Penerapan Algoritma DFS (*Depth-First Search*) dalam Pencarian Struktur Dunia Minecraft Menggunakan Custom Mod *Minecraft Forge*

Brigita Tri Carolina - 13521156
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13521156@std.stei.itb.ac.id

***Abstract*—*Minecraft* merupakan sebuah permainan video yang dikembangkan oleh Mojang Studios. Di dalam permainan ini, pemain dapat menjelelajahi dunia secara acak yang dibangun melalui proses yang disebut dengan "*World Generation*" atau pembangkitan dunia. Dunia *Minecraft* terdiri atas berbagai macam struktur seperti gua, hutan, gunung, dan desa. Pencarian dan eksplorasi struktur dunia merupakan salah satu bagian yang penting dalam permainan. Algoritma DFS adalah salah satu pendekatan pencarian yang dapat digunakan untuk menjelajahi struktur dunia Minecraft secara efisien.**

Keywords*—*Depth-First-Search, Minecraft, Minecraft world, Biomes in Minecraft, DFS, Eksplorasi Dunia

I. PENDAHULUAN

Gambar 1.Gua pada permainan Minecraft
Sumber:
https://www.lacremedugaming.fr/actus/minecaft-presente-la-mise-a-jour-de-caves-cliffs-part-ii-en-video-42779.html

Permainan *Minecraft* merupakan permainan yang dikembangkan oleh Mojang Studios yang telah menjadi fenomena global yang luas dengan jutaan pemain yang terlibat. Permainan ini mengangkat tema membangun, menjelajah, dan mengubah lingkungan yang dibuat secara acak. Dengan dunia yang luas dan begitu kompleks, mencari struktur dunia *Minecraft* seperti gua, hutan, gunung, ataupun desa dengan tujuan tertentu dalam permainan bisa menjadi tugas yang menantang. Dalam upaya memberikan pengalaman bermain yang lebih menarik, dibutuhkan pengembangan atau modifikasi pada *game* sehingga pemain dapat menemukan daerah-daerah tententu dengan lebih cepat dan efisien. Algoritma DFS merupakan salah satu algoritma pencarian dengan menyusuri kedalaman yang dapat diimplementasikan dalam pencarian daerah ini. Algoritma ini memungkinkan pemain untuk secara sistematis menjelajahi dan mencari struktur dunia *Minecraft*.

Dalam makalah ini, penulis akan menjelajadi penerapan algoritma DFS dalam pencarian struktur dunia dalam permainan *Minecraft.* Penulis akan membahas konsep dasar algoritma DFS, prinsip kerjanya, serta bagaimana algoritma ini dapat diterapkan pada pencarian daerah dengan menggunakan custom mod pada permainan Minecraft. Melalui pemahaman yang mendalam tentang penerapan algoritma DFS pada pencarian struktur dunia *Minecraft*, diharapkan pemain akan mendapatkan strategi yang lebih efisien dan efektif dalam menavigasi dan menjelajahi dunia *Minecraft* yang luas ini. Dengan bantuan algoritma DFS, pemain dapat menemukan struktur dan tujuan daerah yang diinginkan dengan lebih cepat dan efisien yang pada akhirnya dapat meningkatkan kesenangan dan tantangan dalam permainan ini.

II. DASAR TEORI

A. Traversal Graf

Algoritma traversal graf adalah algoritma yang mengujungi simpul dengan cara yang sistematik. Algoritma traversal graf di antaranya adalah:

1. Pencarian melebar (*breadth first search)*
2. Pencarian mendalam (*depth first search)*

Dengan asumsi bahwa graf terhubung. Dalam hal ini, graf digunakan sebagai representasi dari sebuah persoalan. Kemudian traversal graf digunakan sebagai alat pencarian solusi.

