# Implementasi Konsep Matematika Diskrit dalam Pembangkitan Dunia Acak pada Game Terraria dan Minecraft

Bernhard Aprillio Pramana - 13524074
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: piyips96@gmail.com , 13524074@std.stei.itb.ac.id

***Abstrak*—Dalam perkembangan dunia gim, terutama pada pembangkitan dunia acak implementasi Matematika Diskrit memiliki peran penting. Makalah ini membahas penerapan materi-materi Matematika Diskrit dalam algoritma pembangkitan dunia acak pada gim Terraria dan Minecraft. Dengan menekankan konsep automata seluler, aljabar Boolean, teori bilangan, rekursi, relasi rekurens, kombinatorika, serta graf dan pohon dalam pengaplikasian algoritma pembangkitan dunia acak. Studi ini bertujuan untuk menunjukkan aplikasi Matematika Diskrit dalam perancangan dunia acak pada gim.**

***Kata kunci—algoritma pembangkitan dunia acak, Matematika Diskrit, Terraria, Minecraft, automata seluler, aljabar boolean, graf, rekursi, teori bilangan***

## I. PENDAHULUAN

Perkembangan industri gim digital memunculkan tantangan dalam desain konten yang bervariasi, imersif, dan tidak repetitif. Perkembangan industri gim menghasilkan banyaknya variasi gim, mulai dari genre, cara bermain, keunikan dari gim dan cara gim itu bekerja. Berdasarkan axe.com genre gim terpopuler adalah gim bergenre action serta pada peringkat 8 ada gim bergenre adventure [2]. Terraria dan Minecraft merupan contoh dari gim bergenre action-adventure dengan fokus kepada petualang pada dunianya yang bersifat acak tiap permainan. Berdasarkan laman resmi Minecraft, Minecraft adalah gim sandbox yang terbuat dari susunan balok, dengan konsep pemain dapat dengan bebas berkreasi pada dunia yang dibentuk secara acak [3]. Sama halnya dengan Minecraft, Terraria juga merupakan gim sandbox dengan dunianya yang ditampilkan secara 2 dimensi [4]. Pembuatan dunia pada gim Terraria dan Minecraft memerlukan algoritma *procedural generation*, dalam pembuatan dunianya. Algoritma *procedural generation* adalah teknik untuk membangkitkan konten secara otomatis menggunakan aturan dan fungsi tertentu. Aturan-aturan ini memerlukan peran Matematika Diskrit sehingga dapat membentuk dunia dengan keunikan yang tinggi. Makalah ini mengkaji bagaimana materi Matematika Diskrit seperti automata seluler, teori bilangan, aljabar Boolean, dan graf diaplikasikan dalam sistem pembangkitan dunia acak pada gim Terraria dan Minecraft.

## II. DASAR TEORI

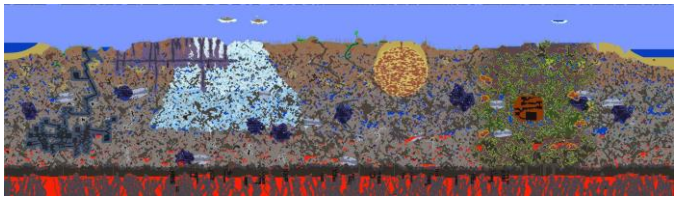
### A. *Pembangkitan Dunia Acak pada Terraria dan Minecraft*

Terraria dan Minecraft merupakan dua gim sandbox bergenre action-adventure yang mengimplementasikan sistem pembangkitan dunia secara acak berdasarkan aturan tertentu yang ditetapkan melalui masukan pemain yang disebut *seed* dalam pembentukan dunia yang akan dimainkan pemain. *Seed* adalah bilangan bulat yang menjadi masukan awal bagi algoritma pembangkitan dunia. Seed ini akan menentukan keunikan dari suatu dunia yang akan dibentuk. Walaupun semua dunia sudah bersifat deterministik, *seed* memastikan setiap dunia memiliki keunikannya sendiri dan tidak mungkin sama dengan *seed* lainnya. Pembuatan dunia 3 dimensi pada Minecraft berbasis bagian perbagian yang disebut dengan *chunk* atau 16×16×384 blok [5]. Sedangkan pada Terraria, pembuatan dunia 2 dimensi didasarkan pada 5 bagian level horizontal, yang dibagi pada kategori langit, permukaan, gua (*cave*), gua besar (*cavern*), dan neraka (*underworld*) [6]. Kedua gim ini menggunakan prinsip Matematika diskrit dalam pembentukan struktur dunianya.

