IMPLEMENTASI ALGORITMA *K-D TREE* PADA PEMBUATAN *3D DUNGEON GENERATOR* MENGGUNAKAN METODE PROSEDURAL MODELING

SKRIPSI

HARRIS KRISTANTO

191401066

PROGRAM STUDI S-1 ILMU KOMPUTER

FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI

UNIVERSITAS SUMATERA UTARA

MEDAN

2025

IMPLEMENTASI ALGORITMA *K-D TREE* PADA PEMBUATAN *3D DUNGEON GENERATOR* MENGGUNAKAN METODE PROSEDURAL MODELING

SKRIPSI

Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh ijazah
Sarjana Ilmu Komputer

HARRIS KRISTANTO

191401066

PROGRAM STUDI S-1 ILMU KOMPUTER

FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI

UNIVERSITAS SUMATERA UTARA

MEDAN

2025

PERSETUJUAN

Judul	:	IMPLEMENTASI ALGORITMA *K-D TREE* PADA PEMBUATAN *3D DUNGEON GENERATOR* MENGGUNAKAN METODE PROSEDURAL MODELING
Kategori	:	SKRIPSI
Nama	:	HARRIS KRISTANTO
Nomor Induk Mahasiswa	:	191401066
Program Studi	:	SARJANA (S-1) ILMU KOMPUTER
Fakultas	:	ILMU KOMPUTER DAN TEKNOLOGI INFORMASI UNIVERSITAS SUMATERA UTARA

Diluluskan di
Medan, 9 Januari 2025

Komis Pembimbing :

Pembimbing 1

Dr. Jos Timanta Tarigan, S.Kom., M.Sc.
NIP. 198501262015041001

Pembimbing 2

Pauzi Ibrahim Nainggolan, S.Komp., M.Sc
NIP. 198809142020011001

Diketahui/Disetujui
Program Studi S-1 Ilmu Komputer
Ketua,

Dr. Amalia, S.T., M.T.
NIP. 197812212014042001

PERNYATAAN

IMPLEMENTASI ALGORITMA *K-D TREE* PADA PEMBUATAN *3D DUNGEON GENERATOR* MENGGUNAKAN METODE PROSEDURAL MODELING

SKRIPSI

Saya mengakui bahwa skripsi ini adalah hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing telah disebutkan sumbernya.

Medan, 9 Januari 2025

Harris Kristanto

191401066

PENGHARGAAN

Puji dan syukur penulis panjatkan ke hadirat Tuhan, karena kasih karunia-Nya, penulis dapat menyelesaikan penyusunan skripsi ini sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer pada Program Studi S1 Ilmu Komputer, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara.

Skripsi ini tentu diiringi oleh bantuan dan kehadiran dari berbagai pihak selama masa penyelesaiannya. Oleh karena itu, penulis ingin menyampaikan ungkapan hormat dan terima kasih yang sebesar-besarnya atas bimbingan, doa, dukungan, ilmu, dan pelajaran yang telah diberikan oleh semua pihak selama penyusunan skripsi ini, di antaranya:

1. Bapak Prof. Dr. Muryanto Amin, S.Sos., M.Si selaku Rektor Universitas Sumatera Utara.
2. Ibu Dr. Maya Silvi Lydia, B.Sc., M.Sc., selaku Dekan Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara.
3. Ibu Dr. Amalia, S.T., M.T. selaku Ketua Prodi Program Studi Ilmu Komputer, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara.
4. Bapak Dr. Jos Timanta Tarigan, S.Kom., M.Sc. sebagai Dosen Pembimbing Pertama Peneliti yang telah banyak memberikan berbagai bimbingan, arahan dan dukungan dalam pengerjaan skripsi ini.
5. Bapak Pauzi Ibrahim Nainggolan, S.Komp., M.Sc. sebagai Dosen Pembimbing Kedua Peneliti yang juga telah banyak memberikan bimbingan dan saran dalam pengerjaan skripsi ini.
6. Bapak Prof. Drs. Poltak Sihombing, M.Kom., Ph.D. selaku Dosen Penguji Pertama yang memberikan masukan yang membangun dalam penyelesaian skripsi ini.
7. Ibu Sri Melvani Hardi S.Kom., M.Kom selaku Dosen Penguji Kedua yang memberikan masukan yang membangun dalam penyelesaian skripsi ini.
8. Seluruh tenaga pengajar dan pegawai Program Studi S-1 Ilmu Komputer Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara.

9. Orang tua dari keluarga peneliti, Lim Kian Huat dan Ko Giok Hoa yang memberikan dukungan semangat kepada Peneliti dalam menyelesaikan skripsi ini.
10. Teman-teman kuliah terdekat Peneliti, Vinny Agustine, Filbert, Christine Angelina Salim, Sinu S.Kom., Prisko Banoza, Ivan Joshua S.Kom., serta Niko S.Kom. Yang telah memberikan dukungan semangat dan bantuan dalam pengerjaan skripsi.
11. Teman-teman Peneliti seluruh keluarga besar Kom C angkatan 2019 Ilmu Komputer Universitas Sumatera Utara yang telah memberikan motivasi kepada Peneliti dalam pengerjaan skripsi.
12. Teman-teman seperjuangan Mahasiswa Ilmu Komputer Universitas Sumatera Utara stambuk 2019 yang telah mendukung dan membantu Peneliti selama pengerjaan skripsi.
13. Dan semua pihak yang telah banyak membantu dan mendukung Peneliti yang tidak bisa disebutkan satu per satu.

Sebagai penutup, penulis berharap semoga skripsi ini dapat memberikan manfaat bagi para pembaca. Penulis juga mengucapkan terima kasih kepada semua pihak yang telah memberikan bantuan dalam proses penyelesaian skripsi ini, serta memohon maaf atas segala kekurangan dalam penulisannya.

Medan, 9 Januari 2025

Penulis,

Harris Kristanto

NIM. 191401066

ABSTRAK

Industri *game* merupakan salah satu industri terbesar di dunia yang terus berkembang. Salah satu teknik yang mendukung perkembangan ini adalah *Procedural Content Generation* (PCG). PCG sering digunakan untuk menciptakan lingkungan *game*, khususnya game dengan genre *Role-Playing Game* (RPG). Dalam genre RPG, salah satu elemen yang penting adalah dungeon. Namun, *dungeon* yang dibuat tanpa metode PCG cenderung memiliki *layout* yang statis, sehingga mengurangi variasi dan tingkat pengalaman bermain. Penelitian ini bertujuan untuk mengembangkan 3D *Dungeon Generator* dalam bentuk *plugin* yang mampu menghasilkan *layout* struktural *dungeon* 3D yang variatif dengan metode *procedural modeling*. Metode yang digunakan berupa implementasi algoritma *K-D Tree* sebagai teknik partisi dalam pembentukan ruangan dungeon. Pemgembangan dilakukan menggunakan *Houdini Engine* untuk membentuk *3D Dungeon Generator* yang kemudian diuji pada *Unity Engine*. Hasil penelitian menunjukkan bahwa algoritma *K-D Tree* berhasil diimplementasikan dan mampu menghasilkan stuktur partisi ruangan 2D secara seimbang, dengan koridor yang menghubungkan semua ruangan. *Dungeon generator* yang dihasilkan mampu menghasilkan 3 hingga 30 ruangan dengan layout yang bervariasi. Kesimpulannya, implementasi algoritma *K-D Tree* efektif dalam mendukung proses *procedural modeling* dalam pembuatan *3D Dungeon Generator.*

Kata Kunci : *Procedural Content Generation* (PCG), *Dungeon Generator*, *K-D Tree*, *Procedural Modeling* dan *Houdini Engine*.

Implementation of K-D Tree Algorithm in Creation of 3D Dungeon Generator Using Procedural Modeling Method

ABSTRACT

The Gaming industry is one of the largest industries in the world that continues to grow. One of the techniques that support this development is Procedural Content Generation (PCG). PCG is often used to create game environments, especially games with the Role-Playing Game (RPG) genre. In the RPG genre, one of the important elements is the dungeon. However, dungeons created without the PCG method tend to have static layouts, reducing the variety and level of playing experience. This study aims to develop a 3D Dungeon Generator in the form of a plugin that is able to produce a variety of 3D dungeon layouts with the procedural modeling method. The method used is the implementation of the K-D Tree algorithm as a partition technique in forming dungeon rooms. Development was carried out using the Houdini Engine to form a 3D Dungeon Generator which was then tested on the Unity Engine. The results ot the study showed that the K-D Tree algorithm was succesfully implemented and was able to produce a balanced 2D room partition structure, with corridors connecting all rooms. The resulting dungeon generator was able to produce 3 to 30 rooms with varying layouts. In conclusion, the implementation of the K-D Tree algorithm is effective in supporting the procedural modeling process in creating a 3D Dungeon Generator.

Keywords : *Procedural Content Generation (PCG), Dungeon Generator, K-D Tree, Procedural Modeling and Houdini Engine.*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1

PENDAHULUAN

1.1. Latar Belakang

Industri *game* sudah menjadi salah satu industri terbesar. Berdasarkan data laporan dari analisis pasar *Newzoo* tentang Pasar Game Dunia 2022, jumlah *gamer* di seluruh dunia telah mencapai 3,2 miliar. Jumlah *gamer* yang semakin banyak terjadi karena *game* sudah menjadi salah satu sumber hiburan masyarakat umum, serta tingkat kompleksitas permainan yang semakin menarik dan berkembang sesuai dengan perangkat keras yang mendukung pembuatan dan permainan *game* tersebut.

Seiring meningkatnya perkembangan industri *game,* maka dibutuhkan inovasi baru untuk mendukung perkembangan tersebut. Salah satu inovasi yang digunakan adalah *Procedural Content Generation* (PCG). *Procedural Content Generation* merupakan metode penghasil konten secara otomatis untuk *game* yang dibuat dengan kumpulan algoritma dan bantuan komputer dengan bantuan manusia. (Viana, 2019). Dengan menggunakan *procedural content generation*, selain dapat menghasilkan konten yang bersifat dinamis, juga dapat mengurangi jumlah waktu serta biaya yang terpakai dalam pembuatan konten tersebut. (Shen, 2022). Salah satu jenis konten yang banyak dihasilkan menggunakan PCG merupakan konten *Dungeon*, disebut juga dengan *Procedural Dungeon Generation* (PDG). Beberapa komponen yang biasanya terdapat dalam pembuatan *dungeon* adalah ruangan *(room)*, tantangan, *puzzle* serta pembatas *(barrier)*. (Viana, 2021). Dungeon akan berhasil dibuat ketika minimal memiliki satu jalan penghubung yang menghubungkan ruangan utama ke ruangan lainnya pada *dungeon* tersebut. (Alvarez, 2019).

Dalam industri *game*, salah satu pengembangan *game* yang sering dilakukan adalah *game* yang memiliki tema *dungeon* 2D. Penggunaan model *dungeon* 2D dalam desain *dungeon* memberikan fleksibilitas serta kreativitas yang tinggi. Pembuatan *dungeon* 2D sudah banyak dilakukan dengan banyak metode seperti

dengan pembuatan ruangan secara acak yang dipadukan dengan algoritma *maze*. Sehingga, *dungeon* 2D telah menjadi bentuk umum dari banyak *game* dengan tema RPG ataupun petualangan. Pada penelitian yang dilakukan (Pratama, 2023) berhasil melakukan penelitian dalam pembuatan peta *dungeon* secara *top-down* dengan mengkombinasikan algoritma *Binary Space Partition* dengan *L-System*. Pada penelitian (Michael, 2022) berhasil melakukan penelitian pembuatan *2D Top-Down dungeon* secara prosedural dengan penggunaan *javascript* untuk melakukan pembuatan ruangan secara acak yang dikombinasikan dengan algoritma A* untuk pembuatan jalan penghubung antar ruangan. Meskipun *dungeon* 2D telah menjadi bentuk umum dari *game* bertema RPG, industri *game* selalu berkembang sehingga pengembangan *dungeon* 3D dapat dianggap sebagai respons terhadap dorongan industri tersebut. Dengan pengembangan *dungeon* 3D, maka tingkat pengalaman bermain suatu permainan game yang berkaitan dengan *dungeon* dapat meningkat.

Oleh karena pembuatan *dungeon* pada setiap *game* banyak yang statis, maka dilakukanlah penelitian pembuatan *dungeon* 3D dengan metode yang berbeda yaitu pembuatan *3D Dungeon Generator* secara Prosedural menggunakan bantuan algoritma *K-D Tree.* Berdasarkan hasil analisis algoritma dalam pembangkitan *dungeon*, proses pembangkitan *dungeon* secara prosedural harus terdiri atas ruangan yang direpresentasikan sebagai *node* serta jalan penghubung atau *corridor* yang dinyatakan sebagai *vertex* (Baron, 2017). *K-D Tree* merupakan algoritma yang digunakan untuk mengatur sejumlah titik ruang dalam dimensi k, dimana setiap level dari *k-d tree* terbagi menjadi 2 partisi dengan membagi jumlah titik yang sama pada tiap partisinya. (Otair, 2013). Sedangkan untuk *corridor* akan dilakukan penghubungan antar titik tengah tiap ruangan.