B. Algoritma Pencarian Solusi

Algoritma pencarian solusi dapat dibedakan menjadi dua, yaitu:

1. Tanpa informasi (*uninformed/blind search*)
    - Tidak ada informasi tambahan
    - Contoh: BFS, DFS, *Depth Limited Search*, *Iterative Deepening Search*, dan *Uniform Cost Search*.
2. Dengan informasi (*informed search*)
    - Pencarian berbasis heuristik dan mengetahui *non-goal state* lebih menjanjikan daripada yang lain
    - Contoh: *Best First Search*, A*

C. Representasi Graf dalam Proses Pencarian

Dalam proses pencarian solusi, terdapat dua macam graf yang dapat digunakan, di antaranya:

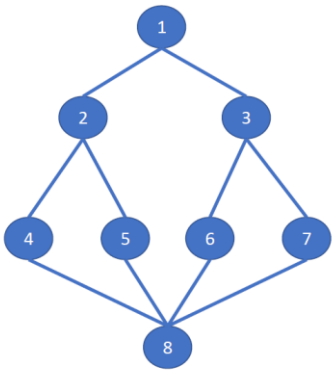
1. Graf Statis

   Graf statis merupakan graf yang sudah terbentuk sebelum proses pencarian dilakukan. Misalnya graf yang direpresentasikan sebagai struktur data.

2. Graf Dinamis

   Graf dinamis merupakan graf yang terbentuk saat proses pencarian dilakukan. Graf tidak tersedia pada awal pencarian dan dibangun pada saat proses pencarian.

D. DFS dengan Graf Statis



Gambar 2. Contoh graf statis
Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag1.pdf

Berikut adalah proses pencarian pada graf statis dengan menggunakan algoritma DFS:

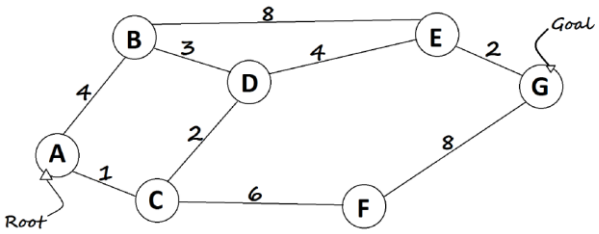
1. Misal simpul *start* adalah simpul a. Kunjungi simpul a.
2. Kunjungi simpul b yang bertetangga dengan simpul a.
3. Ulang DFS dari simpul b.
4. Ketika sampai pada suatu simpul, misalnya simpul c dan semua simpul yang bertetangga telah dikunjungi, lakukan runut-balik (*backtracking*) ke simpul terakhir yang dikunjungi sebelumnya yang mempunyai simpul b yang belum dikunjungi.
5. Pencarian dihentikan ketika tidak ada lagi simpul yang belum dikunjungi yang dapat dicapai oleh simpul yang telah dikunjungi.

Berikut adalah ilustrasi algoritma DFS pada graf statis di atas dengan DFS() berisi simpul yang sedang diekspan atau a dan *array* dikunjungi diperuntukkan untuk menyimpan *boolean* apakah suatu simpul telah dikunjungi atau belum, dan terakhir DFS simpul selanjutnya.

1. DFS(1); a = 1; dikunjungi[1] = *true* → DFS (2)
2. DFS(2) ; a = 2; dikunjungi[2] = *true* → DFS(4)
3. DFS(4); a = 4; dikunjungi[4] = *true* → DFS(8)
4. DFS(8); a = 8; dikunjungi[8] = *true* → DFS(5) (proses *backtracking*)
5. DFS(5); a = 5; dikunjungi[5] = *true* → tidak ada lagi tetangga yang belum dikunjungi yang berhubungan dengan simpul 5. Kunjungi tetangga simpul 8 yang lain → DFS(6)
6. DFS(6); a = 6; dikunjungi[6] = *true* → DFS(3)
7. DFS(3); a = 3; dikunjungi[3] = *true* → DFS(7)
8. DFS(7)' a = 7; dikunjungi[7] = *true* → tidak ada lagi tetangga yang belum dikunjungi maka proses pencarian berakhir.

Urutan simpul yang dikunjungi adalah 1, 2, 4, 8, 5, 6, 3, 7.