### B. *Automata Seluler dan Conway's Game of Life*

Automata seluler adalah sistem dinamika diskrit di mana ruang dibagi kedalam spasial sel pada grid teratur dan waktu berproses pada setiap tahapan dan berubah berdasarkan waktu sebagai iterasi, status sel itu sendiri, dan status tetangganya. Terdapat 5 unsur penting pada automata seluler, yaitu sel yang merupakan unit dasar dalam ruang spasial. Kondisi atau status yang mendefinisikan atribut suatu sel pada waktu tertentu. Ketetanggaan yang dibagi kedalam dua tipe yaitu lingkungan Von Neumann (empat sel) yang bersentuhan langsung dengan suatu sel, dan ketetanggaan Moore (delapan sel) yang membentuk 9 sel 3×3 grid. Aturan transisi yang mengatur perubahan kondisi suatu sel dalam menanggapi kondisi saat ini dan kondisi tetangganya. Waktu menjadi dimensi waktu yang menentukan kalkulasi pada proses automata seluler [7]. Conway's Game of Life adalah contoh automata seluler dengan dua status (hidup/mati) dan aturan

sederhana berbasis tetangga. Penerapan automata seluler dapat dilihat dari pembentukan dunia Terraria pada sistem pembentukan gua yang kemudian diproses berdasarkan aturan tertentu sehingga memungkinkan pembentukan yang benar-benar teracak dengan hanya mengandalkan aturan logika diskrit [8].



Gambar 1. Contoh dunia Terraria. Sumber: [9]

### C. *Aljabar Boolean dan Logika Proposisional*

Aljabar Boolean menggunakan dua nilai yaitu 0 dan satu atau true dan false. Operasi logika yang digunakan pada aljabar Boolean adalah AND, OR, NOT, NAND, NOR, dan XOR. Logika proposisional dan aljabar Boolean digunakan sebagai dasar untuk menentukan status sel dan tetangganya pada automata seluler.

### D. *Teori Bilangan dan Aritmetika Modulo*

Teori bilangan adalah teori dasar dalam memahami algoritma kriptografi. Teori bilangan, khususnya aritmetika modulo, sangat penting dalam pembangkitan bilangan acak semu. Aritmetika modulo adalah sistem aritmetika yang beroperasi dengan sisa setelah pembagian bilangan dengan suatu bilangan tertentu [10]. Dengan implementasi aritmetika modulo dapat menghasilkan algoritma Linear Congruential Generator. Linear Congruential Generator adalah algoritma yang digunakan untuk menghasilkan angka pseudo-acak atau angka deterministik yang terlihat acak. Seed dalam Minecraft atau Terraria diolah melalui fungsi PRNG seperti ini untuk menentukan bentuk dunia, jenis blok, dan distribusi struktur [11].

### E. *Rekursi dan Relasi Rekurens*

Relasi rekursif adalah ekspresi matematika dalam bentuk deret berdasarkan suku-suku sebelumnya contohnya deret Fibonacci. Dalam konteks informatika, rekursi merupakan suatu fungsi yang dapat memanggil dirinya sendiri atau pemanggilan fungsi secara berulang sampai mencapai suatu basis sebagai penanda kapan rekursi itu akan berhenti [10][12]. Dalam pembangkitan dunia acak, Fractal Brownian Motion [13] dan Diamond-Square Algorithm digunakan untuk menghasilkan terrain alami melalui pendekatan rekursif [14] [15].

### F. *Kombinatorika*

Kombinatorika merupakan perhitungan terhadap banyaknya kemungkinan suatu kejadian. Kombinatorika dapat digunakan untuk mengetahui banyaknya cara penyusunan suatu objek yang meliputi pemasangan, pengelompokan, pengurutan, pemilihan, atau penempatan objek dengan karakteristik tertentu [16]. Pada gim Terraria dan Minecraft kombinatorika berperan dalam menghitung jumlah kemungkinan konfigurasi blok, item, atau struktur dalam dunianya. Prinsip kombinasi juga digunakan untuk mengatur probabilitas kemunculan suatu objek dari total kemungkinan yang ada.

### G. *Graf dan pohon*

Graf merupakan struktur yang merepresentasikan objek-objek diskrit yang disebut simpul dan hubungan antara objek-objek tersebut atau sisi. Sedangkan pohon (tree) merupakan suatu graf yang tidak memiliki siklus atau dengan kata lain, pohon tidak dapat memiliki suatu lintasan yang dapat menghubungkan suatu simpul dengan simpul itu sendiri [17]. Dalam konteks pembangkitan dunia, graf digunakan untuk merepresentasikan keterhubungan antar ruang sehingga dunia tetap memiliki hubungan walaupun dibentuk secara acak misalnya digunakan pada penghubung ruang dengan ruang yang lainnya untuk implementasi keterhubungan bagian struktur-struktur pada Terraria dan Minecraft. Pohon digunakan pada game Minecraft dalam pengelompokan chunk yang tersusun kedalam bentuk quadtree [18].

## III. SIMULASI DAN IMPLEMENTASI

### A. *Metode Pembangkit Dunia Acak Berbasis Matematika Diskrit*

Dengan memanfaatkan konsep Teori Bilangan, Pembangkitan dunia acak pada gim Terraria dan Minecraft dapat dibentuk bersifat deterministik, yang artinya dunia dibuat tidak benar-benar acak. Pembangkitan dunia Terraria dan Minecraft menggunakan bilangan input atau yang disebut dengan seed, dengan hal ini, dunia yang dibentuk dengan seed yang sama akan membentuk dunia yang sama. Pembuatan dunia ini dapat dilakukan dengan menggunakan konsep Matematika Diskrit pembangkit bilangan acak yang berbasis kekongruenan lanjar atau linear congruential generator (LCG) [1].

