokoh yang berada di tebing yang curam, pengelolaan lanskap yang baik akan melakukan salah satu diantara kedua hal berikut, antara menimbulkan efek jatuh ke jurang, lalu *game over*, atau membuat tokoh tidak bisa berjalan melewati batas tebing, seolah-olah menabrak sebuah dinding tak terlihat. Sedangkan penanganan lanskap yang buruk untuk kasus di atas misalnya adalah memperbolehkan tokoh berjalan di udara. Hal yang lebih parah misalnya tokoh bisa berjalan menembus dinding manapun. Di mata pemain, hal itu bisa dianggap sebuah *glitch/bug* yang bisa dieksploitasi, atau menimbulkan kesan pembuat *game* tidak bersungguh-sungguh

membuatnya.

Ada beberapa unsur yang harus dipenuhi dalam pembentukan lanskap yang baik, antara lain latar belakang, objek-objek sekitar, partikel dan *collision detection*.

Latar belakang bisa berupa langit, pemandangan yang jauh di sana, dinding-dinding di sekitar ruangan, langit-langit, dan sebagainya. Latar belakang adalah pembangun suasana jauh yang tidak bisa dicapai. Walaupun berfungsi sebagai penghias, latar belakang adalah hal yang vital.

Objek-objek sekitar bisa sangat bervariasi, tergantung tema dari lanskap tersebut. Misalnya untuk suatu lanskap dengan tema pedesaan 3D, maka dengan latar belakang langit, objek-objek sekitarnya bisa berupa pepohonan, bebatuan, rumah, kolam, ember, dan sebagainya. Objek berperan dalam membangun tema dan suasana dari suatu alam. Misalnya, untuk tema desa terbakar, objek-objek yang ada di area tersebut haruslah barang-barang dan rumah yang telah terbakar, tidak utuh, dibantu dengan latar belakang langit yang semakin memerah akibat pijaran api. Semakin dekat kesan yang ditimbulkan terhadap suatu tema, semakin dalam impresi yang dihasilkan kepada pemain.

Partikel merupakan objek-objek kecil yang bersifat memberi efek pada jalannya permainan. Efek yang ditimbulkan memberikan efek yang hidup atau efek yang wah pada pemain. Contoh dari partikel adalah efek-efek seperti api, sinar, percikan, hujan, asap; objek bergerak seperti peluru atau kupu-kupu, dan lain-lain. Tanpa adanya kehadiran partikel, lanskap akan terlihat sangat statis dan kurang interaktif.

Keselarasan tema padang pasir dengan latar belakang langit, objek bebatuan dengan pepohonan kering, dan partikel asap

C*ollision detection* adalah suatu sistem algoritma mengenai momentum dan impuls yang mengecek terjadinya suatu tumbukan antara dua buah objek atau lebih. *Collision detection* ini digunakan untuk menyimulasikan interaksi antarobjek, misalnya antara padat dan padat, padat dan cair, cair dan cair. Hal ini penting diaplikasikan dalam 3D dan 2D *video games*, karena tanpanya, tokoh dan objek-objek bergerak bisa bergerak menembus dinding, berjalan melewati jurang laut, atau sungai, atau berjalan vertikal mendaki tebing. Fungsi dari collision detection adalah membatasi ruang pergerakan objek yang berada di dalam lanskap dan memberi kesan realistik. Untuk masalah tokoh menabrak dinding, pada sebagian besar *video games*, respon terhadap adanya tumbukan/tabrakan terhadap dinding biasanya dilakukan dengan mengulangi pergerakan terakhir tokoh tersebut tanpa mengubah vektornya atau membelokkannya ke arah yang rasional.

Orang melayang menunjukan tidak adanya *collision detection*

Collision detection sederhana bisa dilakukan dengan *pointing per pixel*. Bila suatu objek dinamik bergerak menuju suatu daerah yang diberi deteksi, maka respon terhadap tabrakan tersebut akan segera dilaksanakan. Collision detection macam ini biasa dilakukan pada permainan 2D yang sederhana. Pada level permainan 3D, objek-objek 3D sudah dilangkapi dengan *helper* dan *vertex* yang sudah siap diset untuk melakukan *collision detection*.