E. DFS dengan Graf Dinamis



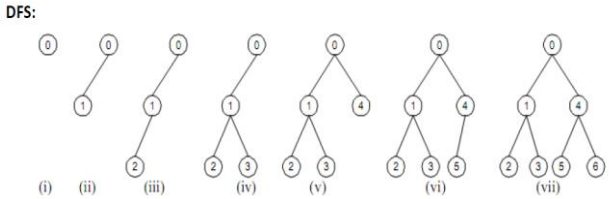
Gambar 3. Contoh graf representasi path
Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag2.pdf

Algoritma DFS (*Depth First* Search) dengan graf dinamis pada umumnya digunakan pada pencarian rute. Pencarian solusi dapat dilakukan dengan pembentukan pohon dinamis. Berikut adalah beberapa hal yang perlu

diperhatikan dalam menggunakan graf dinamis untuk menyelesaikan suatu persoalan:

1. Pada pembangkitan suatu simpul, setiap simpul diperiksa apakah merupakan suatu solusi (*goal*) atau tidak. Jika simpul merupakan suatu solusi, pencarian dapat diselesaikan (mendapatkan satu solusi) atau melanjutkan pencarian lagi (lebih dari satu solusi).
2. Simpul adalah sebuah *problem state* (membentuk solusi) terdiri atas:
    - Akar sebagai *initial state*
    - Daun sebagai *solution state*, satu atau lebih status yang menyatakan solusi persoalan
3. Cabang adalah operator/langkah dalam persoalan
4. Ruang status adalah himpunan semua simpul
5. Ruang solusi adalah himpunan status solusi
6. Solusi merupakan *path* ke status solusi

Terdapat proses pembangkitan status dengan cara mengaplikasikan operator atau langkah legal pada status simpul di suatu jalur. Jalur dari simpul akar ke simpul daun merupakan rangkaian operator yang mengarah ke solusi persoalan. Berikut adalah contoh pembentukan pohon ruang status dengan DFS.

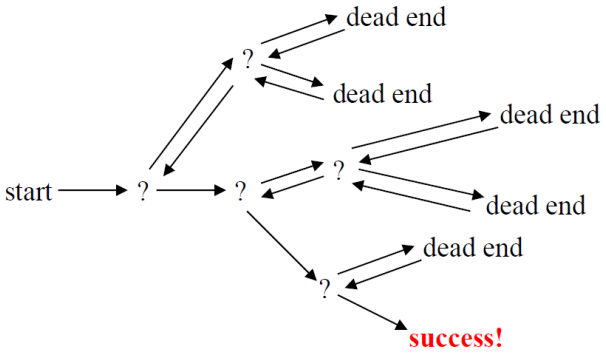


Gambar 4. Pembentukan pohon ruang status dengan DFS
Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag2.pdf

Berdasarkan gambar di atas, pohon ruang status dibentuk dengan menelusuri dan membentuk simpul secara mendalam. Kemudian ketika tidak ada lagi simpul yang dapat dikunjungi , misalnya pada simpul 2, maka akan runut-balik atau *backtrack* ke simpul 1. Setelah itu, akan dibangkitkan lagi simpul tetangga yang lain dari simpul 1 yaitu simpul 3. Dari simpul 3 sudah tidak ada lagi tetangga yang dapat dikunjungi maka *backtrack* ke simpul 1, dan seterusnya hingga sudah tidak ada lagi status atau simpul yang dapat dibangkitkan atau dapat juga berhenti ketika solusi dikatakan sudah ditemukan.