$$Xn = (aXn - 1 + b) \bmod m$$

Xn = bilangan acak ke-n dari deretnya

Xn – 1 = bilangan acak sebelumnya

a = faktor pengali

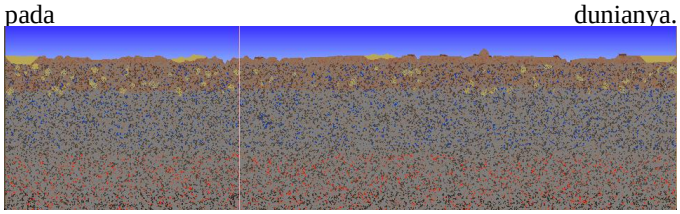
b = increment

m = modulus

$X_0$ adalah umpan dari pembangkit bilangan acak atau yang disebut dengan seed. Pada bahasa pemrograman C, fungsi srand(seed) dimanfaatkan untuk menghasilkan angka acak semu yang konsisten sesuai input seed dari pengguna. Oleh karena itu dengan menggunakan seed yang sama, maka setiap kali melakukan pembangkitan dunia, dunia yang terbentuk sama.

Konsep Matematika Diskrit yang digunakan lainnya adalah Logika, Aljabar Boolean, dan Kombinatorika. Automata seluler dapat digunakan dalam pembangkitan dunia acak

bedasarkan suatu grid di mana setiap sel memiliki status tertentu yang bergantung pada statusnya sendiri dan status tetangganya. Blok pada gim diproses sebagai status sel dengan aturan-aturan tertentu seperti aturan ketinggian pada peta suatu dunianya. Dalam pembangkitan dunia Terraria dan Minecraft, konsep ini digunakan pada pembentukan lubang di tanah sehingga dapat membentuk gua di bawah tanah. Konsep operasi AND, OR, NOT, dan XOR digunakan secara implisit pada pembentukan dunianya. Konsep kombinatorika digunakan untuk penentuan kemungkinan kemunculan struktur dan mineral-mineral dengan menggunakan aturan ketinggian pada dunianya.



Gambar 2. Contoh dunia Terraria. Sumber: arsip pengguna

Penggunaan konsep Rekursi dan Relasi Rekurens pada pembangkitan dunia acak dapat dilihat pada pembentukan dunia Minecraft dengan pendekatan Diamond-Square Algorithm dan Fractal Brownian Motion (fBm) untuk menentukan ketinggian permukaan dari dunia yang dibentuk dengan pendekatan pola alami dan tidak berulang.

Konsep Graf dan Pohon pada penerapan pembangkitan dunia acak menjadi aturan dasar pembentukan dan memastikan bagian-bagian dunia memiliki konektivitas dan struktur yang tidak kacau.

*B. Skema Simulasi Pembangkitan Dunia Terraria*

Pada simulasi dunia Terraria, dunia dua dimensi dibagi menjadi lima lapisan vertikal. Langit 0-6% dari ketinggian maksimal, permukaan 6-27% dari ketinggian maksimal, gua 27-48% dari ketinggian maksimal, gua besar 48-84% dari ketinggian maksimal, dunia bawah 84-100% dari ketinggian maksimal. Simulasi pembangkitan dunia menggunakan program berbasis bahasa C.

Program utama pembangkitan dunia acak

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#define WIDTH 140
#define HEIGHT 40

#define SPACE ' '
#define DIRT 'D'
#define GRASS 'G'
#define STONE 'S'
#define ORE 'O'
#define WATER 'W'
#define LAVA 'L'
#define CLOUD 'C'
#define FLOAT 'F'
#define SNOW 'N'
#define ICE 'I'
#define SAND 'A'
#define SANDSTONE 'T'
#define MUD 'M'
#define ASH 'H'

char map[HEIGHT][WIDTH];
int spaceY, surfaceY, undergroundY, cavernY,
underworldY;


int main() {
    unsigned int seed;
    printf("Masukkan seed: ");
    scanf("%u", &seed);
    srand(seed);
    initMap();
    defineLayers();
    generateSky();
    for (int i = 0; i < 10; i++)
simulateCave();
    fillLiquids();
    addBiomes();
    underworldStructures();
    placeOre(0.1);
    convertDirtToGrass();
    createBeaches();
    removeLonelyWaterRows();
    printMap();
    return 0;
}
```

```c
void initMap() {
    for (int y = 0; y < HEIGHT; y++)
        for (int x = 0; x < WIDTH; x++)
            map[y][x] = SPACE;
}
```

Inisiasi array map dengan mengisi semua sebagai ruang kosong.

```c
void defineLayers() {
    spaceY = 0;
    surfaceY = HEIGHT * 6 / 100;
    undergroundY = HEIGHT * 27 / 100;
    cavernY = HEIGHT * 48 / 100;
    underworldY = HEIGHT * 84 / 100;
    int surfaceHeight[WIDTH];
    int baseHeight = surfaceY + 2;
    int variation = 1;
    for (int x = 0; x < WIDTH; x++) {
        surfaceHeight[x] = baseHeight +
rand() % (variation * 2 + 1);
        if (x < 6 || x > WIDTH - 7) {
```

```
            surfaceHeight[x] = baseHeight +
1;
        }
        for (int y = surfaceHeight[x]; y <
undergroundY; y++) {
            map[y][x] = (rand() % 100 < 85) ?
DIRT : SPACE;
        }
        for (int y = spaceY; y <
surfaceHeight[x]; y++) {
            map[y][x] = SPACE;
        }
    }
    for (int y = undergroundY; y < cavernY;
y++)
        for (int x = 0; x < WIDTH; x++)
            if (map[y][x] == SPACE)
                map[y][x] = (rand() % 100 <
72) ? STONE : SPACE;
    for (int y = cavernY; y < HEIGHT; y++)
        for (int x = 0; x < WIDTH; x++)
            if (map[y][x] == SPACE)
                map[y][x] = (rand() % 100 <
70) ? STONE : SPACE;
}
```

Membagi dunia kedalam lima layer yang berbeda.