1.2. Rumusan Masalah

Pembuatan *dungeon* merupakan hal yang sering dilakukan pada pembuatan konten *game* dengan genre *Rogue-like* dan RPG. Perancangan dan pembuatan *dungeon* biasanya membutuhkan waktu yang cukup lama dan hasil yang didapatkan bersifat statis. Dengan memanfaatkan *Procedural Content Generation* maka pembuatan dungeon dapat menghasilkan dungeon yang bersifat acak serta otomatis. Untuk meningkatkan jenis *dungeon* dan kualitas permainan yang

berkaitan dengan *dungeon*, maka dilakukanlah implementasi Algoritma *K-D Tree* untuk menghasilkan *3D Dungeon Generator* yang bersifat dinamis dan otomatis.

1.3. Batasan Masalah

Untuk melakukan penelitian, penulis membatasi masalah yang akan diteliti. Batasan masalah dalam penelitian ini adalah :

1. *Dungeon* yang dihasilkan terdiri atas sejumlah ruangan yang dihubungkan dengan koridor berupa *cave*.
2. Pembuatan *Dungeon Generator* dengan menggunakan *K-D Tree Algorithm* serta metode pembuatan jalan penghubung antar ruangan.
3. Algoritma *K-D Tree* digunakan sebagai metode partisi membentuk ruangan.
4. Pemrograman dibuat dengan *Houdini Engine*.
5. Batas lingkup *Dungeon Generator* hanya terbatas pada penghasilan struktur dan layout saja, bukan termasuk pembangunan interior.

1.4. Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk mengimplementasikan algoritma *K-D Tree* pada pembuatan *3D Dungeon Generator* dengan pemrograman secara prosedural dengan *Houdini Engine*. Implementasi ini akan menghasilkan sebuah *dungeon generator* yang dapat menghasilkan *dungeon* secara acak untuk membantu para *game developer* dalam membangun permainan yang memerlukan pembuatan *dungeon* dalam *game*.

1.5. Manfaat Penelitian

Manfaat yang diperoleh dari penelitian ini adalah :

1. Menjadi referensi untuk penelitian atau pengembangan yang berkaitan dengan *procedural content generation*.
2. Menghasilkan *dungeon* dengan layout yang bersifat dinamis atau *random generated* dengan pemodelan secara prosedural.

1.6. Metodologi Penelitian

Metode Penelitian yang dilakukan dalam penelitian ini adalah :

1. Studi Pustaka

   Pada tahap ini penelitian dimulai dengan mencari referensi dari berbagai sumber terpercaya dan melakukan peninjauan pustaka melalui buku-buku, *e-book,* jurnal, artikel ilmiah yang berhubungan dengan *Procedural Content Generation*, *Dungeon* dan Algoritma *K-D Tree.*

2. Analisis dan Perancangan

   Berdasarkan ruang lingkup penelitian, penulis melakukan analisa terhadap apa saja yang akan dibutuhkan dalam penelitian untuk segera dirancang dalam sebuah diagram alir (flowchart).

3. Implementasi Sistem

   Pada tahap ini, membuat dungeon generator dengan menggunakan *Houdini Engine* dengan pemodelan prosedural sesuai dengan diagram alir yang telah dirancang seperti pada gambar diatas.

4. Pengujian Sistem

   Pada tahap ini, dungeon generator yang telah dirancang dilakukan tahap uji coba untuk mengetahui apakah dungeon generator berjalan sesuai tujuan dan dilakukan perbaikan jika ada kesalahan.

5. Dokumentasi

   Pada tahap ini, penelitian yang telah dilakukan akan didokumentasikan mulai dari tahap analisis sampai kepada tahap pengujian dalam bentuk skripsi.

1.7. Sistematika Penulisan

Berikut penjelasan tentang beberapa bagian penting dari sistematika penulisan skripsi.

BAB 1 PENDAHULUAN

Pada bab ini terdapat latar belakang masalah dan menjelaskan rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metode penelitian, dan sistematika penulisan skripsi.

BAB 2 LANDASAN TEORI

Dalam bab ini terdapat tinjauan teoritis dari algoritma-algoritma

yang digunakan, serta penjelasan tambahan yang berkaitan dengan penelitian ini.

BAB 3 ANALISIS DAN PERANCANGAN

Pada bab ini terdapat arsitektur umum untuk penelitian sistem dan analisis masalah untuk menganalisis hal-hal yang diperlukan untuk membangun sistem.

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Dalam bab ini mencakup proses implementasi hal-hal yang diperlukan untuk membangun sistem dan menguji masalah yang telah diuraikan sebelumnya.

BAB 5 KESIMPULAN DAN SARAN

Pada bab ini, penulis menguraikan kesimpulan dari pembahasan pada bab-bab sebelumnya dan memberikan saran sebagai masukan atas permasalah yang muncul dan diharapkan dapat membantu menyelesaikan permasalah tersebut.

BAB 2

LANDASAN TEORI

2.1. *Procedural Content Generation*

Salah satu komponen penting dari proses pembuatan *game* adalah proses pembuatan *content.* Konten yang dihasilkan berupa objek serta lingkungan yang akan diinteraksikan dengan pemain/*player*. (De Carli et al, 2011). Setiap konten yang dihasilkan memiliki dampak terhadap biaya serta waktu dalam pengerjaan sebuah proyek, yang berkaitan langsung dengan kemampuan artis dan model 3D yang dibuat. Seperti dalam mendesain *environment* utama yang menjadi tempat dimana pemain akan menghabiskan waktunya di sana. Waktu yang diinvestasikan untuk mendesain lingkungan utama tersebut dapat digunakan untuk berbagai hal lain berupa desain *event* game serta *game mechanic*.

Sebelumnya sudah disebutkan bahwa *Procedural Content Generation* (PCG) merupakan metode penghasil konten secara otomatis untuk *game* yang dibuat dengan kumpulan algoritma dan bantuan komputer dengan bantuan manusia. (Viana, 2019). Dalam beberapa tahun terakhir ini, penggunaan generasi *content* secara prosedural menjadi banyak digunakan pada pembuatan *game* terutama pada proses desain. Desain konten secara prosedural dapat digunakan menjadi landasan desain sehingga hanya perlu melakukan beberapa modifikasi. *Procedural generation* pada *video game* adalah generasi konten secara algoritmik yang dilakukan untuk meningkatkan *replay value* dan unsur yang tidak dapat diprediksi dalam game. Penggunaan metode *procedural generation* sudah banyak diaplikasikan dalam pembuatan konten berupa *game level, map, terrain, character model* dan *textures* (Liu Jian, 2021). Pada penelitian ini, akan dikhususkan untuk generasi *content game level* berupa *dungeon*. Contoh aplikasi *procedural dungeon* adalah *Minecraft Dungeon.*

Gambar 2.1. level permainan pada *Minecraft Dungeon*

2.2. *Dungeon*

Dungeon merupakan salah satu area yang sering ditemui dalam *role-playing game* atau yang biasanya disebut dengan RPG. Menurut Dahlskog (2015), *Dungeon* merupakan kumpulan ruangan serta lorong yang berisikan monster dan harta karun. Pada umumnya, setiap *dungeon* yang dibuat memiliki sebuah tema atau cerita yang menjadi bagian dari cerita atau dapat menjadi area diluar alur cerita yang dapat dijelajahi pemain.

Dalam suatu *video game*, bentuk *dungeon* yang umum ditemui terbentuk dari susunan ruangan-ruangan yang saling terhubung, jalan masuk serta jalan keluar. Serta umumnya jalan keluar serta jalan masuk *dungeon* akan terletak saling berjauhan sehingga memaksimalkan tingkat penjelajahan suatu *dungeon*.

Gambar 2.2. *Map Dungeon* pada *game Diablo III*

2.3. K-D Tree

Algoritma *K-D Tree* merupakan salah satu algoritma yang dipakai untuk menghasilkan beberapa partisi. *K-D tree* merupakan sebuah pohon *binary* dimana

setiap simpul merupakan suatu titik dengan k-dimensi (Otair, 2013). Setiap simpul dalam pohon didefinisikan oleh sebuah bidang melalui salah satu dimensi yang membagi set titik menjadi 2 bagian yaitu set kiri dan set kanan, masing-masing dengan setengah titik dari simpul induk. Anak-anak dari set kiri/kanan kemudian dibagi lagi menjadi setengah yang sama, menggunakan bidang dengan dimensi yang berbeda. Pembagian berhenti setelah log n tingkat, dengan setiap titik berada di sel daunnya sendiri. Pembagian berulang melalui dimensi yang berbeda untuk tingkatan yang berbeda dalam pohon, menggunakan titik median untuk pembagian.

K-D tree merupakan algoritma yang terdiri atas suatu struktur dengan n-dimensi untuk mengatur sejumlah n-dimensi data untuk pengambilan informasi yang cepat menggunakan pencarian asosiatif, di mana k mewakili dimensi ruang pencarian. Salah satu jenis struktur *index k-d tree* berdasarkan perspektif desain adalah median *k-d tree*. (Sumeet Gill, 2021)
Sesuai dengan jenis nya, median *k-d tree* berfungsi untuk melakukan pembagian struktur data dengan mengambil nilai tengah dari suatu data/dimensi.

Berikut algoritma Median *k-d tree* :

1. Masukkan sejumlah titik P pada dimensi k
2. Urutkan daftar titik yang ada pada dimensi k dan memilih titik median sebagai elemen pivot.
3. Sejumlah titik P dipisah menjadi 2 *subset* menggunakan garis pemisah vertikal melalui titik median koordinat-x dalam P.
    1. Setiap titik yang berada di sebelah kiri dan tepat berada di garis pemisah akan dimasukkan ke subset A.
    2. Setiap titik yang berada disebelah kanan garis pemisah akan dimasukkan ke subset B.

2.4. Pembuatan Koridor

Koridor merupakan ruang berupa jalan atau lorong memanjang untuk menghubungkan dua kawasan atau ruangan. Berikut merupakan tahapan dalam pembuatan jalan penghubung antar ruangan :

1. Pengumpulan titik tengah dari setiap ruangan
2. Setiap titik tengah dari ruangan ke ruangan lain akan dihubungkan dengan sebuah garis lurus (*line*)
3. Garis lurus tersebut kemudian akan di *resample* menjadi sekumpulan garis-garis pendek yang masih terhubungkan
4. Setiap titik dari kumpulan garis pendek yang sudah di *resample* diberikan efek *scattering* sehingga garis penghubung antar ruangan memiliki *volume* yang berbeda.
5. Dari setiap titik yang sudah di *scattering* akan dibentuk suatu *polywire* yang berbentuk seperti tabung.

2.5. Metode Prosedural Modeling

Prosedural Modeling merupakan suatu teknik dalam menghasilkan objek tiga dimensi menggunakan pendekatan konstruktif dan generatif. (Ullrich, 2010) Pendekatan generatif pada proses pemodelan suatu objek berkaitan dengan penggunaan algoritma komputasi untuk menghasilkan alternatif desain model secara otomatis berdasarkan kriteria atau parameter yang ditentukan. Pendekatan konstruktif berkaitan dengan kemampuan dan pengetahuan pembuat desain model yang menyesuaikan dengan perlu atau tidaknya tambahan desain model.

2.6. Penelitian yang Relevan

Berikut merupakan beberapa penelitian yang berkaitan dengan penelitian ini, antara lain :

1. Dalam penelitian yang dilakukan oleh (William Hamilton, 2019) dengan judul "*Procedural Generation of Three-Dimensional Game Levels with Interior Architecture*" menghasilkan ruangan berdasarkan aturan struktural yang ditentukan dengan tata bahasa bentuk berupa *graph traversal algorithm* yang kemudian disusun sesuai *fitness function* yang berupa parameter dalam pengaturan posisi ataupun sejumlah aturan yang harus diikuti dalam penempatan tiap ruangan yang dibangkitkan dan dihubungkan kembali dengan algoritma pencarian.

2. Pada penelitian (Dahlskog, S.,Björk, S., Togelius, J., 2015) dengan judul "*Patterns, Dungeons and Generators*" berisikan tentang desain-desain *dungeon* dan komponen-komponennya. Pada penelitian tersebut dijelaskan tentang desain-desain umum yang disusun atas beberapa pola yang dikategorikan menjadi *micro-pattern*, *meso-pattern*, dan *macro-pattern* yang sudah diaplikasikan pada *game-game* terkenal. Dimana setiap desain dari *game-game* tersebut memiliki keunikannya yang sesuai dengan *genre*-nya masing-masing.

3. Pada Penelitian (Alexander Baldwin, 2017) dengan judul "*Towards Pattern-Based Mixed-Initiative Dungeon Generation*" berisikan penelitian lanjutan terhadap penelitian Dahlskog terutama terkait pola-pola umum *micro-pattern, meso-pattern,* dan *macro-pattern* yang dipadukan dengan alat pembuat *Procedural Content Generation* yang berdasarkan pola-pola umum tersebut, dan didapatkan hasil bahwa pola-pola umum tersebut sangatlah berpengaruh terhadap variasi dalam tingkat kemenarikan suatu permainan.