F. Aplikasi Baktracking di dalam DFS pada Proses Pencarian Solusi

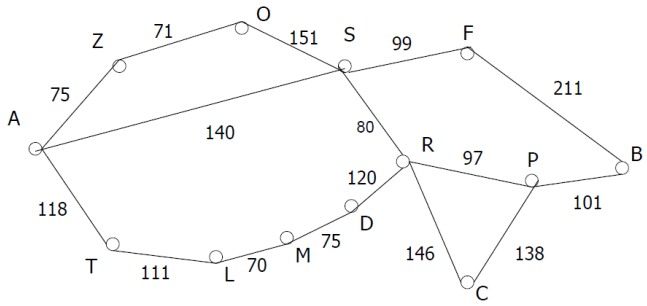
Backtracking pada algoritma DFS akan sangat berguna ketika terdapat banyak alternatif pilihan selama proses pencarian. Solusi didapatkan dengan mengekspansi pencarian menuju *goal* dengan aturan *depth*-first. *Bactracking* utamanya digunakan ketika tidak punya cukup informasi untuk mengetahui langkah apa yang akan dipilih, sebuah keputusan akan mengarah pada sekumpulan pilihan baru, dan beberpaa pilihan mungkin merupakan solusi dari persoalan. *Backtracking* dalam DFS adalah mencoba beberapa sekuens keputusan hingga menemukan suatu sekuens yang bekerja. Berikut adalah gambaran dari langkah *backtracking*.



Gambar 5. Gambaran langkah Backtracking pada DFS
Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag1.pdf

G. Pencarian Rute dengan Algoritma DFS

Berikut adalah contoh pencarian rute dari suatu simpul menuju simpul *goal*.

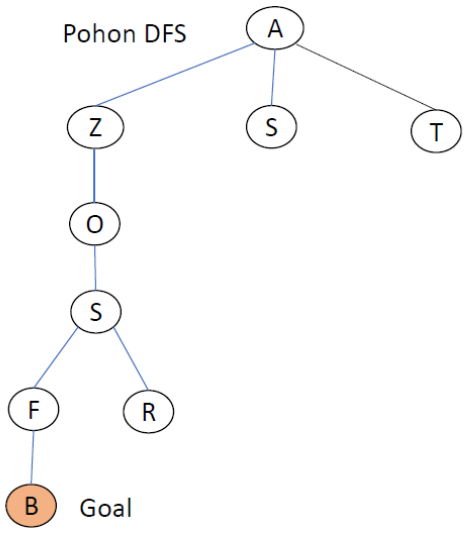


Gambar 6. Contoh rute yang digunakan
Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag2.pdf

Dilakukan pencarian rute dari simpul A menuju simpul Z. Berikut adalah *list* dari simpul ekspan dan simpul yang hidup.

Simpul Ekspan	Simpul Hidup
A	Z_A, S_A, T_A
Z_A	O_{AZ}, S_A, T_A
O_{AZ}	S_{AZO}, S_A, T_A
S_{AZO}	F_{AZOS}, R_{AZOS}, S_A, T_A
F_{AZOS}	B_{AZOSF}, R_{AZOS}, S_A, T_A
B_{AZOSF}	Solusi ditemukan

Berikut adalah graf dinamis yang akan terbentuk.



Gambar 7. Pohon dinamis yang terbentuk dari pencarian rute dari simpul A ke simpul B
Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag2.pdf

III. Aplikasi Algoritma DFS Dalam Pencarian Struktur Dunia Minecraft Menggunakan Custom Mod

A. *Macam-macam Daerah pada Minecraft*

Bioma dalam permainan *Minecraft* adalah daerah dengan geografi, macam-macam tanaman, dan karakteristik lain yang unik. Bioma pada *Minecraft* misalnya terdiri atas bioma *plains*, *forests*, *jungles*, *mountains*, *deserts*, *taiga*, *beach*, dan bioma-bioma lainnya. Sedangkan beberapa daerah unik di antaranya gua, laut, dan sungai. Tentunya beberapa daerah ini berperan penting dalam perkembangan pemain dalam permainan *Minecraft*. Namun dalam program yang penulis buat kali ini, program hanya akan membantu pemain untuk menemukan beberapa daerah krusial yang dapat membantu meningkatkan kehidupan pemain pada dunia *Minecraft* seperti sebuah desa, gua, sungai, ataupun hutan. Berikut adalah beberapa penjelasan mengenai daerah-daerah tersebut.