```
void generateSky() {
    for (int y = spaceY; y < surfaceY/2; y++)
{
        for (int x = 2; x < WIDTH - 2; x++) {
            if (rand() % 100 < 5) map[y][x] =
CLOUD;
            if (rand() % 100 < 2 && y <
surfaceY - 1) map[y + 1][x] = FLOAT;
        }
    }
}
```

Membuat awan pada bagian langit.

```
int countWalls(int x, int y) {
    int count = 0;
    for (int dy = -1; dy <= 1; dy++)
        for (int dx = -1; dx <= 1; dx++) {
            int nx = x + dx, ny = y + dy;
            if (nx < 0 || ny < surfaceY || nx
>= WIDTH || ny >= HEIGHT)
                count++;
            else if (map[ny][nx] == DIRT ||
map[ny][nx] == STONE)
                count++;
        }
    return count;
}
```

```
void simulateCave() {
    char temp[HEIGHT][WIDTH];
    int dirtTop = surfaceY + (underworldY -
surfaceY) / 3;
    for (int y = surfaceY; y < HEIGHT; y++) {
        for (int x = 0; x < WIDTH; x++) {
            int walls = countWalls(x, y);
            if (walls >= 6) {
                if (y < dirtTop)
                    temp[y][x] = (map[y][x]
== STONE) ? STONE : DIRT;
                else
                    temp[y][x] = (map[y][x]
== DIRT) ? DIRT : STONE;
            } else {
                temp[y][x] = SPACE;
            }
        }
    }
    for (int y = surfaceY; y < HEIGHT; y++)
        for (int x = 0; x < WIDTH; x++)
            map[y][x] = temp[y][x];
}
```

Pembangkitan dunia dimulai dengan memasukan seed untuk determinasi pembentukan dunia. Kemudian secara acak memasang tiles setiap grid dengan tanah dan batu yang merupakan status grid hidup atau ruang yang merupakan status grid mati. Dengan menggunakan algoritma seluler automata, pembentukan lubang-lubang sebagai gua dapat dibentuk.

```
void placeOre(double oreProb) {
    for (int y = HEIGHT / 2; y < HEIGHT; y++)
{
        for (int x = 0; x < WIDTH; x++) {
            if (map[y][x] == STONE &&
((double)rand() / RAND_MAX < oreProb)) {
                map[y][x] = ORE;
            }
        }
    }
}
```

Memasang beberapa mineral yang tersebar di bagian bawah dunia.

```
void fillLiquids() {
    for (int y = (underworldY + (HEIGHT -
underworldY) / 3); y < HEIGHT; y++) {
        for (int x = 0; x < WIDTH; x++) {
            if (map[y][x] == SPACE) {
                map[y][x] = LAVA;
            }
        }
    }
}
```

Membuat ruang kosong di bagian paling bawah dunia menjadi lahar.

```
void addBiomes() {
    int start = (WIDTH / 20) + rand() %
(WIDTH / 2);
    for (int x = start; x < start + WIDTH /
10; x++)
        for (int y = surfaceY; y < HEIGHT /
2; y++) {
            if (map[y][x] == DIRT) map[y][x]
= SNOW;
            if (map[y][x] == STONE) map[y][x]
= ICE;
        }
    start = (WIDTH / 20) + rand() % (WIDTH /
2);
    for (int x = start; x < start + WIDTH /
10; x++)
        for (int y = surfaceY; y < HEIGHT /
2; y++) {
            if (map[y][x] == DIRT) map[y][x]
= SAND;
            if (map[y][x] == STONE) map[y][x]
= SANDSTONE;
        }
    start = (WIDTH / 20) + (WIDTH / 2) +
rand() % (WIDTH / 4);
    for (int x = start; x < start + WIDTH /
10 && x < WIDTH; x++)
        for (int y = surfaceY; y < HEIGHT;
y++) {
            if (map[y][x] == DIRT ||
map[y][x] == STONE)
                map[y][x] = MUD;
        }
}
```

Kemudian pembangkitan dunia dilanjutkan dengan membuat beberapa bagian dunia kedalam bioma yang berbeda.

```
void underworldStructures() {
    for (int y = underworldY; y < HEIGHT;
y++) {
        for (int x = 0; x < WIDTH; x++) {
            if (y < underworldY + (HEIGHT -
underworldY) / 10) {
                if (map[y][x] == STONE)
map[y][x] = ASH;
            } else if (y < underworldY +
(HEIGHT - underworldY) / 3) {
                map[y][x] = SPACE;
            } else {
                if (map[y][x] == STONE)
map[y][x] = ASH;
            }
        }
    }
}
```

Mengubah batu menjadi debu untuk bagian bawah dunia.