III. HUBUNGAN ANTARA MATRIKS DAN LANSKAP

A. Sub-Landscape

Lanskap utuh merupakan suatu integrasi antara koleksi objek-objek kecil satu sama lain. Terkadang, akan ada beberapa area yang memiliki tema yang sama, namun *layout* area yang berbeda. Walaupun layoutnya berbeda, biasanya ada kemiripan/kesamaan antar satu area dengan area lainya.

Kemiripan yang terjadi antar satu area dengan area lainnya ini ditandai dengan adanaya kemiripan koleksi objek yang bersangkutan, misalnya untuk suatu area hutan, map hutan 1 dipenuhi dengan pepohonan dan semak belukar, sedangkan pada map hutan 2, pepohonan hanya ada di pinggir hutan dan ditengah terbentang suatu danau yang luas. Kemiripan objek yang ada pada keduanya adalah objek pepohonan. Bayangkan bila diambil suatu area kecil berupa daerah berpohon, maka area kecil berpohon ini bisa digunakan baik untuk map 1 maupun map 2. Area kecil yang membangun suatu lanskap utuh ini disebut dengan sub-lanskap. Gabungan suatu sub-lanskap yang baik akan menghasilkan suatu lanskap yang baik pula. Dibutuhkan ketelitian dan kesabaran untuk membangun sebuah lanskap yang bagus, karena pemilihan dan pemosisian sub-lanskap akan sangat mempengaruhi persepsi pemain. Suatu lanskap yang dibangun terlalu kotak-kotak akan terlihat kaku dan tidak indah dilihat mata.

Contoh sebuah sub-lanskap dengan tema air

B. Fungsi Matrix Tiling

Pada subbab sebelumnya telah disebutkan bahwa dengan memanfaatkan potongan kecil dari lanskap yang disebut sub-lanskap, berbagai variasi lanskap bisa dibangun. Jika sub-lanskap tersebut berbentuk sebuah gambar yang mudah diakses, maka proses yang diperlukan dalam membuat lanskap yang diperlukan adalah dengan cara memanggil gambar yang bersangkutan dan tempelkan gambar yang satu dengan gambar lainnya sehingga membentuk lanskap yang utuh.

Membuat lanskap sederhana dengan menggabungkan sub-lanskap satu dengan yang lainnya berfungsi untuk menghemat penggunaan memori yang tersedia. Ditinjau dari mana hematnya? Bayangkan untuk satu map dengan besar area 640 pixel x 640 pixel sepenuhnya berisi air yang membentuk lanskap samudera. Coba bandingkan dengan satu map yang dibangun dari 10 x 10 buah sub-lanskap berukuran 64 pixel x 64 pixel yang dipanggil melalui suatu *source code*, manakah yang lebih hemat? Jika dihitung-hitung, lanskap kedua memiliki besar file kira-kira 1/10 kali lanskap pertama + besar file *source code*. Jika akan dibuat sepuluh buah map utuh seperti itu, maka pemborosan memori yang terjadi akan sangat besar. Sedangkan, dengan menggunakan source code, satu source code dan 1 gambar yang diulang-ulang bisa menghasilkan lebih dari satu buah map. Ini merupakan metode optimasi memori yang cukup baik. Metode seperti ini biasa dilakukan pada pemrograman Flash dengan bahasa ActionScript. Metode ini disebut dengan metode *matrix tiling*.