1. Gua

   Pada permainan *Minecraft* terdapat berbagai macam gua yang dapat dieksplorasi. Gua ini terletak secara acak di bawah permukaan dunia *Minecraft* dan pada umumnya menyediakan sumber daya berharga seperti batu, bijih, *iron*, *gold*, maupun diamond yang dapat meningkatkan daya tahan *player* dalam game. Beberapa sumber daya tersebut juga dapat digunakan untuk membuat beberapa perkakas yang sangat berguna bagi *player*. Terdapat berbagai macam gua, di antaranya gua batu karang, gua lava, gua lembah, gua gletser, gua hutan, gua abad kuno, dan gua air. Karena sumber dayanya yang melimpah, tempat ini merupakan salah satu tempat yang paling dicari oleh pemain.

2. Hutan

   Hutan pada *Minecraft* merupakan salah satu bioma yang sangat umum dan terdapat dalam berbagai variasi. Beberapa sumber daya yang disediakan hutan ini adalah pohon-pohon yang memiliki karakteristik unik yang dapat digunakan untuk membangun rumah, membuat berbagai macam perabot, dan merupakan komponen awal yang digunakan untuk membuat komponen-komponen lainnya.

3. Sungai

   Dalam permainan *Minecraft*, sungai merupakan salah satu fitur geografis yang sering ditemukan di berbagai macam bioma. Namun, kemunculan sungai ditemukan secara acak pada bioma. Mereka menambah keindahan dan juga memberikan beberapa variasi dalam lanskap permainan di antaranya sebagai alat transportasi dengan menggunakan sarana perahu dan merupakan tempat yang baik untuk mencari berbagai macam ikan yang dapat digunakan untuk menjaga kelangsungan hidup pemain pada dunia *Minecraft*.

4. Desa

Gambar 8. Pemukiman pada Minecraft
Sumber:
https://www.rockpapershotgun.com/minecraft-village-how-to-find-a-village

Desa merupakan salah satu fitur yang tak kalah penting dalam *Minecraft*. Desa adalah pemukiman yang dihuni oleh penduduk desa dan menyediakan berbagai macam sumber daya serta peluang perdagangan dengan para *villager*. Hal yang dapat dilakukan di desa ini salah satunya adalah pertanian, desa biasanya memiliki berbagai macam lahan pertanian di sekitarnya, di mana bisa ditanam berbagai macam tanaman seperti gandum, kentang, wortel, dan bit. Selain itu, desa juga dapat dijadikan sebagai tempat proteksi bagi pemain dari serangan para monster. Desa pada permainan *Minecraft* adalah pusat kehidupan dan aktivitas sosial yang menawarkan sumber daya, perdagangan, dan interaksi dengan penduduk desa. Desa juga

merupakan titik focus eksplorasi dan petualangan pada permainan.

Daerah-daerah yang telah disebutkan di atas merupakan beberapa daerah yang penting untuk ditemukan dan dieksplorasi oleh pemain. Maka digunakan algoritma DFS untuk menemukan beberapa daerah tersebut pada permainan *Minecraft.* Algoritma DFS ini kemudian dihubungkan dan diimplementasikan pada permainan *Minecraft* dengan menggunakan mod program yaitu *Minecraft Forge*.

B. *Minecraft Forge*

Figure *Minecraft Forge* adalah sebuah modding platform yang dapat digunakan untuk mengubah dan memodifikasi permainan *Minecraft*. *Forge* menyediakan fasilitas bagi para pemain untuk menginstal dan menggunakan mod-mod baru yang dikembangkan oleh suatu komunitas. Dengan menggunakan *forge*, para pemain dapat menambahkan *item* baru, mengubah tampilan visual, ataupun menambahkan *register command* yang ada.

Forge menyediakan *framework* dan API yang memungkinkan pengembang untuk membuat dan menambahkan mod dengan lebih mudah. Dengan menggunakan *forge*, para pengembang dapat dengan mudah mengakses dan memanipulasi berbagai elemen dalam permainan seperti blok, item, entitas, antarmuka, dan lainnnya. *Forge* meenyediakan *code template* yang dapat diunduh dari *official web forge*, kemudian dengan menggunakan IDE pilihan pengembang misalnya Intellij IDEA, pengembang dapat dengan mudah memodifikasi kode dan menambahkan berbagai macam fitur sesuai dengan kemauan pengguna.