```
void convertDirtToGrass() {
    for (int y = 0; y < undergroundY; y++) {
        for (int x = 0; x < WIDTH; x++) {
            if (map[y][x] == DIRT) {
                if ((y > 0 && map[y - 1][x]
== SPACE) || (y < HEIGHT - 1 && map[y + 1][x]
== SPACE) ||
                    (x > 0 && map[y][x - 1]
== SPACE) || (x < WIDTH - 1 && map[y][x + 1]
== SPACE)) {
                    map[y][x] = GRASS;
                }
            }
        }
    }
}
```

Dengan menggunakan pengkondisian bahwa tanah dikelilingi setidaknya satu ruang kosong, maka tanah akan berubah menjadi rumput

```
void createBeaches() {
    int beachWidth = WIDTH/20;
    int waterDepth = HEIGHT * 6/100;
    for (int x = 0; x < WIDTH; x++) {
        if (x < beachWidth || x > WIDTH -
beachWidth - 1) {
            int topGroundY = -1;
            for (int y = 0; y < HEIGHT; y++)
{
                if (map[y][x] == DIRT ||
map[y][x] == GRASS || map[y][x] == SNOW ||
map[y][x] == SAND) {
                    topGroundY = y;
                    break;
                }
            }
            if (topGroundY != -1) {
                for (int d = 0; d <
waterDepth && (topGroundY + d) < HEIGHT; d++)
{
                    int wy = topGroundY + d;
                    if (map[wy][x] == SPACE
|| map[wy][x] == DIRT || map[wy][x] == GRASS
|| map[wy][x] == SNOW || map[wy][x] == SAND)
{
                        map[wy][x] = WATER;
                    }
                }
            }
```

```
        }
    }
    for (int y = 1; y < HEIGHT - 1; y++) {
        for (int x = 1; x < WIDTH - 1; x++) {
            if (map[y][x] == WATER) {
                if (map[y + 1][x] != WATER &&
map[y + 1][x] != SPACE) map[y + 1][x] = SAND;
                if (map[y][x - 1] != WATER &&
map[y][x - 1] != SPACE) map[y][x - 1] = SAND;
                if (map[y][x + 1] != WATER &&
map[y][x + 1] != SPACE) map[y][x + 1] = SAND;
            }
        }
    }
}
```

Pada bagian ujung dunia, dibentuk pantai sebagai pertanda ujung dari dunia.

```
void removeLonelyWaterRows() {
    for (int y = 0; y < HEIGHT; y++) {
        int hasWater = 0;
        int hasSolidLeft = 0, hasSolidRight =
0;

        for (int x = 0; x < WIDTH; x++) {
            if (map[y][x] == WATER) {
                hasWater = 1;
                for (int lx = x - 1; lx >= 0;
lx--) {
                    if (map[y][lx] != SPACE
&& map[y][lx] != WATER) {
                        hasSolidLeft = 1;
                        break;
                    }
                }
                for (int rx = x + 1; rx <
WIDTH; rx++) {
                    if (map[y][rx] != SPACE
&& map[y][rx] != WATER) {
                        hasSolidRight = 1;
                        break;
                    }
                }
                break;
            }
        }
        if (hasWater && !hasSolidLeft &&
!hasSolidRight) {
            for (int x = 0; x < WIDTH; x++) {
                if (map[y][x] == WATER) {
                    map[y][x] = SPACE;
                }
            }
        }
    }
}
```

Pemeriksaan dibuat agar air selalu diapit oleh grip yang bukan ruang kosong.