Matrix tiling adalah sebuah metode yang memungkinkan bagi *landscape designer* untuk merancang dan membuat lanskap hanya dengan memanggil setiap bagian sub-lanskap melalui sebuah matriks. Cara pengaksesan suatu sub-lanskap hanya dengan menuliskan kode sub-lanskap ke dalam sebuah matriks. Contoh source code dalam bahasa ActionScript sebagai berikut:

```
map1 = [
[2,2,2,2,2,2,2,2,2,2],
[2,2,2,0,0,0,0,0,2,2],
[2,2,1,0,0,0,0,0,2,2],
[2,1,1,1,0,0,0,0,1,1],
[1,1,1,1,0,0,0,0,1,1],
[1,1,1,1,0,0,0,0,2,2],
[1,1,1,0,0,0,1,1,2,2],
[1,1,0,0,0,0,0,1,2,2],
[1,1,1,1,2,2,2,2,2,2]];
```

Angka-angka dalam matriks tersebut maksudnya adalah jenis/macam sub-lanskap yang diinginkan dalam membuat sebuah lanskap. Source code tersebut hanyalah sebagian dari source code penuhnya. Source code yang memegang penuh atas seluruh lanskap di atas ada pada contoh berikut.

```
mvWdth = 576;
mvHght = 576;
tileWdth = 64;
tileHght = 64;

tile0 = function () {};
tile0.prototype.pos = 1;
tile0.prototype.barrier = false;

tile1 = function () {};
tile1.prototype.pos = 2;
tile1.prototype.barrier = true;

tile2 = function () {};
tile2.prototype.pos = 3;
tile2.prototype.barrier = true;

map1 = [
[2,2,2,2,2,2,2,2,2,2],
[2,2,2,0,0,0,0,0,2,2],
[2,2,1,0,0,0,0,0,2,2],
[2,1,1,1,0,0,0,0,1,1],
[1,1,1,1,0,0,0,0,1,1],
[1,1,1,1,0,0,0,0,2,2],
[1,1,1,0,0,0,1,1,2,2],
[1,1,0,0,0,0,0,1,2,2],
[1,1,1,1,2,2,2,2,2,2]];
```

Penjelasan pada source di atas, programmer membuat tiga macam jenis sub-lanskap, yakni tile0, tile1, dan tile3. Sub-lanskap tile0 pada contoh di atas dimaksudkan untuk sebuah sub-lanskap padang rumput, sebuah area di mana pemain bisa berjalan dengan bebas. Sub-lanskap tile1 merupakan sebuah sub-lasnkap air, di mana pemain tidak bisa berjalan di atas air. Sub-lanskap tile2 merupakan sub-lanskap tembok bebatuan, jadi tidak mungkin untuk pemain berjalan di sini. Untuk melengkapi spesifikasi di atas, maka terdapatlah syntax berikut.

```
tile0 = function () {};
tile0.prototype.pos = 1;
tile0.prototype.barrier = false;

tile1 = function () {};
tile1.prototype.pos = 2;
tile1.prototype.barrier = true;

tile2 = function () {};
tile2.prototype.pos = 3;
tile2.prototype.barrier = true;
```

Penjelasan dari syntax di atas adalah:

- objek.prototype.pos menunjukan kode frame di mana gambar yang bersangkutan berada. Semakin tinggi nomor frame, semakin tinggi levelnya dalam hal tumpang tindih.
- objek.prototype.barrier menunjukan apakah suatu lanskap itu *free-roamable*. Jika true, maka lanskap tersebut disamakan dengan suatu objek dinding pembatas.

Untuk sebuah landscape sederhana di mana satu map adalah samudera dengan area 576 pixel x 576 pixel, maka source code yang diperlukan akan menjadi seperti berikut:

```
mvWdth = 576;
mvHght = 576;
tileWdth = 64;
tileHght = 64;

tile0 = function () {};
tile0.prototype.pos = 1;
tile0.prototype.barrier = false;

tile1 = function () {};
tile1.prototype.pos = 2;
tile1.prototype.barrier = true;

tile2 = function () {};
tile2.prototype.pos = 3;
tile2.prototype.barrier = true;

ocean = [
[1,1,1,1,1,1,1,1,1],
[1,1,1,1,1,1,1,1,1],
[1,1,1,1,1,1,1,1,1],
[1,1,1,1,1,1,1,1,1],
[1,1,1,1,1,1,1,1,1],
[1,1,1,1,1,1,1,1,1],
[1,1,1,1,1,1,1,1,1],
[1,1,1,1,1,1,1,1,1],
[1,1,1,1,1,1,1,1,1]];
```

Dalam pembuatan lanskap tersebut, source code tersebut hanyalah menjadi library dalam source ActionScript yang ada. Implementasi nyata dari source tersebut belum bisa dihasilkan tanpa adanya fungsi atau prosedur pembuat lanskap dari matriks. Dengan ada fungsi pembuat map, maka library di atas akan ada hasil jadinya.