4. Pada Penelitian (Sumeet Gill, 2021) dengan judul "*The Design Perspective of the Structures Based on k-d Tree*" berisikan penelitian terkait pengindeksan struktur data yang berdasarkan *k-d tree*. Pada penelitian tersebut, telah dijelaskan 14 jenis struktur pengindeksan data berdasarkan *k-d Tree*, dan diantaranya merupakan Median *k-d Tree* yang bisa dipakai untuk melakukan partisi terhadap suatu dimensi secara merata.

5. Dalam penelitian yang dilakukan oleh (Zhenyuan Shen, 2022) dengan judul "*Procedural Generation in Games: Focusing on Dungeons*" menganalisis bahwa sebuah *procedural generation* terbagi menjadi tiga bagian yaitu pembangkit daratan, pembangkit konten, dan pembangkit objek. Penelitian ini menggunakan *Perlin Noise* dalam membuat suatu pembangkit daratan dengan hasil suatu daratan yang bagus dan terlihat natural, serta penggunaan *Perlin Noise* ini juga dapat diaplikasikan dalam pembuatan suatu dungeon secara prosedural.

BAB 3

ANALISIS DAN PERANCANGAN SISTEM

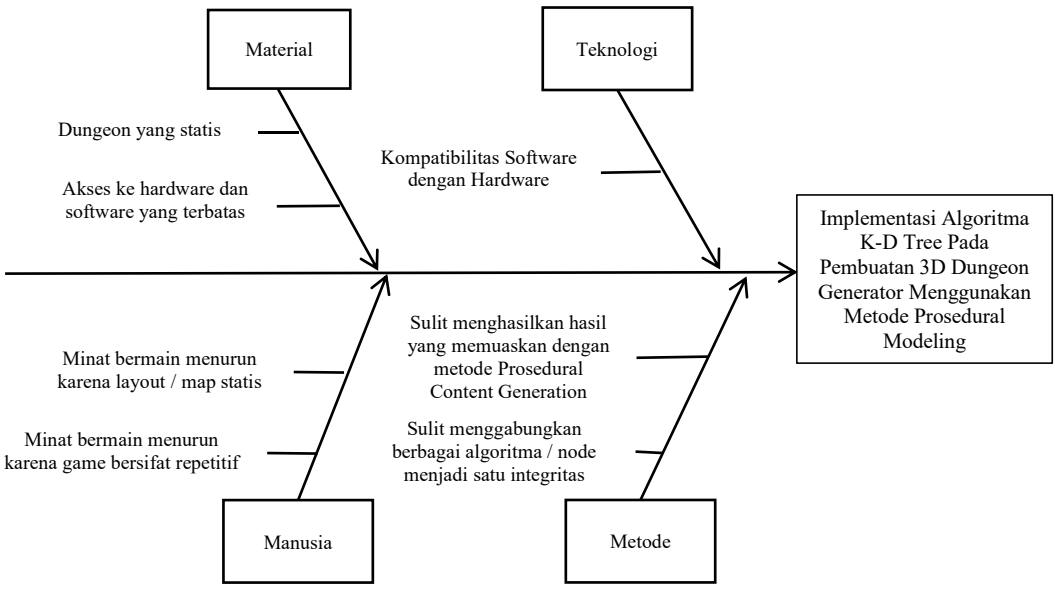
3.1. Analisis Sistem

Analisis Sistem merupakan tahap awal dalam pembuatan sistem yang bertujuan untuk mendapatkan solusi dalam mengatasi masalah pada sistem dengan cara membedah masalah menjadi bagian-bagian agar dapat dipelajari dan dievaluasi. Proses ini meliputi analisa *input,* proses, *output*, dan batasan-batasan yang diperlukan. Analisis sistem terdiri atas analisis masalah dan analisis kebutuhan

3.1.1. Analisis Masalah

Analisis masalah merupakan sebuah proses yang bertujuan untuk mengidentifikasi masalah yang dihadapi oleh sistem agar sistem dapat mengatasi masalah tersebut dan berjalan sesuai dengan tujuannya. *Game design* merupakan salah satu proses yang penting dalam pembuatan suatu permainan, dimana proses desain game dapat memakan waktu yang lama karena beberapa tahapan yang harus dilakukan berupa *prototyping*, *testing* dan *maintenance*. *Dungeon* merupakan salah satu hal yang penting dalam game dengan genre *rogue-like* dan *role-playing*. Pembuatan dungeon dapat memakan waktu yang lama, serta hasil dungeon yang didapatkan berupa dinamis. Oleh karena itu, dibuatlah suatu *generator* untuk menghasilkan *layout 3D dungeon*. Salah satu metode yang dapat digunakan adalah *Procedural Content Generation* (PCG). PCG dapat diaplikasikan menggunakan banyak metode serta algoritma, contohnya seperti Algoritma median *k-d tree*.

Analisis masalah pada penelitian ini dilakukan dengan menggunakan diagram Ishikawa (*fishbone diagram*), diagram Ishikawa (*fishbone diagram*) berfungsi untuk mengidentifikasi sebab-akibat dari sebuah masalah. Gambar 3.1 merupakan Diagram Ishikawa (*fishbone diagram*) penelitian ini.



Gambar 3.1. *Diagram Ishikawa (Fishbone Diagram)*

3.1.2. Analisis Kebutuhan

Analisis kebutuhan merupakan sebuah proses yang bertujuan untuk mendapatkan dan menjelaskan kemampuan dari sistem yang akan dibuat. Analisis kebutuhan terdiri atas kebutuhan fungsional dan non-fungsional.

3.1.2.1. Analisis Kebutuhan Fungsional

Kebutuhan fungsional sistem merupakan penjelasan tentang fitur-fitur yang dapat dilakukan oleh sistem. Kebutuhan fungsional pada sistem yang akan dibuat adalah :

1. Sistem harus dapat membangkitkan kumpulan ruangan terpisah dengan algoritma Median *K-D Tree*.
2. Sistem harus dapat membangkitkan jalur penghubung antar ruangan.
3. Semua ruangan atau area yang dibangkitkan sistem harus dapat diakses oleh pemain.

3.1.2.2. Analisis Kebutuhan Non-Fungsional

Kebutuhan non-fungsional sistem merupakan penjelasan tentang bagaimana sistem fungsi, tanpa secara langsung berkaitan dengan fitur-fitur sistem. Kebutuhan non-fungsional pada sistem yang akan dibuat adalah :

1. Sistem dapat menghasilkan *layout 3D Dungeon* dalam waktu yang efisien.
2. Sistem tidak memerlukan perangkat dengan spesifikasi yang tinggi.
3. Sistem dapat digunakan dalam proses *game development.*

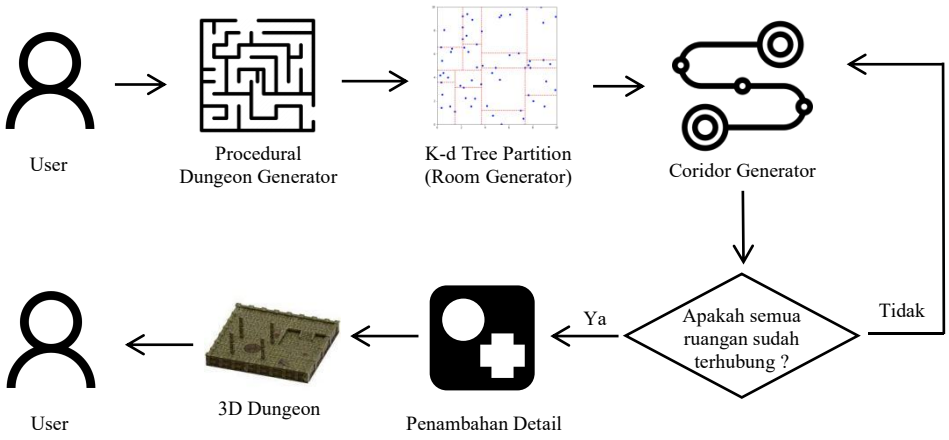
3.2. Penjelasan Penelitian

Pada penelitian ini, akan dibuat suatu sistem untuk menghasilkan 3D Dungeon Generator. Sistem akan mengimplementasikan algoritma *median k-d tree* sebagai metode partisi pada proses generasi sekumpulan area yang akan diproses lagi menjadi berbagai ruangan berupa *cave*. Hasil dari partisi dengan *median k-d tree* berupa sekumpulan grid atau area yang memiliki panjang serta lebar yang berbeda.

Setelah kumpulan area untuk membentuk ruangan telah dibangkitkan, sistem akan melanjutkan pembentukan ruangan berupa *cave* serta penghubung antar ruang dengan jarak terdekat.

3.2.1. General Arsitektur Umum

General Arsitektur Umum merupakan desain dan struktur keseluruhan dari sebuah sistem yang meliputi relasi, alur dan interaksi antara setiap komponen dalam suatu sistem. Pada gambar 3.2 terdapat perancangan keseluruhan aplikasi.



Gambar 3.2. Arsitektur Umum Sistem

User akan menjalankan program, selanjutnya *procedural dungeon generator* akan dimulai dengan menjalankan *partition* terhadap suatu grid untuk membagi grid menjadi area yang lebih kecil dengan ukuran yang berbeda. Kemudian, dilakukan pemilihan sejumlah area secara acak. Selanjutnya, *Procedural Dungeon*

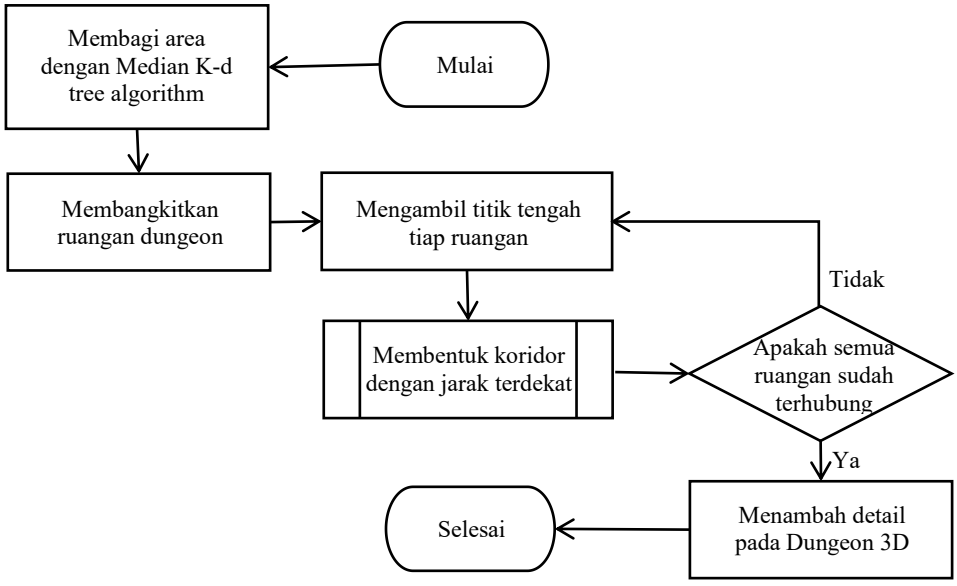
Generator akan membangkitkan koridor terhadap sejumlah area sehingga semua area tersambung satu sama lainnya. Selanjutnya, dilakukan penambahan detail tekstur *cave*. Kemudian, *dungeon* 3D tersebut dapat diakses oleh user.

3.3. *Flowchart*

Flowchart merupakan diagram simbol yang menggambarkan urutan proses atau tahapan dalam bentuk simbol-simbol yang dihubungkan oleh tanda panah yang menunjukkan arah dan hubungan antar proses.

3.3.1. *Flowchart Dungeon Generator*

Pada Gambar 3.3 *Flowchart Dungeon Generator*, sistem dimulai dengan membagi area menggunakan algoritma Median *K-d tree*. Kemudian, dilanjutkan pemilihan area secara teracak sesuai dengan jumlah ruangan yang telah ditentukan. Kemudian, ruangan-ruangan dungeon dibentuk berupa *cave*. Setelah ruangan dibangkitkan, dilakukan pengambilan titik tengah tiap ruangan. Setiap titik tengah ruangan akan disambungkan satu sama lainnya dengan jarak terdekat dan menghasilkan koridor. Proses penyambungan titik tengah tiap ruangan akan dilakukan hingga semua ruangan tersambung satu sama lainnya seperti suatu *graph*. Setelah koridor terbentuk, dilakukan penambahan detail terhadap *Dungeon* 3D berupa penambahan *texture*. Setelah penambahan detail, *Dungeon Generator* selesai membentuk *dungeon* 3D yang tersambung satu sama lainnya.