```
void printMap() {
    for (int y = 0; y < HEIGHT; y++) {
        for (int x = 0; x < WIDTH; x++)
            putchar(map[y][x]);
        putchar('\n');
    }
}
```

Terakhir, setelah map berhasil dibuat, hasil akan ditampilkan pada layar.

Pengaplikasian kode pembangkitan dunia acak menggunakan konsep Teori Bilangan saat pembentukan acak deterministik dengan memasukan umpan atau seed. Pengaplikasian juga memerlukan konsep Logika dan Aljabar Boolean, dan Kombinatorika agar hasil dari pembentukan melalui seluler automata dapat bersifat acak. Konsep Graf dan Tree digunakan secara implisit melalui aturan dan hubungan antar elemen dalam dunia.

Contoh hasil dari kode:

Masukkan seed: 123456789

Masukkan seed: 987654321

### *C. Skema Simulasi Pembangkitan Dunia Minecraft*

Minecraft membagi dunianya menjadi satuan chunk berukuran 16×16 blok horizontal dan 384 blok vertikal. Simulasi pembangkitan dunia Minecraft dibentuk bedasarkan dilakukan secara modular per chunk berdasarkan koordinat dan

nilai seed. Dengan menggunakan konsep Graf dan Tree di mana setiap chunk dapat direpresentasikan sebagai simpul dan keterhubungan antar chunk sebagai sisi. Diamond-Square Algorithm dan Fractal Brownian Motion (fBm), sehingga untuk ketinggian setiap bagian tidak terlalu jauh dibandingan dengan tetangganya.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

#define SIZE 17
#define MAX_HEIGHT 384.0
#define OCTAVES 5

double heightmap[SIZE][SIZE];

double noise(int x, int y) {
    int n = x + y * 57;
    n = (n << 13) ^ n;
    return (1.0 - ((n * (n * n * 15731 +
789221) + 1376312589) & 0x7fffffff) /
1073741824.0);
}

double smoothNoise(double x, double y) {
    int intX = (int)x;
    int intY = (int)y;
    double fracX = x - intX;
    double fracY = y - intY;
    double v1 = noise(intX, intY);
    double v2 = noise(intX + 1, intY);
    double v3 = noise(intX, intY + 1);
    double v4 = noise(intX + 1, intY + 1);
    double i1 = v1 * (1 - fracX) + v2 *
fracX;
    double i2 = v3 * (1 - fracX) + v4 *
fracX;
    return i1 * (1 - fracY) + i2 * fracY;
}

double fBm(double x, double y) {
    double total = 0, amplitude = 1,
frequency = 1;
    for (int i = 0; i < OCTAVES; i++) {
        total += smoothNoise(x * frequency, y
* frequency) * amplitude;
        amplitude *= 0.5;
        frequency *= 2.0;
    }
    return total;
}

void diamondSquare() {
    int step = SIZE - 1;
    double scale = MAX_HEIGHT / 2.0;
    while (step > 1) {
        int half = step / 2;
        for (int y = half; y < SIZE - 1; y +=
step) {
            for (int x = half; x < SIZE - 1;
x += step) {
                double avg = (
                    heightmap[y - half][x -
half] +
                    heightmap[y - half][x +
half] +
                    heightmap[y + half][x -
half] +
                    heightmap[y + half][x +
half]
                ) / 4.0;
                heightmap[y][x] = avg +
((rand() % 100) / 100.0 - 0.5) * scale;
            }
        }
        for (int y = 0; y < SIZE; y += half)
{
            for (int x = (y + half) % step; x
< SIZE; x += step) {
                double sum = 0; int count =
0;
                if (x - half >= 0) { sum +=
heightmap[y][x - half]; count++; }
                if (x + half < SIZE) { sum +=
heightmap[y][x + half]; count++; }
                if (y - half >= 0) { sum +=
heightmap[y - half][x]; count++; }
                if (y + half < SIZE) { sum +=
heightmap[y + half][x]; count++; }

                heightmap[y][x] = (sum /
count) + ((rand() % 100) / 100.0 - 0.5) *
scale;
            }
        }
        step /= 2;
        scale /= 2.0;
    }
}

void generateMinecraftWorld() {
    for (int y = 0; y < SIZE; y++)
        for (int x = 0; x < SIZE; x++)
            heightmap[y][x] = (fBm(x * 0.1, y
* 0.1) + 1.0) * MAX_HEIGHT / 2;

    diamondSquare();
}

void printHeightmap() {
    for (int y = 0; y < SIZE; y++) {
```

```
        for (int x = 0; x < SIZE; x++) {
            printf("%3.0f ", heightmap[y][x]
- 64.0);
        }
        printf("\n");
    }
}

int main() {
    unsigned int seed;
    printf("Masukkan seed dunia (angka): ");
    scanf("%u", &seed);
    srand(seed);
    generateMinecraftWorld();
    printHeightmap();
    return 0;
}
```

Diamond-Square Algorithm digunakan untuk menghasilkan peta ketinggian sedangkan Fractal Brownian Motion digunakan untuk menghasilkan gelombang atau kontur dunia yang natural. Diamond-Square Algorithm menggunakan grid berukuran $n^2+1$ dalam implementasi algoritmanya. Implementasi konsep Rekursi dan Relasi Rekurens pada pembangkitan dunia acak Minecraft menghasilkan dunia yang acak tetapi tetap ada hubungannya antar chunk dengan chunk lainnya.

Contoh hasil dari kode:

Masukkan seed dunia (angka): 1234566789

```
Masukkan seed dunia (angka): 1234566789
 23  50  74  84 106 109  95 122 136 100  87  81  79  98 100 113 131
 33  45  70  68  91  91  94 100 103  98  97  87  90  87 109 109 101
 64  65  49  72  77  86  71  74 101  85 101  86  74  83  94  96  75
 88  69  77  75  66  66  70  68  78  78  90  79  74  55  53  68  63
 96  93  87  60  49  63  70  54  71  62  71  61  45  48  24  36  32
 82  82  77  54  63  62  58  50  45  61  51  53  30  23  13  21  29
 90  68  71  52  48  45  52  58  42  51  56  33  29  12  21  20  32
 91  73  54  50  47  60  61  59  59  45  34  38  17  10  12  29  27
 81  81  67  60  62  60  51  53  87  57  39  26  -4  18  17  29   3
 71  85  71  75  74  77  75  65  67  52  39  21  18  16  28  25  25
 66  72  83  82  94  73  71  70  70  47  43  38  46  38  23  33  38
 81  81  88  86 102  89  98  92  76  66  61  55  48  30  43  32  28
 98  84  80  96  93  88 102  87  94  85  60  45  33  28  53  45  24
101  89 101  98 105 110 108  99  97  95  88  72  49  41  52  38  38
104  97  93  97 114  97 100 112 110  97  89  84  81  62  43  43  27
115 103  92 103 101  99  86  99  83  80  81  77  71  63  51  59  49
128 117 118 103  77  78  71  94  83  78  75  89  82  80  75  76  64
```