Fungsi MapMaker yang dibuat berfungsi memanggil library matriks yang tersedia, kemudian menghasilkan dalam bentuk gambar lanksap utuh. Source code ActionScript dari matriks tersebut:

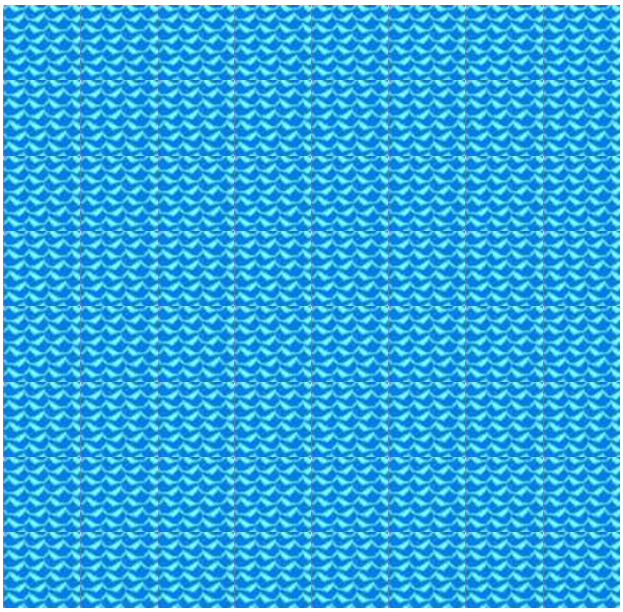
```
function MapMaker (map)
{
    _root.attachMovie("blank", "tiler", d++);
    var mapWdth = map[0].length;
    var mapHght = map.length;
    for (var i = 0; i < mapHght; i++) {
    for (var j = 0; j < mapWdth; j++) {
    var name = "tile "+j+" "+i;
    _root[name] = new _root["tile"+map[i][j]];
    _root.tiler.attachMovie("TileSet1", name,
        i*100+j*2);

     root.tiler[name]. x = (j* root.tileWdth);
    _root.tiler[name]._y = (i*_root.tileHght);
    _root.tiler[name].gotoAndStop(
        _root[name].pos);

    }
    }
}

//Panggil map bernama ocean:
MapMaker(ocean)
```

Hasil dari map ocean setelah diperkecil ukurannya adalah sebagai berikut:



Hasil dari source code *matrix tiling* ocean

Hasil dari matrix tiling tidak hanya berupa background. Dengan memberikan sifat *layer/frame* yang tumpang tindih, maka objek-objek tambahan bisa dengan mudah diletakkan di atas lanskap yang sudah matang. Misalnya lanskap dasar adalah sebuah padang rumput dan danau di tengah. Memberikan suatu MapMaker selanjutnya bisa menjadikan padang rumput tersebut ditumbuhi oleh pepohonan, di mana pepohonan merupakan bagian dari sub-lasnkap lainnya yang dipanggil melalui suatu matriks tiling yang berbeda.

Hierarki dari suatu pembuatan suatu lanskap dideskripsikan sebagai tumpukan *layer* berikut:

Hierarki tumpang tindih layer lanskap, paling dasar fundamental, paling atas opsional

Lanskap layer 1 merupakan sebuah dasar, fundamental dari sebuah pembentukan lanskap 2D. Lanskap ini yang menentukan seluruh jalan atau area yang bisa diakses oleh pemain. Seperti halnya pada *racing games*, layer 1 terdiri atas jalur sirkuit di mana mobil meluncur dan area yang tidak bisa ditembus oleh mobil. Area yang tidak bisa ditembus oleh mobil ini, dipadukan dengan layer 2 sebagai penghias. Bentuk hiasan dalam sirkuit balap bisa

berupa tribun penonton, gedung-gedung membentang di kiri-kanan sirkuit, pepohonan, dan lainnya. Layer 3 dalam sirkuit, bisa berupa langit siang, malam, atau saat hujan.