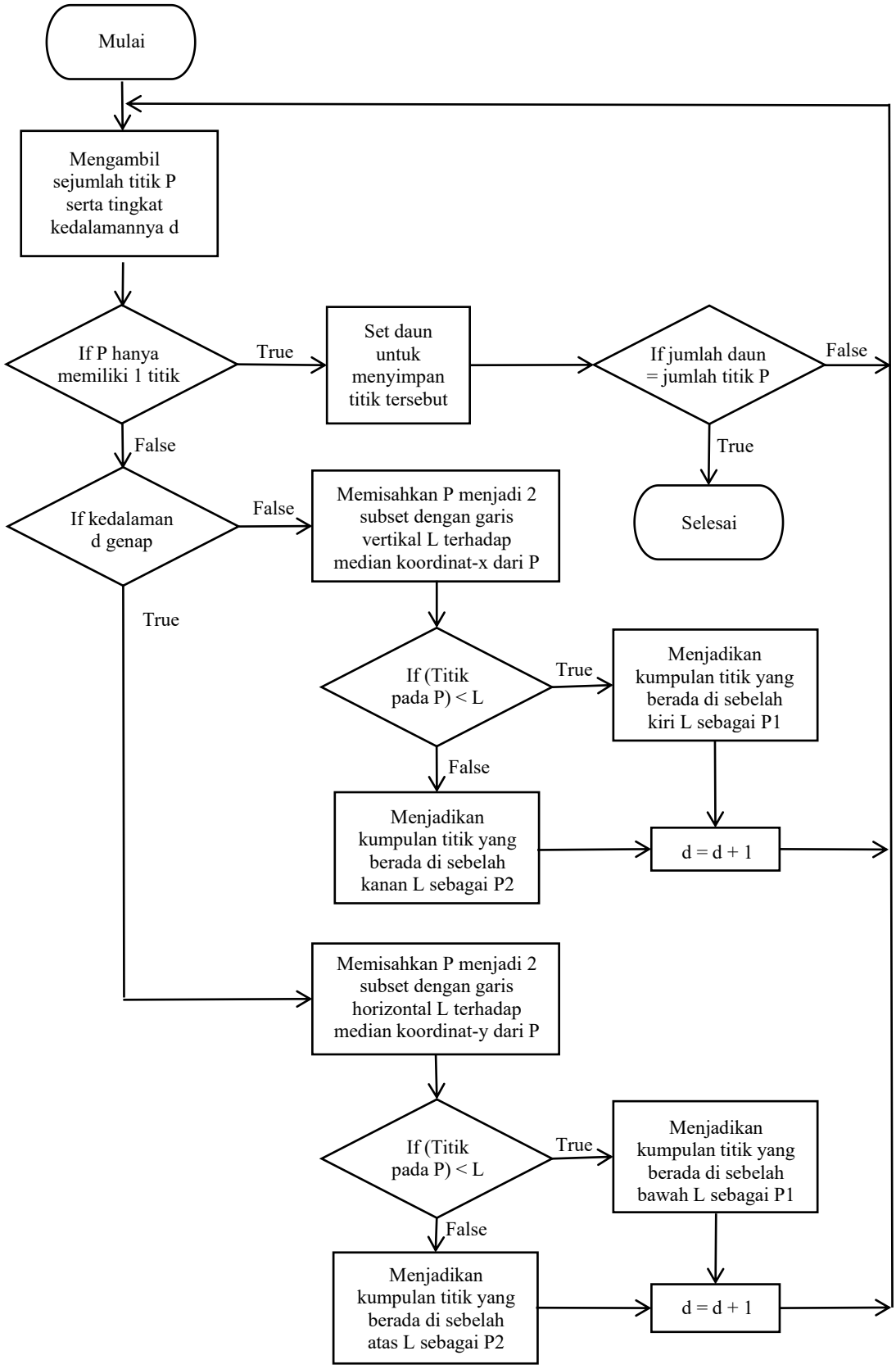


Gambar 3.3. *Flowchart Dungeon Generator*

3.3.2. *Flowchart* Median *K-D Tree*

Algoritma Median *K-D Tree* merupakan salah satu algoritma dari *k-d tree* yang biasanya dipakai dalam mengorganisasikan sejumlah titik pada dimensi k. Algoritma Median *K-D Tree* akan dipakai sebagai pemisah sejumlah titik yang terdapat pada grid / area dua dimensi. Algoritma Median *K-D Tree* akan menjadi algoritma yang memulai proses pembangkitan ruangan dungeon dengan membagi area grid berdasarkan median dari sejumlah titik pada sejumlah area.

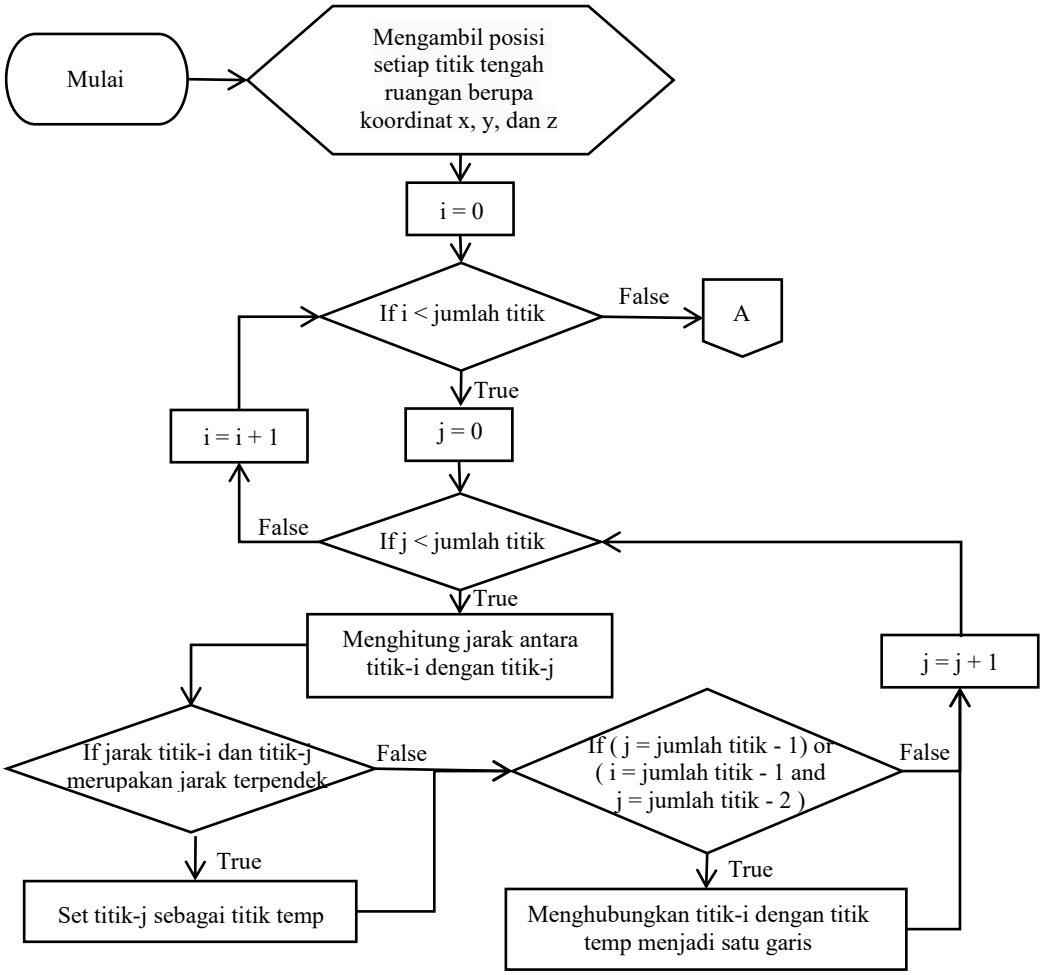
Flowchart Median *K-D Tree* dapat dilihat pada Gambar 3.4. Algoritma Median *K-D Tree* akan mengambil sejumlah titik P serta tingkat kedalamannya d. Kemudian, dilanjutkan dengan pengecekan jumlah titik P. Apabila jumlah titik P hanya berjumlah satu, maka set daun untuk menyimpan titik tersebut. Serta jika jumlah daun dan jumlah titik mempunyai nilai yang sama maka partisi algoritma Median K-D Tree selesai. Apabila jumlah titik P berjumlah lebih dari 1 titik, maka dilanjutkan dengan pengecekan tingkat kedalamannya d. Jika tingkat kedalaman bernilai genap, dilakukan pembagian P menjadi 2 subset berdasarkan garis horizontal pembagi L. Jika titik yang dipisah berada di posisi lebih kecil daripada garis pembagi L, maka titik tersebut diletakkan pada area bawah. Sedangkan, jika titik berada di posisi lebih besar, maka titik tersebut diletakkan pada area atas. Apabila tingkat kedalaman bernilai ganjil, dilakukan pembagian P menjadi subset berdasarkan garis vertikal pembagi L. Jika titik yang dipisah berada pada posisi lebih kecil dari garis vertikal pembagi L, maka titik tersebut diletakkan pada area subset sebelah kiri. Sedangkan, jika titik berada pada posisi lebih besar dari garis vertikal pembagi L, titik tersebut diletakkan pda area subset sebelah kanan. Proses partisi akan dilakukan secara rekursif hingga selesai.



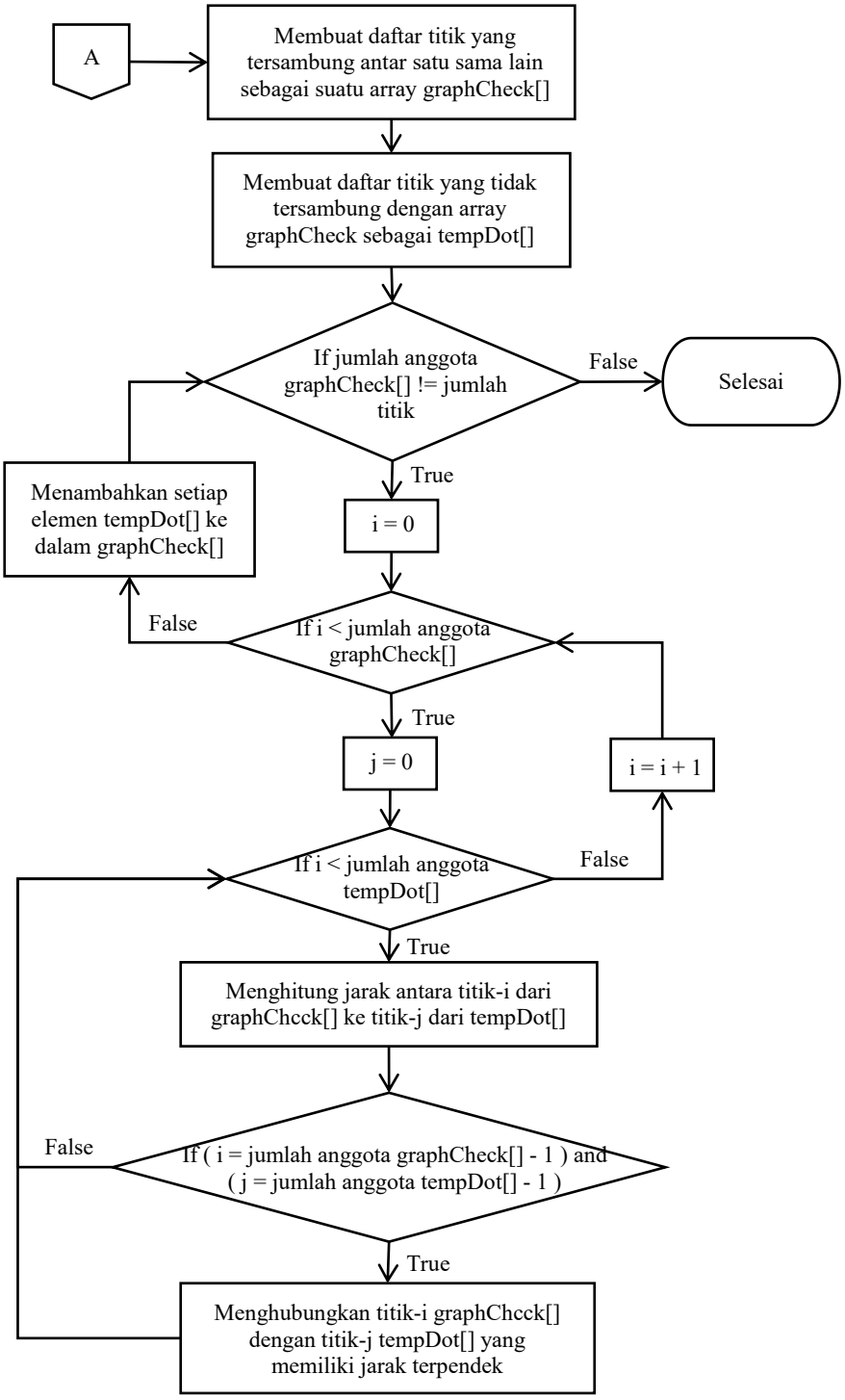
Gambar 3.4. *Flowchart Median K-D Tree*

3.3.3. *Flowchart* Pembuatan Koridor

Pembuatan koridor merupakan salah satu bagian dari proses pembuatan *dungeon*. Flowchart Pembuatan Koridor dapat dilihat pada Gambar 3.5a dan Gambar 3.5b. Proses pembuatan koridor akan dimulai dengan pengambilan posisi koordinat x, y dan z dari setiap titik tengah ruangan. Setelah setiap koordinat titik tengah ruangan didapatkan, dilanjutkan dengan proses pembuatan n-jumlah garis penghubung terhadap setiap titik tengah ruangan secara berurutan dengan jarak terpendek. Setelah pembuatan n-jumlah garis, dilanjutkan pengecekan terhadap kumpulan garis terkait sudah atau tidaknya terhubung menjadi satu kesatuan antara satu garis dengan garis lainnya. Apabila semua garis belum terhubung seluruhnya, maka proses berlanjut dengan mencari jarak terpendek antara garis yang sudah terhubung dengan garis yang belum terhubung serta dilakukan pembuatan garis tambahan diantara garis tersebut. Apabila semua garis sudah terhubung secara keseluruhan, maka proses pembuatan koridor selesai.