Masukkan seed dunia (angka): 1234566789

```
Masukkan seed dunia (angka): 987654321
 23   7  13  23  33  25  -5   6   6  50  71  64  78  74  86  97 131
 12   5   8  14  27  16  10   5  16  51  66  73  71  68  68  87 104
 19   7  -9   3  14  12   1   5  29  42  63  54  62  47  59  83  90
 -2 -16 -21  -4   1   9  25  32  35  58  54  40  36  44  70  79  67
-16 -21 -19 -22 -13   8  37  50  47  44  48  34  28  49  67  57  56
 -4 -13  -8  -7   4  18  28  41  52  43  38  22  21  49  54  49  66
  3  16  15  -8 -11   9  39  45  59  37  22  27  28  27  37  52  67
  3  12  10   5  17  26  42  37  60  44  37  18  37  38  36  65  68
  1   5  -3  -4   9   7  22  34  54  41  47  38  21  30  33  60 106
  2   9   4  20  18  17  36  34  40  48  35  35  42  33  46  66  83
 25  21  23  32  49  33  31  39  58  38  27  30  34  40  57  67  52
 44  61  39  41  56  42  50  55  52  42  38  27  34  64  63  57  52
 80  72  72  54  46  51  67  74  60  45  31  37  47  55  78  63  61
 82  62  65  69  60  76  64  75  77  65  59  47  47  61  71  65  74
 74  74  64  65  69  68  75  76  71  87  84  71  65  57  79  74  72
 89  85  90  83  73  75  83  95 105  91  80  79  75  68  68  77  60
128 101  91  75  72  93 118 110 143 114  89  68  54  76  76  75  64
```

### *D. Evaluasi Konsep Matematika Diskrit*

Penerapan konsep Matematika Diskrit pada gim Terraria terlihat pada pengaplikasian Automata Seluler, Aljabar Boolean, dan Logika untuk pembentukan gua yang terbentuk dari status grid pada dunianya. Pembangkitan dunia acak secara deterministik menggunakan Konsep Teori Bilangan linear congruential generator sehingga untuk umpan yang sama, pembentukan dunia selalu sama, hal ini juga diimplementasi pada gim Minecraft. Konsep Rekursi dan Relasi Rekurens pada gim Terraria digunakan secara implisit pada penggunaan Automata Seluler, sedangkan pada gim Minecraft, konsep digunakan pada Diamond-Square Algorithm dan Fractal Brownian Motion dalam pembentukan heightmap dunia. Konsep Graf dan Tree digunakan secara implisit sebagai aturan dasar pembentukan dunia, seperti aturan sambungan gua dan hubungan bioma. Konsep Kombinatorika diimplementasikan untuk menghasilkan pemilihan material, gua, bioma, dan peletakan mineral serta penempatan blok dan struktur pada dunia.

## IV. KESIMPULAN DAN SARAN

Pembangkitan dunia acak pada gim Terraria dan Minecraft menunjukkan penerapan nyata dari berbagai konsep Matematika Diskrit. Konsep Teori Bilangan linear congruential generator dengan menggunakan seed atau umpan sebagai input deterministik untuk menghasilkan angka pseudo-acak, membuat dunia yang dihasilkan bersifat konsisten namun tetap menghadirkan keunikan dalam setiap permainan. Konsep Automata Seluler berperan dalam pembentukan struktur gua yang kompleks secara lokal namun menyatu secara global dengan menggunakan konsep Aljabar Boolean dan Logika yang mendasari kondisi dan transisi status tiap blok dalam dunia. Konsep Kombinatorika mengatur probabilitas distribusi objek dan elemen. Penggunaan konsep Rekursi seperti Diamond-Square dan Fractal Brownian Motion menunjukkan bahwa pendekatan berbasis Relasi Rekurens dan Rekursi dapat menghasilkan dunia yang realistis. Keterhubungan antar bagian dunia juga memanfaatkan konsep Graf dan Tree, sehingga dunia yang dihasilkan tidak hanya acak, tetapi juga memiliki keterhubungan dan dapat dijelajahi secara logis oleh pemain.

Adapun saran pengembangan agar sistem pembangkitan dunia memiliki visualisasi interaktif dengan implementasi sistem grafik 2D atau 3D menggunakan library seperti SDL, SFML, atau OpenGL sehingga dapat meningkatkan pengalaman pengguna dan pemodelan yang lebih nyata. Untuk gim Minecraft, simulasi dapat diperluas ke dimensi 3D penuh dengan pembentukan struktur yang lebih kompleks seperti gua dan bioma. Selain itu, metode pembangkitan dunia dapat diperluas dengan pendekatan matematis lain seperti Voronoi diagram, Simplex noise, atau bahkan pemodelan berbasis teori peluang spasial, guna menghasilkan struktur dunia yang lebih variatif dan alami.

## LAMPIRAN

Berikut lampiran github yang berisi program lengkap mengenai simulasi pembangkitan dunia acak Terraria dan heightmap Minecraft https://github.com/IzzyMeow/Matdis.