Pada lanskap 3D, pemanfaatan matrix tiling juga bisa dilakukan, tentunya dengan berbeda *script* dari *source code* ActionScript yang sebelumnya. ActionScript pada Flash tidak mendukung untuk melakukan *multi-layer* pada objek 3D. Untuk lanskap 3D, setiap objek dipisah-pisah menjadi bagian kecil alias sub-lanskap, yang nanti akan dipanggil kembali untuk dijadikan satu kesatuan utuh.

3D *landscape editting* per bagian matriks

C. Random Maps

Random map adalah suatu fasilitas yang diberikan oleh permainan untuk menghasilkan *auto-generated landscape* yang dihasilkan langsung oleh komputer. Pembuatan map secara acak ini biasanya memanfaatkan bagian-bagian sub-lanskap dari suatu lanskap. Sistem generasi ini dibuat berdasarkan kategori, sehingga objek-objek yang berada di dalamnya akan selaras dengan tema dari lanskapnya.

Sistem pengategorian ini dilakukan dengan memberi *library* baru pada tiap kategori nomor berapa saja objek yang diperbolehkan untuk muncul. Misal untuk lanskap dengan kategori ruang bawah tanah/*dungeon*, objek-objek seperti pepohonan atau rumah, terrain berupa padang pasir, tidak dimasukkan ke dalam library. Tanpa adanya sistem kategori, proses *auto-generation* mungkin saja menghasilkan lanskap dengan objek-objek yang tidak relevan dengan tema.

Sistem random maps ini sebagian besar terdapat dalam permainan *Real-Time Strategy* (RTS).

V. KESIMPULAN

Dalam pembuatan *video games* sederhana, lanskap merupakan unsur paling berat yang dikelola oleh suatu *games*. Dengan pengelolaan memori dan penggunaan fungsi yang baik, maka manajemen memori bisa dibuat sehemat mungkin. Lanskap yang dibangun dengan metode *matrix tiling*, memiliki efisiensi memori yang lebih bagus dibandingkan dengan pembuatan lanskap sederhana dengan metode lanskap utuh. Variasi lanskap bisa didapatkan hanya dengan mengubah *library* variasi matriks yang menjadi acuan pembuatan.

DAFTAR PUSTAKA

[1] Liem, Inggriani. 2008. *Draft Diktat Struktur Data* (Edisi 2008). Bandung: Program Studi Teknik Informatika ITB.

[2] Munir, Rinaldi. 2008. *Diktat Kuliah Struktur Diskrit*. Bandung: Penerbit Informatika ITB.

[3] http://en.wikipedia.org/wiki/Array_data_type diakses Minggu, 12 Desember 2010.

[4] http://en.wikipedia.org/wiki/Array_data_structure diakses Senin, 13 Desember 2010.

[5] http://en.wikipedia.org/wiki/Collision_detection diakses Minggu, 12 Desember 2010.

[6] http://id.wikipedia.org/wiki/Fungsi_(matematika) diakses Rabu, 14 Desember 2010.

[7] http://www.scirra.com/forum/viewtopic.php?f=3&t=7344 diakses Minggu, 12 Desember 2010.

[8] http://screwattack.com/blogs/Glitch-Uncharted-2/Glitch-Uncharted-2/1#comment_205356 diakses Senin, 13 Desember 2010.

[9] http://twobrothersandasister.com/wp-content/images/games/ff12/mob/05a.jpg diakses Senin, 13 Desember 2010.

[10] http://www.esperiae.net/ffxii/walkthrough/chapter1.1.php diakses Senin, 13 Desember 2010.

[11] http://www.kuriupa.com diakses Senin, 13 Desember 2010.

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, 12 Desember 2010

Rido Ramadan – 13509049