Gambar 3.5a. Flowchart Pembuatan Koridor bagian pertama



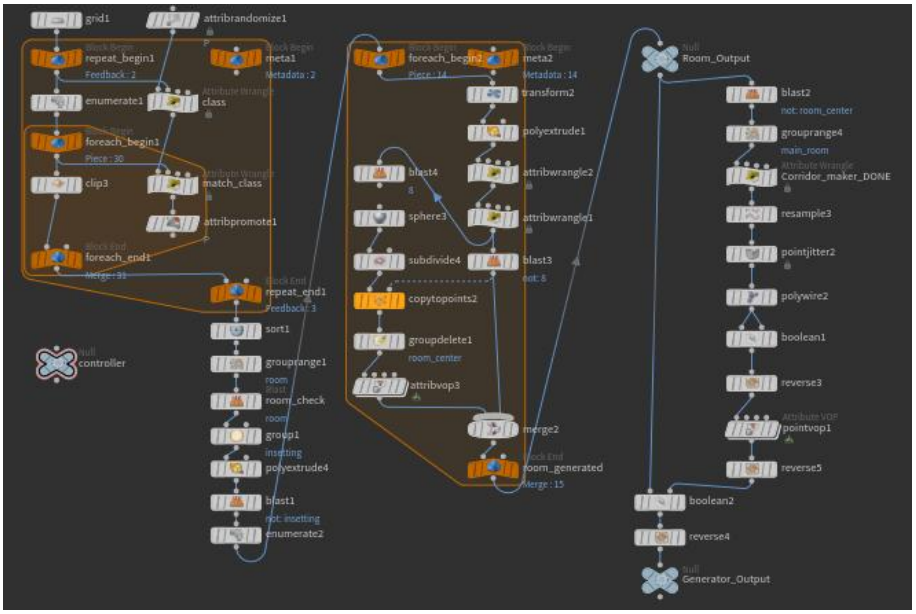
Gambar 3.5b. Flowchart Pembuatan Koridor bagian kedua

BAB 4

IMPLEMENTASI DAN PENGUJIAN

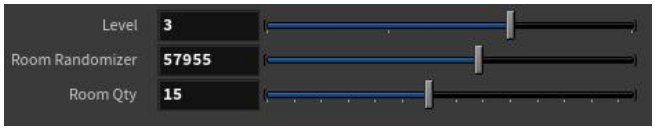
4.1. Hasil *Dungeon Generator*

Kumpulan *nodes* yang dipakai dalam melakukan pembangunan *Dungeon Generator* tertera pada Gambar 4.1, yang terbagi menjadi 3 bagian utama diantaranya berupa Partisi Median *K-D Tree*, Pembuatan Ruangan *Dungeon* serta Pembuatan Koridor.



Gambar 4.1. *Nodes* Pembentuk *Dungeon Generator*

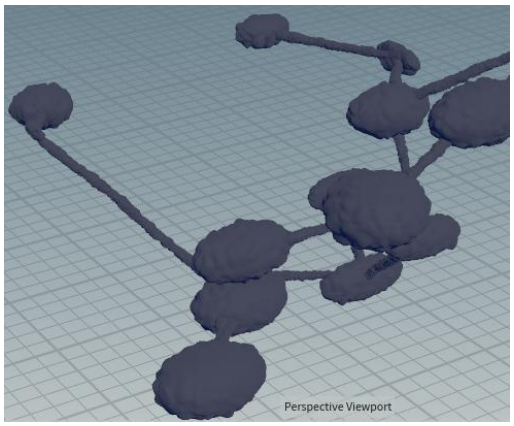
Dungeon Generator dibangkitkan secara prosedural dimana salah satu node yang bertuliskan '*Controller'* berisikan sejumlah parameter seperti pada Gambar 4.2 yang berfungsi untuk memberikan nilai terhadap node-node lain dalam pembangkitan 3D *Dungeon*.



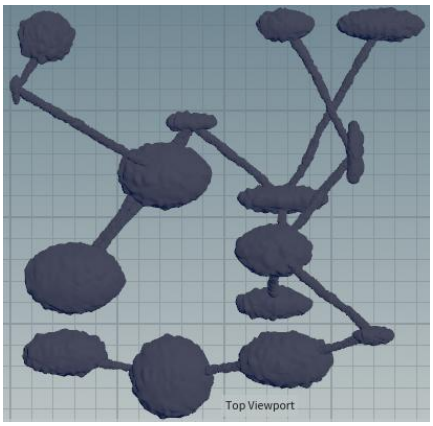
Gambar 4.2. Parameter pada *node* '*controller*'

Parameter yang terdapat pada Gambar 4.2 diantaranya adalah *Level*, untuk menentukan tingkat ketinggian Dungeon, *Random Randomizer* sebagai *seed* untuk

menentukan nilai acak, serta *Room Qty* untuk menentukan jumlah ruangan *dungeon* yang dihasilkan. Berdasarkan Gambar 4.2, maka *Dungeon* yang dihasilkan akan memiliki 15 (lima belas) jumlah ruangan, *seed* 57955 dan tingkat ketinggian *level* 3.



Gambar 4.3. *3D-Perspective View Dungeon*

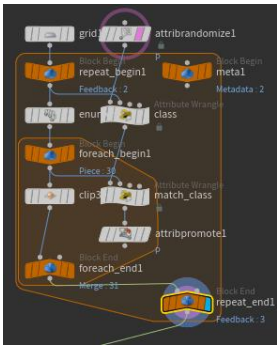


Gambar 4.4. *Top-View Dungeon*

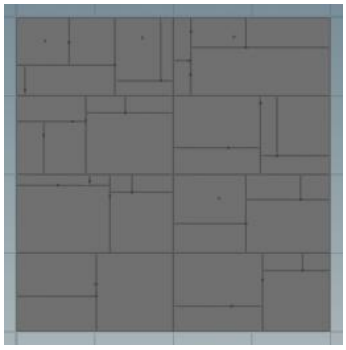
Pada Gambar 4.3 dan Gambar 4.4 terlihat hasil *Dungeon* dengan *Viewport* secara *Perspective* dan *Top View* dimana Dungeon yang dihasilkan mengikuti parameter *level* 3, *seed* 57955 serta ruangan dengan jumlah 15.

4.1.1. Implementasi Partisi Median *K-D Tree*

Nodes yang berfungsi untuk melakukan partisi dengan menggunakan algoritma Median *K-D Tree* terdapat pada Gambar 4.5, menghasilkan hasil partisi yang tertera pada Gambar 4.6.

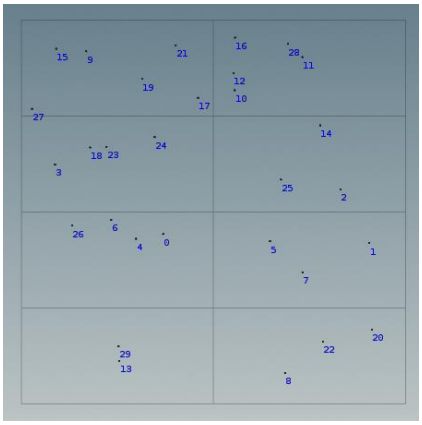


Gambar 4.5. *Nodes* Partisi Median *K-D Tree*



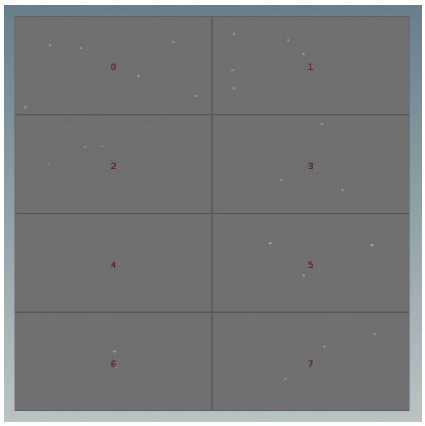
Gambar 4.6. Hasil Akhir Partisi Median *K-D Tree*

Hasil partisi pada Gambar 4.6 diawali dengan pembuatan *grid* dengan jumlah 4 baris serta 2 kolom, dilanjutkan dengan pengambilan kumpulan titik acak yang dihasilkan pada bidang xz sesuai dengan ukuran *grid* 20x20 menggunakan *node* ‘attribrandomize1’ seperti pada Gambar 4.7.



Gambar 4.7. Tampilan *wireframe grid* dan titik acak dari *node* 'attribrandomize1'

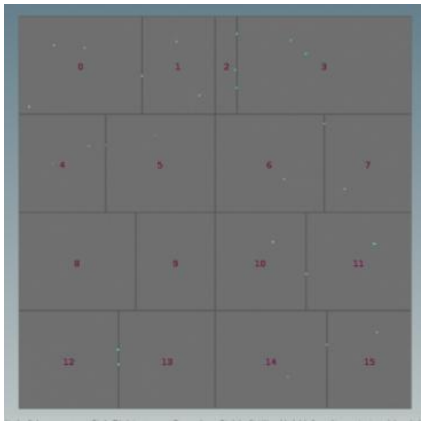
Sebelum proses partisi Median *K-D Tree*, dilakukan pemberian *Index* terhadap *primitive* yang dihasilkan sesuai jumlah baris x jumlah kolom, serta dilakukan pemberian warna terhadap titik-titik acak untuk menentukan posisi kedalaman titik dalam bidang *primitive* seperti pada Gambar 4.8.



Gambar 4.8. Penomoran bidang *grid* serta pemberian warna titik

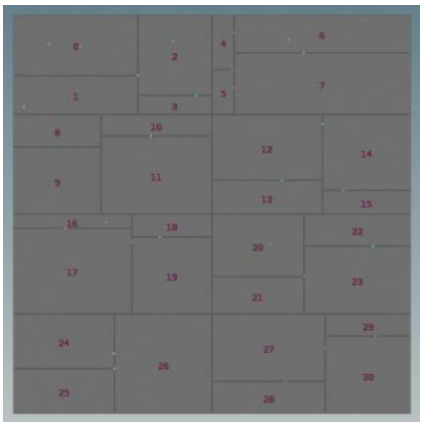
Proses partisi dilakukan secara iterasi dan berurutan sesuai dengan bidang *primitive* sebagai tingkat kedalaman. Setiap *primitive* dilakukan proses pencarian titik yang berada di tengah bidang *primitive*. Setelah titik median ditemukan, maka titik tersebut dinyatakan sebagai pivot, dan dilakukan pembuatan garis untuk membagi *primitive* menjadi berdasarkan posisi koordinat xyz titik pivot serta tingkat iterasinya.

Pada tahap iterasi pertama, pembuatan garis terhadap titik pivot dibuat terhadap sumbu z (vertikal) pada setiap *primitive* dan menghasilkan partisi seperti Gambar 4.9a.

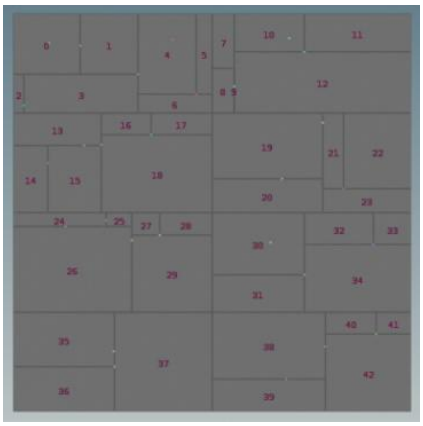


Gambar 4.9a. Partisi pertama

Setelah partisi pertama, dilanjutkan oleh partisi kedua, dimana pembuatan garis terhadap titik pivot dibuat terhadap sumbu x (horizontal) terhadap setiap *primitive* seperti Gambar 4.9a. Pada Gambar 4.9a, *primitive* yang ke-13 tidak terdapat titik untuk melakukan partisi, sehingga partisi dilakukan terhadap setiap *primitive* selain *primitive* ke-13 dan menghasilkan partisi seperti Gambar 4.9b. Partisi ketiga dilakukan terhadap sumbu z (vertikal) dan menghasilkan partisi seperti pada Gambar 4.9c.