## UCAPAN TERIMAKASIH

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa atas segala rahmat, karunia, dan petunjuk-Nya sehingga penulis dapat menyelesaikan makalah ini yang berjudul

“Penerapan Matematika Diskrit dalam Pembangkitan Dunia Acak pada Gim Terraria dan Minecraft” tepat waktu tanpa kendala yang berarti.

Penulis menyampaikan terima kasih yang sebesar-besarnya kepada Bapak Arrival Dwi Sentosa, S.Kom., M.T. selaku dosen pengampu mata kuliah IF2120 Matematika Diskrit kelas K02 atas bimbingan, arahan, dan ilmu yang telah diberikan selama perkuliahan berlangsung. Ucapan terima kasih juga penulis sampaikan kepada Bapak Dr. Ir. Rinaldi Munir, M.T. yang telah memberikan sumber-sumber referensi pembelajaran Matematika Diskrit melalui situs dan tulisan beliau yang sangat membantu dalam proses penyusunan makalah ini.

Penulis juga mengucapkan terima kasih kepada keluarga yang telah memberikan semangat, masukan, dan dukungan selama proses pengerjaan makalah ini berlangsung. Penulis berharap makalah ini dapat memberikan manfaat bagi pembaca, khususnya dalam memahami aplikasi nyata Matematika Diskrit dalam dunia pemrograman dan pengembangan gim.

## REFERENSI

[1] Munir, Rinaldi. 2024. “Teori Bilangan (Bagian 3)”. https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/17-Teori-Bilangan-Bagian3-2024.pdf. Diakses pada 8 Juni 2025 2:38.

[2] AXE Indonesia. 2022. “10 Genre Game Terpopuler, Mana yang Paling Seru?”. https://www.axe.com/id/inspirasi/culture/genre-game.html. Diakses pada 8 Juni 2025 3:48

[3] Landin, Peter. 2023. What is Minecraft. https://www.minecraft.net/en-us/article/what-minecraft. Diakses pada 8 Juni 2025 3:48

[4] Re-Logic. 2021. Terraria Official Website. https://terraria.org/. Diakses pada 8 Juni 2025 3:48

[5] Minecraft Community. 2025. “Minecraft Wiki – World generation,”. https://minecraft.wiki. Diakses pada 8 Juni 2025 4:51

[6] Terraria Community. 2025. “Terraria Wiki – World generation,”. https://terraria.wiki. Diakses pada 8 Juni 2025 4:51

[7] Institut Tekonologi Sepuluh Nopember. 2017. “*Pengantar Cellular Automata”*. https://www.landusesim.com/wp-content/uploads/2017/08/1.-Pengantar-Cellular-Automata.pdf. Diakses pada 8 Juni 2025 4:51

[8] Kun, Jeremy. 2012. “*The Cellular Automaton Method for Cave Generation”*. https://www.jeremykun.com/2012/07/29/the-cellular-automaton-method-for-cave-generation/. Diakses pada 8 Juni 2025 4:51

[9] Stack Exchange. 2017. “*Terraria Map Explanation”*. https://gaming.stackexchange.com/questions/297787/terraria-map-explanation. Diakses pada 8 Juni 2025 4:51

[10] Munir, Rinaldi. 2004. “*Teori Bilangan – Kriptografi”*. https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/Teori%20Bilangan.pdf. Diakses pada 8 Juni 2025 22:25

[11] Minecraft Fandom. . “*Seed (Level Generation)”*. https://minecraft.fandom.com/wiki/Seed_(level_generation). Diakses pada 8 Juni 2025 22:25

[12] Geeks for Geeks. 2024. “*Recurrence Relations – A Complete Guide”*. https://www.geeksforgeeks.org/recurrence-relations-a-complete-guide/. Diakses pada 9 Juni 2025 00:23

[13] Vivo, Patricio Gonzalez. Lowe, Jen. 2015. “*Fractal Brownian Motion – The Book of Shaders”*. https://thebookofshaders.com/13. Diakses pada 9 Juni 2025 01:03

[14] White Box Dev. 2021. “Diamond Square | Procedural Generation | Game Development Tutorial”. https://www.youtube.com/watch?v=4GuAV1PnurU. Diakses pada 9 Juni 2025 01:03

[15] O'Brien, Nick. 2018. “*Diamond-Square Algorithm: Explanation and C++ Implementation”*. https://medium.com/@nickobrien/diamond-square-algorithm-explanation-and-c-implementation-5efa891e486f. Diakses pada 9 Juni 2025 01:03

[16] Kumparan. 2022. “*Apa Itu Kombinatorika? Ini Penjelasan dan Contoh Soalnya”*. https://kumparan.com/berita-hari-ini/apa-itu-kombinatorika-ini-penjelasan-dan-contoh-soalnya-1yrVJeJCv0n/2. Diakses pada 9 Juni 2025 02:09

[17] Munir, Rinaldi. 2024. “*Graf – Matematika (Diskrit Bagian 1)”*. https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf. Diakses pada 9 Juni 2025 02:09

[18] SimonDev. 2020. “3D World Generation: #3 (Quadtree & LOD)”. https://www.youtube.com/watch?v=YO_A5w_fxRQ. Diakses pada 9 Juni 2025 02:09

## 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, 1 Juni 2025

Bernhard Aprillio Pramana - 13524074