Dengan menggunakan *Forge*, penulis dapat mengakses beberapa *item* atau *block* yang ada pada *Minecraft* menggunakan beberapa elemen berikut.

```
import com.mojang.logging.LogUtils;
import net.minecraft.core.BlockPos;
import net.minecraft.core.Vec3i;
import net.minecraft.world.level.block.Block;
import net.minecraft.world.level.block.Blocks;
import net.minecraft.world.level.block.state.BlockState;
import net.minecraft.world.level.storage.WorldData;
import net.minecraftforge.event.ServerChatEvent;
```

Gambar 9. Beberapa elemen yang dapat membantu proses modifikasi

Elemen-elemen pada gambar di atas merupakan beberapa elemen yang cukup sering digunakan dalam penerapan algoritma DFS. Elemen BlockPos digunakan untuk mengetahui *current position* dari *player*, elemen Block digunakan untuk mengetahui tipe block yang sedang dilewati player yang nantinya digunakan untuk mengklasifikasikan daerah, elemen BlockState merupakan *parent* dari Block, dan elemen *ServerChatEvent* yang digunakan untuk memeriksa *chat* dari pengguna apakah merupakan *command* untuk memulai algoritma DFS atau bukan.

C. *Penerapan Algoritma DFS dalam Pencarian Struktur Dunia Minecraft)*

Setelah beberapa langkah awal telah dilakukan untuk men-*setup Minecraft Forge code*, maka langkah selanjutnya adalah memodifikasi kode *template* yang telah disediakan. Salah satunya adalah menambahkan *event handler* untuk *command* dari pengguna berupa */startdfs*. Pada kode berikut *command* akan dihubungkan untuk memanggil algoritma DFS yang telah dibuat.

```
@SubscribeEvent
public static void onChatMessage(ServerChatEvent event) {
    String message = String.valueOf(event.getMessage());
    player = event.getPlayer();
    if (message.equalsIgnoreCase( anotherString: "startdfs")) {
        DFS(player.getCommandSenderWorld(), player.getOnPos());
    }
}
```

Gambar 10. Set up command untuk memanggil algoritma DFS dari chat

Selanjutnya adalah implementasi dari algoritma DFS dengan menggunakan komponen-komponen yang telah disediakan. Berikut adalah gambaran singkat dari implementasi dari algoritma DFS untuk pencarian struktur dunia *Minecraft*.

```
stack.push(startPos);

while (!stack.isEmpty()) {
    BlockPos currentPos = stack.pop();

    if (visited.contains(currentPos)) {
        continue;
    }

    visited.add(currentPos);

    BlockState blockState = world.getBlockState(currentPos);
    Block block = blockState.getBlock();

    if (block == Blocks.STONE) {
        player.sendSystemMessage(Component.literal( p_237114_: "Cave found at: " + currentPos.getX() + ", " +
    } else if (block == Blocks.ACACIA_LEAVES) {
        player.sendSystemMessage(Component.literal( p_237114_: "Acacia forest found at: " + currentPos.getX()
    } else if (block == Blocks.OAK_LEAVES) {
        player.sendSystemMessage(Component.literal( p_237114_: "Oak forest found at: " + currentPos.getX() + "
    } else if (block == Blocks.DARK_OAK_LEAVES) {
        player.sendSystemMessage(Component.literal( p_237114_: "Dark oak forest found at: " + currentPos.getX(
    } else if (block == Blocks.SAND) {
        player.sendSystemMessage(Component.literal( p_237114_: "Sea found at: " + currentPos.getX() + ", " + c
    }
    addNeighborsToStack(world, stack, currentPos);
}
```