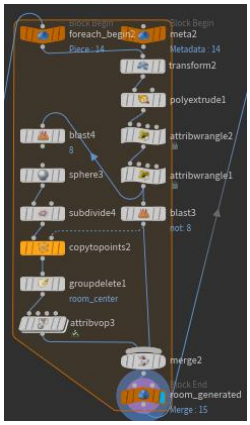
Gambar 4.9b. Partisi kedua



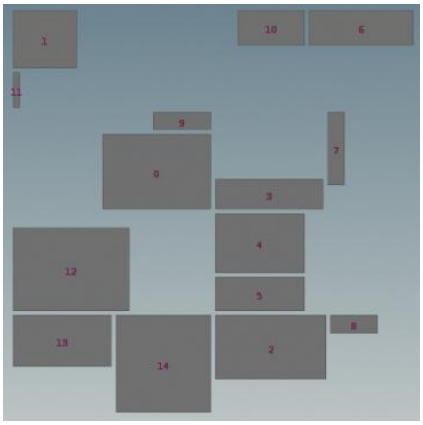
Gambar 4.9c. Partisi ketiga

4.1.2. Implementasi Pembuatan Ruangan *Dungeon*

Setelah proses partisi selesai, maka dilanjutkan dengan kumpulan *nodes* yang dipakai pada proses pembentukan ruangan yang tertera pada Gambar 4.10. Proses pembentukan ruangan diawali terlebih dahulu dengan pengacakan dengan *node sort* terhadap penomoran *primitive* dan pemilihan dasar ruangan secara acak berdasarkan parameter *Room Qty*. Dilanjutkan dengan pengecilan *primitive* agar tiap ruangan tidak tergabung satu sama lainnya seperti pada Gambar 4.11.

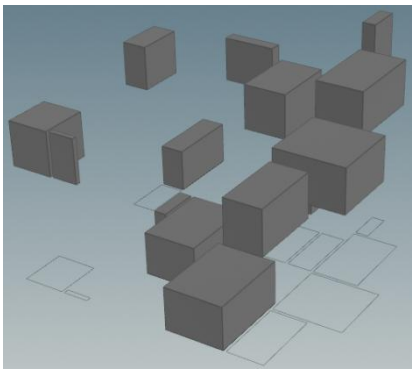


Gambar 4.10. *Nodes* Ruangan *Dungeon*



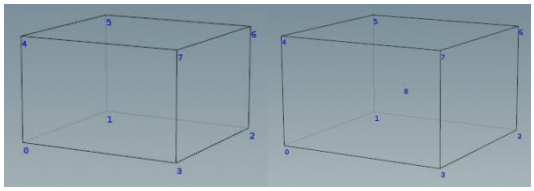
Gambar 4.11. Hasil Pemilihan Dasar Ruangan

Setelah pemilihan dasar ruangan, dilanjutkan dengan mengubah ketinggian dasar dengan *transform y-axis* serta *extrude* untuk memberi ketinggian ruangan hingga membentuk ruangan-ruangan berbentuk balok seperti Gambar 4.12.



Gambar 4.12. Hasil *Transform* y-axis dan Extrude

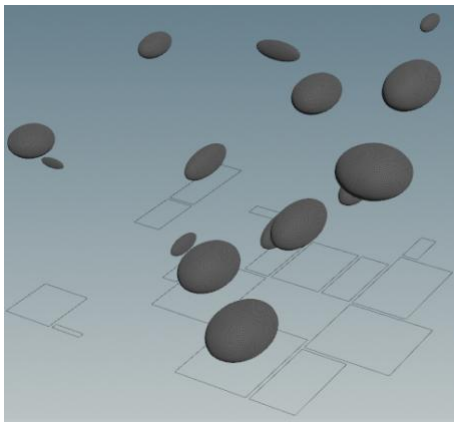
Setelah balok ruangan terbentuk, dilakukan penambahan titik tengah pada tiap balok ruangan seperti pada *primitive* ke-2 pada Gambar 4.13.



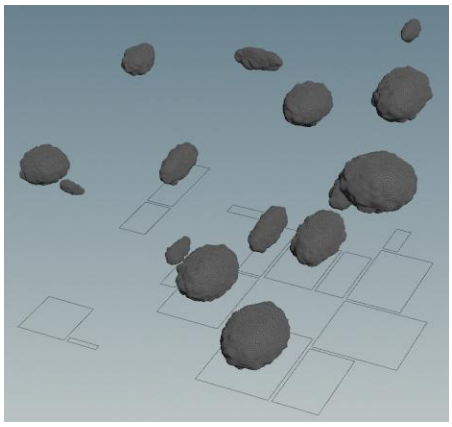
Gambar 4.13. *Wireframe* view sebelum (kiri) dan sesudah (kanan) penambahan titik tengah ruangan

Balok-balok ruangan kemudian diubah bentuknya menjadi bentuk *sphere* dengan mengambil jarak dari setiap 2 sisi berhadapan pada balok ruangan yang dibagi 2 sebagai nilai radius untuk sisi x, y dan z *sphere* sehingga menghasilkan Gambar 4.14. Selanjutnya, setiap *sphere* yang sudah terbentuk diberikan *detail*

bergelombang terhadap sisi *primitive* secara acak sehingga menghasilkan ruangan-ruangan yang lebih berbentuk *cave* seperti pada Gambar 4.15.



Gambar 4.14. Ruangan *Dungeon* berbentuk *Sphere*

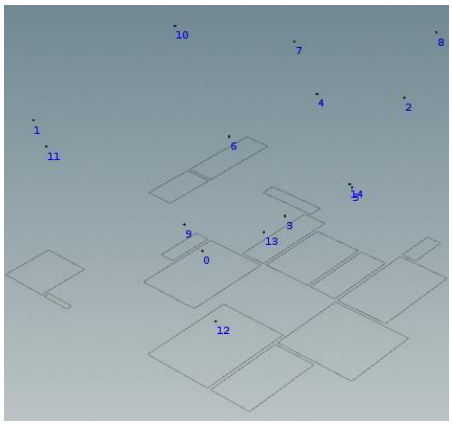


Gambar 4.15. Hasil Akhir Ruangan *Dungeon*

4.1.3. Implementasi Pembuatan Koridor

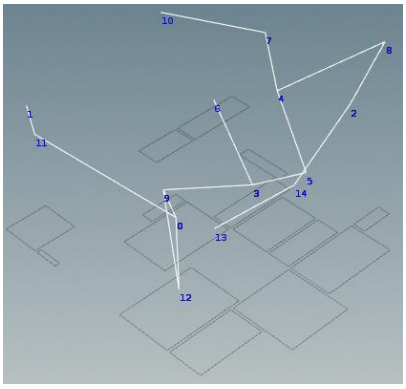
Proses selanjutnya berupa pembuatan koridor dimana kumpulan *nodes* pembuatan koridor terdapat pada Gambar 4.16. Pembuatan koridor dimulai terlebih dahulu dengan pengambilan setiap titik tengah ruangan seperti pada Gambar 4.17.

Gambar 4.16. *Nodes* Pembuatan Koridor



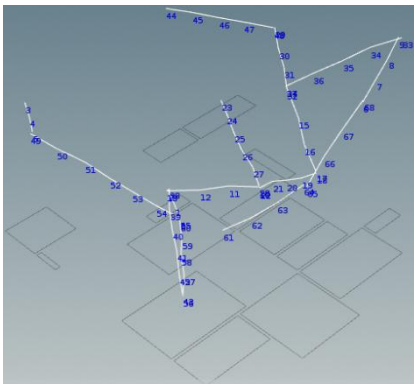
Gambar 4.17. Kumpulan Titik Tengah Ruangan

Posisi koordinat kumpulan titik tengah ruangan terlebih dahulu diinisiasi kedalam suatu himpunan / *array*. Setiap titik dihubungkan dengan titik lainnya dengan jarak terdekat. Setelah dilakukan penghubungan titik sebanyak jumlah titik tengah ruangan, maka dilanjutkan dengan penghubungan kumpulan titik yang tidak menyatu agar menjadi satu jalur yang utuh seperti pada Gambar 4.18.



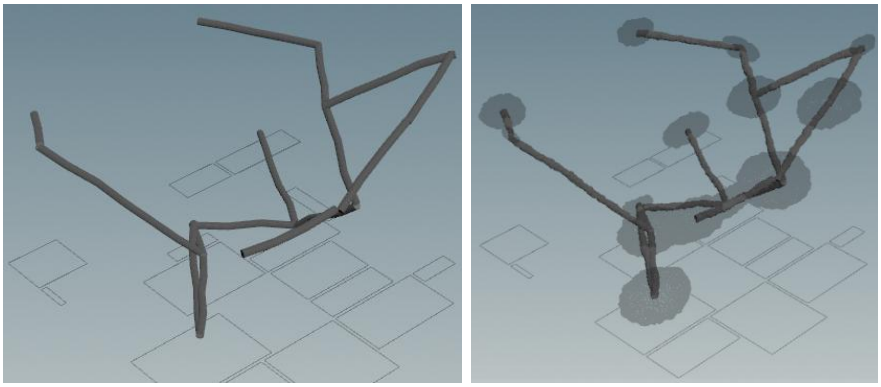
Gambar 4.18. Garis Koridor

Setelah garis koridor terbentuk, dilakukan proses *Resample* dan *Jitter* untuk menghasilkan garis yang lebih pendek dan tidak teratur seperti pada Gambar 4.19.



Gambar 4.19. *Resample* dan *Jitter*

Proses pembuatan koridor dilanjutkan dengan mengubah garis yang sudah tidak beraturan menjadi *polywire* dan diberikan tambahan *detail* terhadap setiap *primitive polywire* agar berbentuk seperti goa yang tidak beraturan bentuknya. Tampilan keseluruhan *polywire* sebelum diberikan *detail* dan setelah diberikan *detail* akan terlihat seperti pada Gambar 4.20



Gambar 4.20. *Polywire* tanpa *detail* (kiri) dan dengan *detail* (kanan)

4.2. Hasil Implementasi *Dungeon* dalam *Unity Engine*

Pengujian terhadap *3D Dungeon Generator* ini akan berfokus kepada 3*D Dungeon* yang dihasilkan. Pengujian dilakukan dengan membangkitkan sejumlah *3D Dungeon* pada *Unity* dimana akan terdapat sebuah *playable character* yang bisa digerakkan untuk menjelajahi bagian dalam dari *dungeon*. Pengujian ini dilakukan untuk menguji apakah *dungeon generator* yang dihasilkan sudah bekerja dengan semestinya sesuai dengan kebutuhan yang telah diidentifikasi sebelumnya. Hasil implementasi *dungeon* yang dihasilkan dibagi menjadi 2 bagian berupa hasil *dungeon* yang berisikan dungeon secara keseluruhan dan tampilan dalam dungeon.

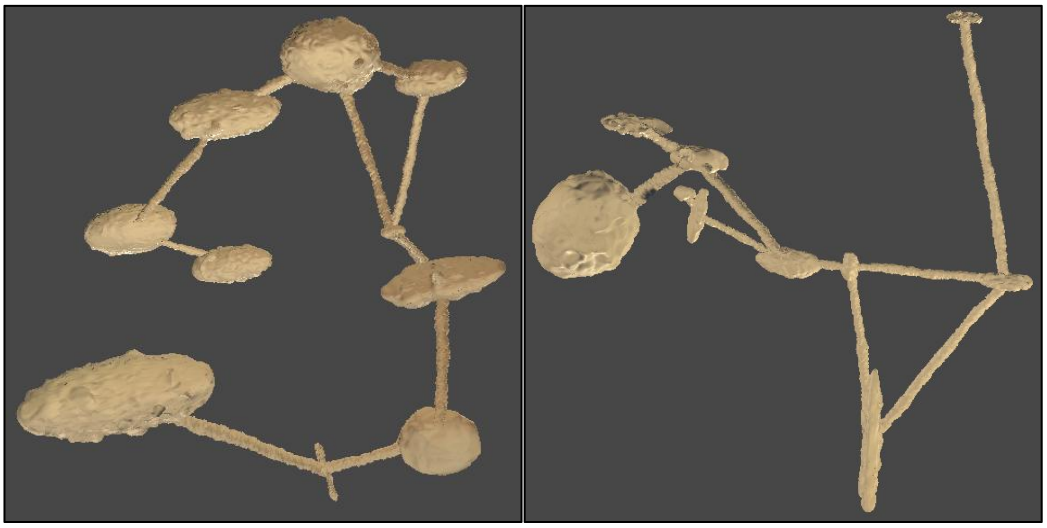
4.2.1. Hasil *Dungeon*

Pada bagian ini, tertera tampilan secara keseluruhan *dungeon* yang telah dihasilkan dengan parameter *level* 3, dengan jumlah ruangan mulai dari 5 hingga maksimum 30 ruangan, serta pengacak berupa *seeds*. Jumlah ruangan dari *dungeon* yang dihasilkan dan diimplementasikan berjumlah 5, 10, 15, 20, 25 dan 30 ruangan dengan nilai *seeds* pengacak yang berbeda-beda.

Tampilan *dungeon* dengan jumlah 5 ruangan dengan *seeds* pengacak 22598 serta *seeds* pengacak 61016 tertera pada Gambar 4.21. Pada bagian kiri, terlihat *dungeon* dengan *seeds* 22598 memiliki 3 ruangan yang berbentuk hampir sama dengan koridor yang memiliki jarak yang cukup berjauhan. Sementara itu, pada bagian kanan, *dungeon* dengan seeds 61016 memiliki ukuran ruangan yang bervariasi serta adanya koridor yang sangat dekat.

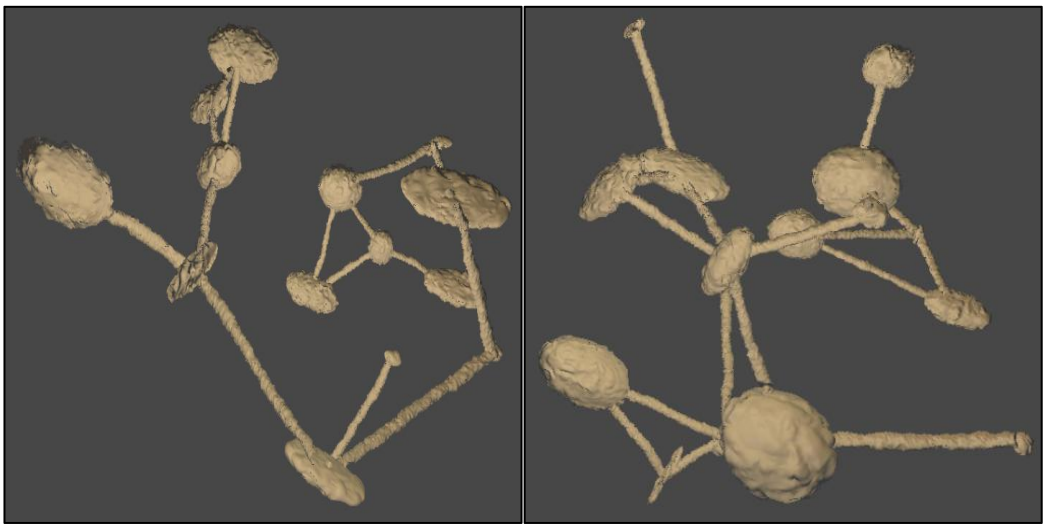
Gambar 4.21, *Dungeon* 5 ruangan dengan *seeds* 22598 (kiri) dan 61016 (kanan)

Dungeon dengan jumlah 10 ruangan dengan *seeds* pengacak 44067 serta *seeds* pengacak 74576 dapat dilihat pada Gambar 4.22. Pada bagian kiri, terlihat *dungeon* dengan *seeds* 44067 memiliki 2 ruangan yang sangat kecil dibandingkan dengan ruangan lainnya. Sementara itu, pada bagian kanan, *dungeon* dengan *seeds* 74576 memiliki ukuran ruangan yang cenderung kecil.



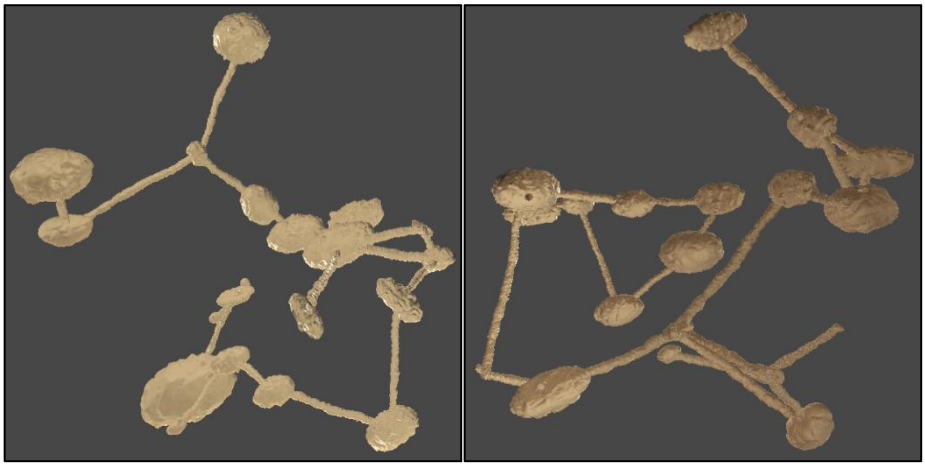
Gambar 4.22. *Dungeon* 10 ruangan dengan *seeds* 44067 (kiri) dan 74576 (kanan)

Tampilan *dungeon* berjumlah 15 ruangan dengan *seeds* pengacak 14689 serta *seeds* pengacak 27118 tertera pada Gambar 4.23. Pada bagian kiri, ruangan *dungeon* dengan *seeds* 14689 cenderung berbentuk pipih serta ada beberapa ruangan yang kecil. Di sisi lain, pada bagian kanan, ruangan *dungeon* dengan *seeds* 27118 cenderung berukuran besar dan adanya 1 ruangan yang terhubung langsung dengan 5 ruangan lainnya.



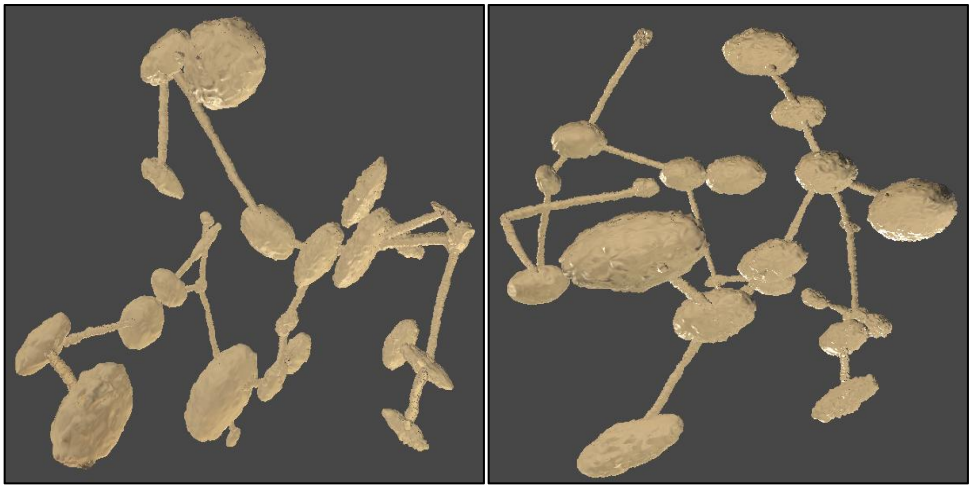
Gambar 4.23. *Dungeon* 15 ruangan dengan *seeds* 14689 (kiri) dan 27118 (kanan)

Tampilan *dungeon* berjumlah 20 ruangan dengan *seeds* pengacak 19774 serta *seeds* pengacak 74011 tertera pada Gambar 4.24. Pada bagian kiri, *dungeon* dengan *seeds* 19774 memiliki bentuk ruangan yang bervariasi dari kecil hingga besar serta koridor penghubung antar ruangan yang tidak terlalu bercabang. Sebaliknya, pada bagian kanan, *dungeon* dengan *seeds* 74011 memiliki ukuran ruangan yang hampir sama serta koridor yang cukup bercabang.



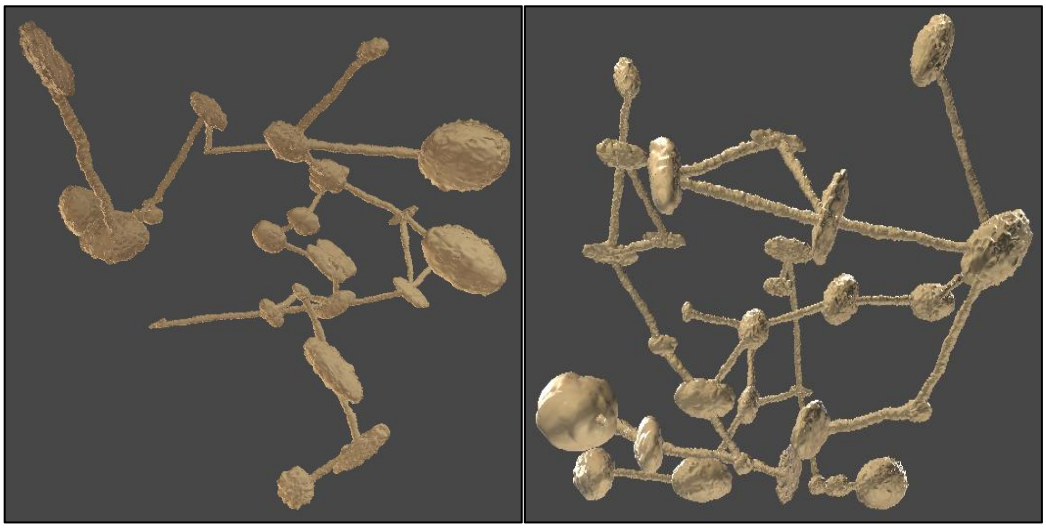
Gambar 4.24. *Dungeon* 20 ruangan dengan *seeds* 19774 (kiri) dan 74011 (kanan)

Dungeon berjumlah 25 ruangan dengan *seeds* pengacak 19774 dan 55367 dapat dilihat pada Gambar 4.25. Pada bagian kiri, *dungeon* dengan *seeds* 19774 terbentuk dari ruangan yang berukuran kecil hingga berukuran besar serta koridor yang tidak terlalu kompleks. Di sisi lain, pada bagian kanan, *dungeon* dengan *seeds* 55367 memiliki ukuran ruangan yang banyak berukuran sedang hingga besar serta dihubungkan dengan koridor yang cukup kompleks.



Gambar 4.25. *Dungeon* 25 ruangan dengan *seeds* 19774 (kiri) dan 55367 (kanan)

Tampilan *dungeon* dengan jumlah 30 ruangan serta *seeds* pengacak 22033 dan 40677 dapat dilihat pada Gambar 4.26. Pada bagian kiri terlihat *dungeon* dengan *seeds* 22033 memiliki banyak ruangan yang ujungnya tertutup dan lebih terfokus pada bagian tengah *dungeon*. Di sisi lain, pada bagian kanan, *dungeon* dengan *seeds* 40677 terbentuk dari kumpulan ruangan yang tersebar di segala arah dan terhubung satu sama lainnya seperti jaring sehingga kompleksitas *dungeon* semakin meningkat.



Gambar 4.26. *Dungeon* 30 ruangan dengan *seeds* 22033 (kiri) dan 40677 (kanan)

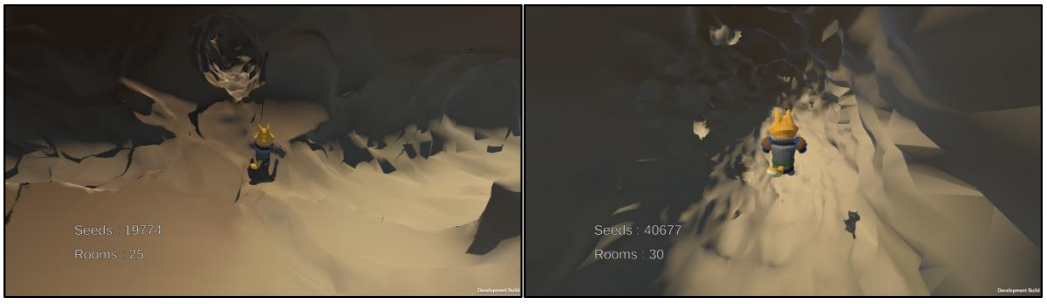
4.2.2. Tampilan Dalam *Dungeon*

Dari sejumlah *dungeon* yang telah dihasilkan dengan jumlah ruangan 3 ruangan hingga 30 ruangan, dilakukan pemilihan beberapa tampilan bagian dalam *dungeon* dengan *seeds* yang berbeda-beda yang terbagi menjadi tampilan bagian dalam ruangan serta koridor penghubung antar ruangan untuk mendapatkan gambaran lebih lanjut tentang *dungeon* yang dihasilkan.

Tampilan bagian dalam *dungeon* dengan jumlah 5 ruangan dan 10 ruangan tertera pada Gambar 4.27. Pada bagian kiri, salah satu tampilan dari ruangan dengan *seeds* 22598 berbentuk cenderung pipih dengan adanya 1 koridor di sebelah kiri ruangan. Di sisi lain, salah satu ruangan dengan *seeds* 44067 memiliki ukuran yang besar serta adanya 2 koridor yang berada di bagian depan dan bagian bawah ruangan.

Gambar 4.27. Tampilan *dungeon* dengan jumlah ruangan 5 (kiri) dan 10 (kanan)

Tampilan bagian dalam *dungeon* dengan jumlah 25 ruangan dan 30 ruangan tertera pada Gambar 4.28. Pada bagian kiri, salah satu tampilan dari ruangan *dungeon* dengan *seeds* 19774 berbentuk oval dengan kondisi ruangan memiliki sedikit tumpukan tanah dengan adanya 1 koridor yang berada tepat di bagian depan ruangan. Sedangkan, pada bagian kanan, salah satu ruangan dari *dungeon* dengan *seeds* 40677 memiliki ukuran yang memanjang ke depan serta terdiri atas 2 koridor yang berdekatan terfokus pada sebelah kiri ruangan.



Gambar 4.28. Tampilan dungeon dengan jumlah ruangan 25 (kiri) dan 30 (kanan)

Selanjutnya, salah satu tampilan koridor *dungeon* dari *dungeon* dengan jumlah ruangan 5 dan 15 ruangan tertera pada Gambar 4.29. Koridor yang tertera pada Gambar 4.29 memiliki tekstur yang kasar seperti pada ruangan *dungeon* yang dihubungkan oleh koridor tersebut.

Gambar 4.29. Tampilan koridor dengan jumlah ruangan 5 (kiri) dan 15 (kanan)

Tampilan koridor *dungeon* dengan jumlah ruangan 25 dan 30 tertera pada Gambar 4.30. Pada sisi kiri, salah satu koridor dari *dungeon* dengan *seeds* 19774 terlihat memiliki jarak yang sangat jauh serta tinggi ke atas yang mengakibatkan lebih susah dalam melakukan explorasi ruangan. Di sisi lain, salah satu koridor *dungeon* dengan *seeds* 40677 cenderung mendatar dan bertekstur halus.