Gambar 11. DFS pada struktur dunia Minecraft

```
public static void addNeighborsToStack(Level world, Stack<BlockPos> stack, BlockPos currentPos) {
    // Define the neighbor
    Vec3i[] neighborOffset = {
            new Vec3i( p_123296_: 1,  p_123297_: 0,  p_123298_: 0), // move to east
            new Vec3i( p_123296_: -1,  p_123297_: 0,  p_123298_: 0), // move to west
            new Vec3i( p_123296_: 0,  p_123297_: 1,  p_123298_: 0), // move up
            new Vec3i( p_123296_: 0,  p_123297_: -1,  p_123298_: 0), // move down
            new Vec3i( p_123296_: 0,  p_123297_: 0,  p_123298_: 1), // move to south
            new Vec3i( p_123296_: 0,  p_123297_: 0,  p_123298_: -1) // move to north
    };

    // adding neighbors to stack
    for (Vec3i offset : neighborOffset) {
        BlockPos neighborPos = currentPos.offset(offset);
        if (world.isInWorldBounds(neighborPos)) {
            stack.push(neighborPos);
        }
    }
}
```

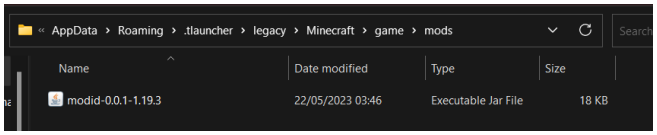
Gambar 12. Mekanisme penambahan neighbors

Kelas yang berisi *event handler* kemudian di-register pada *constructor* dari mod seperti sebagai berikut.

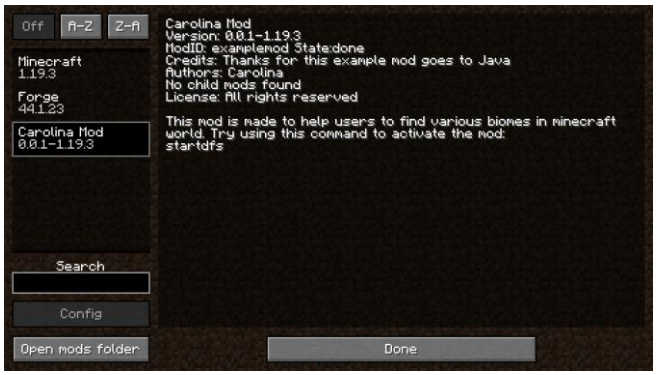
```
public ExampleMod() {
    IEventBus modEventBus = FMLJavaModLoadingContext.get().getModEventBus();
    // Register the commonSetup method for modloading
    modEventBus.addListener(this::commonSetup);
    // Register ourselves for server and other game events we are interested in
    MinecraftForge.EVENT_BUS.register( target: this);
    MinecraftForge.EVENT_BUS.register(new ChatEventHandler());
}
```

Gambar 13. Penambahan command pada mod constructor

Maka set up dari modifikasi *Minecraft* dengan algoritma DFS telah selesai, selanjutnya tinggal menghubungkan kode pada permainan. Seluruh kode dan kode template yang telah disediakan dalam format *gradle* tadi kemudian dijadikan sebuah jar dengan menuliskan *command* berikut pada terminal di *directory* kode ".\gradlew build". Setelah terbentuk sebuah jar, jar tersebut siap di-*copy* kan ke folder mods dari game *Minecraft* (path dari mods folder setiap *user* akan berbeda-beda). Berikut adalah contoh dari folder mods yang dapat dimasukkan sebuah jar.



Gambar 14. Contoh path untuk memasukkan jar file



Gambar 15. Tanda mod telah terbaca

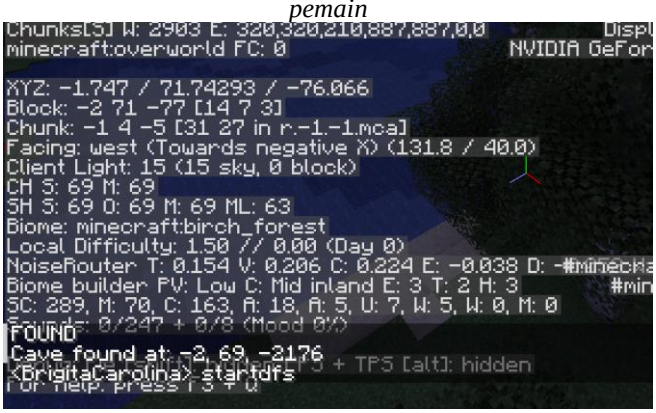
Setelah itu kode siap dites di dalam game. Berikut adalah beberapa tes yang telah dilakukan oleh penulis.

Gambar 16. Testing algoritma pada block air

Gambar 17. Cave dapat ditemukan dan sistem dapat mengirimkan suatu koordinat untuk diikuti oleh pemain

Gambar 18. Forest dapat ditemukan dan sistem dapat mengirimkan suatu koordinat untuk diikuti oleh pemain

Gambar 19. Sea dapat ditemukan dan sistem dapat mengirimkan suatu koordinat yang dapat diikuti oleh pemain



Gambar 20. Pemain dapat menemukan cave saat posisi pemain jauh dari cave

Berdasarkan beberapa tes yang telah dilakukan, algoritma DFS yang telah diaplikasikan tadi terlihat dapat membantu pemain dalam menemukan beberapa daerah yang krusial seperti *cave*, *forest*, dan lainnya.

IV. Kesimpulan

Dapat disimpulkan bahwa penerapan algoritma DFS dalam pencarian struktur dunia *Minecraft* dengan menggunakan custom mod *Minecraft Forge* dapat memberikan pengalaman bermain yang menarik dan dinamis. Melalui penggunaan modifikasi ini, pemain dapat memanfaatkan algoritma DFS untuk melakukan pencarian dan eksplorasi yang lebih efisien dalam lingkungan *Minecraft.* Penerapan algoritma DFS memungkinkan pemain untuk mengidentifikasi dan menjelajahi berbagai struktur seperti gua-gua, labirin, bangunan kompleks, gunung atau sungai dengan cara yang sistematis. Dengan memanggil *command* yang terhubung dengan algoritma DFS, pemain dapat langsung memanfaatkan keunggulan algoritma ini untuk menemukan koordinat daerah-daerah tersebut di sekitarnya.

Namun, dalam penggunaan modifikasi ini perlu diingat bahwa penerapan algoritma yang digunakan harus dengan memperhatikan aturan dan etika permainan. Penggunaan modifikasi ini perlu diikuti dengan menghormati integritas permainan, menghargai karya dan upaya para pengembang, dan mematuhi pedoman yang telah ditetapkan oleh Mojang dan komunitas *Minecraft.*

V. Ucapan Terima Kasih

Penulis mengucapkan syukur kepada Tuhan yang Maha Esa karena berkat rahmat-Nya penulis diberikan kelancaran agar sehingga dapat menyelesaikan makalah IF2211 Strategi Algoritma semester II tahun 2022/2023. Penulis juga ingin mengucapkan terima kasih kepada keluarga yang telah memberikan dukungan selama proses penulisan makalah ini. Penulis mengucapkan terima kasih kepada Ibu Dr. Nur Ulfa Maulidevi, S.T., M.Sc., selaku dosen pengampu mata kuliah IF2211 Strategi Algoritma kelas K2 yang telah memberikan materi untuk penulisan makalah ini.

Referensi

[1] https://docs.minecraftforge.net/en/1.19.x/ Diakses pada tanggal 21 Mei 2023.

[2] https://files.minecraftforge.net/net/minecraftforge/forge/ Diakses pada tanggal 21 Mei 2023.

[3] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag1.pdf Diakses pada tanggal 21 Mei 2023.

[4] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag2.pdf Diakses pada tanggal 21 Mei 2023.

[5] https://www.minecraft.net/en-us/about-minecraft Diakses pada tanggal 21 Mei 2023.

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, 22 Mei 2023

Brigita Tri Carolina
13521156