Gambar 4.30. Tampilan koridor dengan jumlah ruangan 25 (kiri) dan 30 (kanan)

4.3. Hasil Pengujian

Pada bagian ini berisi hasil pengujian terhadap Dungeon Generator yang terbagi atas pengujian dalam membangkitkan ruangan dungeon, pengujian atas pembangkit koridor dungeon.

4.3.1. Pengujian Pembangkit Ruangan Dungeon

Pengujian terhadap pembangkit ruangan *dungeon* terbagi atas keberhasilan dalam melakukan partisi area ruangan, keberhasilan dalam membentuk jumlah ruangan sesuai dengan parameter *Room Qty* serta memberikan detail pada ruangan yang sudah dibentuk. Keberhasilan pengujian pembangkit ruangan *dungeon* tertera pada Tabel 4.1.

Tabel 4.1. Pengujian Pembangkit Ruangan *Dungeon*

Kelas Uji	Skenario Uji	Hasil yang diharapkan	Kesimpulan
Pembangkit Ruangan *Dungeon*	Membangkitkan Ruangan *Dungeon*	Membagi area ruangan	[✓] Berhasil [] Tidak
		Jumlah ruangan sesuai parameter *Room Qty*	
		Memberi detail ruangan	

4.3.2. Pengujian Pembangkit Koridor *Dungeon*

Pengujian terhadap pembangkit koridor *dungeon* terbagi atas keberhasilan dalam menghubungkan semua ruangan menjadi satu *dungeon*, keberhasilan dalam membentuk koridor dengan ukuran yang sama dan tidak tumpang tindih, serta memberikan detail pada koridor yang sudah dibangkitkan. Keberhasilan pengujian pembangkit koridor *dungeon* tertera pada Tabel 4.2.

Tabel 4.2. Pengujian Pembangkit Koridor *Dungeon*

Kelas Uji	Skenario Uji	Hasil yang diharapkan	Kesimpulan
Pembangkit Koridor *Dungeon*	Membangkitkan Koridor *Dungeon*	Semua ruangan terhubung menjadi satu *dungeon*	[✓] Berhasil [] Tidak
		Koridor yang terbentuk memiliki ukuran yang sama	
		Koridor yang dibangkitkan tidak tumpang tindih	
		Memberi detail koridor	

4.3.3. Performa

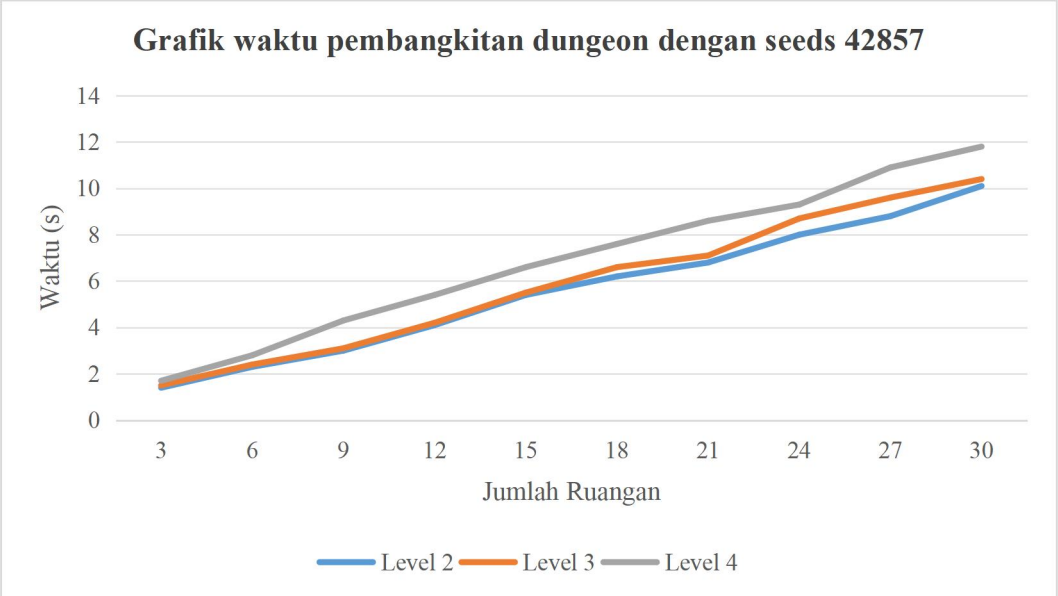
Pengujian performa dilakukan untuk menentukan seberapa efisien algoritma dalam membangkitkan *dungeon* dengan konfigurasi yang berbeda. Pengujian performa pembangkitan *dungeon* dilakukan berdasarkan dua aspek berupa waktu pembangkitan *dungeon* dan kinerja grafis. Pada aspek pertama, pengujian dilakukan untuk mengukur waktu yang dibutuhkan dalam membangkitkan *dungeon* dengan variasi jumlah ruangan (3, 6, 9, 12, 15, 18, 21, 24, 27, dan 30), ketinggian (level 2, 3, dan 4) serta *seeds* tertentu yaitu seed 42857 dan 80952.

Data pengukuran waktu terhadap pembangkitan *dungeon* dengan *seeds* 42857 tertera pada Tabel 4.3.

Tabel 4.3. Data waktu pembangkit *dungeon* dengan *seeds* 42857

Level	**Jumlah Ruangan**									
	3	6	9	12	15	18	21	24	27	30
2	1.4	2.3	3.0	4.1	5.4	6.2	6.8	8.0	8.8	10.1
3	1.5	2.4	3.1	4.2	5.5	6.6	7.1	8.7	9.6	10.4
4	1.7	2.8	4.3	5.4	6.6	7.6	8.6	9.3	10.9	11.8

Data pada Tabel 4.3 divisualisasikan dalam bentuk grafik pada Gambar 4.31. Grafik pada Gambar 4.31 menunjukkan bahwa waktu pembangkitan *dungeon* meningkat ketika jumlah ruangan bertambah. Selain itu, terjadi pertambahan waktu yang cukup berbeda ketika tingkat ketinggiannya berubah dari *level* 3 ke *level* 4. Namun, waktu pembangkitan *dungeon* antara *dungeon level* 2 dengan *level* 3 hanya memiliki perbedaan saat *dungeon* menghasilkan ruangan berjumlah 24 dan 27.



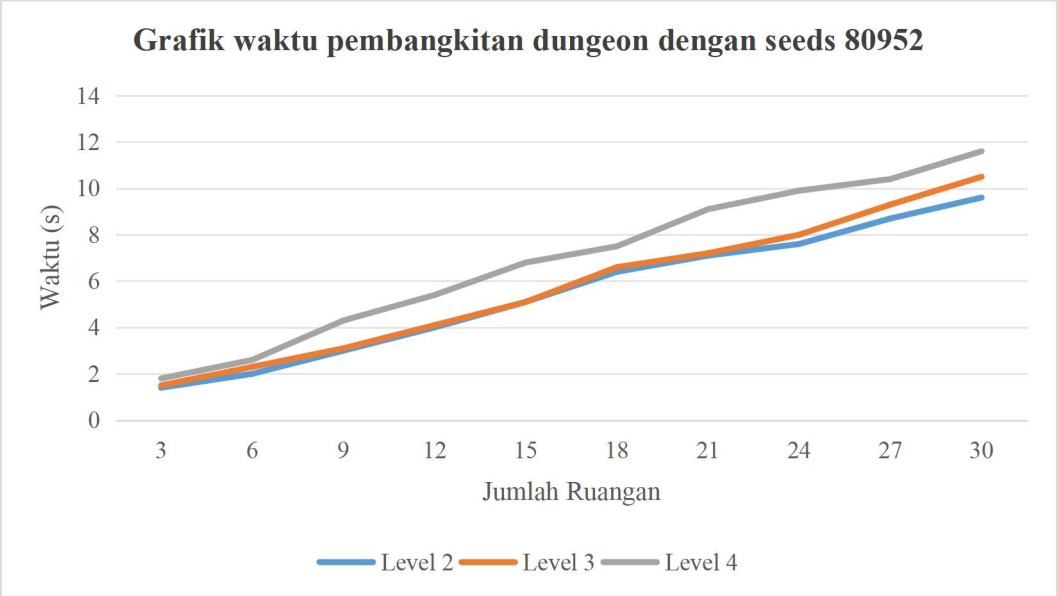
Gambar 4.31. Waktu Pembangkitan *Dungeon* untuk Jumlah Ruangan dan *Level* dengan *seeds* 42857

Data pengukuran waktu terhadap pembangkitan *dungeon* dengan *seeds* 80952 tertera pada Tabel 4.4.

Tabel 4.4. Data waktu pembangkit *dungeon* dengan *seeds* 80952

Level	Jumlah Ruangan									
	3	6	9	12	15	18	21	24	27	30
2	1.4	2.0	3.0	4.0	5.1	6.4	7.1	7.6	8.7	9.6
3	1.5	2.3	3.1	4.1	5.1	6.6	7.2	8.0	9.3	10.5
4	1.8	2.6	4.3	5.4	6.8	7.5	9.1	9.9	10.4	11.6

Data pada Tabel 4.4 divisualisasikan dalam bentuk grafik pada Gambar 4.32. Grafik pada Gambar 4.32 menunjukkan bahwa waktu pembangkitan *dungeon* meningkat ketika adanya pertambahan jumlah ruangan. Selain itu, terjadi pertambahan waktu yang cukup berbeda ketika tingkat ketinggiannya berubah dari *level* 3 ke *level* 4. Pada *level* 2 dan *level* 3, dapat dilihat bahwa waktu pembangkitan dungeon antara *level* 2 dan *level* 3 hanya memiliki perbedaan pada saat *dungeon* menghasilkan 27 dan 30 ruangan.



Gambar 4.32. Waktu Pembangkitan *Dungeon* untuk Jumlah Ruangan dan *Level* dengan *seeds* 80952

Aspek kedua melibatkan pengukuran kinerja grafis, yaitu jumlah *triangles* yang dihasilkan dan rata-rata *frame per second* (FPS) saat dungeon dengan variasi jumlah ruangan (5, 10, 15, 20, 25, dan 30) serta seeds berbeda divisualisasikan dengan dua skenario berupa dengan *shadow* dan tanpa *shadow*. Jumlah Triangles yang didapatkan berdasarkan jumlah ruangan serta *seeds* ruangan yang berbeda tertera pada Tabel 4.5.

Tabel 4.5. Jumlah *Triangles*

Room Qty	Seeds	Jumlah Tris	
		Shadow On	Shadow Off
5	22598	604.3k	217.8k
5	61016	467.0k	172.0k
10	44067	872.7k	307.2k
10	74576	805.2k	284.7k
15	14689	1.2m	422.7k
15	27118	1.2m	418.6k

Room Qty	Seeds	Jumlah Tris	
		Shadow On	Shadow Off
20	19774	1.4m	487.4k
20	74011	1.6m	533.8k
25	19774	1.7m	572.4k
25	55367	1.7m	595.8k
30	22033	2.1m	708.4k
30	40677	2.2m	754.9k

Selanjutnya, rata-rata *frame per second* (FPS) yang didapatkan berdasarkan jumlah ruangan dan *seeds* yang berbeda tertera pada Tabel 4.6.

Tabel 4.6. Jumlah FPS

Room Qty	Seeds	FPS	
		Shadow On	Shadow Off
5	22598	123	161
5	61016	127	163
10	44067	111	153
10	74576	110	152
15	14689	93	140
15	27118	93	138

Room Qty	Seeds	FPS	
		Shadow On	Shadow Off
20	19774	84	126
20	74011	82	127
25	19774	78	124
25	55367	79	122
30	22033	69	114
30	40677	66	109

BAB 5

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Kesimpulan yang dapat diambil dari penelitian ini adalah sebagai berikut:

1. Penggunaan algoritma *K-D Tree* dalam membagi ruang 2 dimensi menjadi partisi-partisi menghasilkan struktur partisi yang seimbang.
2. *Dungeon generator* berhasil menghasilkan *dungeon* dengan jumlah ruangan 3 hingga 30 ruangan dengan *level* 2 hingga 4 serta dengan *seeds* yang berbeda-beda.
3. Waktu pembangkitan *dungeon* oleh *Dungeon Generator* dipengaruhi oleh tingkat ketinggian *level*, jumlah ruangan, *seeds* dan tingkat kompleksitas *dungeon* yang dihasilkan yang dapat dilihat pada Tabel 4.3 dan Tabel 4.4.
4. Berdasarkan ada tidaknya *shadow*, jumlah triangles dari dungeon dengan *shadow* dua kali lebih banyak daripada dungeon tanpa *shadow*.
5. Jumlah *triangles* dan *shadow* berpengaruh terhadap tingkat FPS, *dungeon* dengan *shadow* memiliki FPS dengan rata-rata 40-45 FPS lebih rendah dibandingkan dengan dungeon tanpa *shadow*.

5.2. Saran

Beberapa saran yang diberikan untuk penelitian selanjutnya adalah sebagai berikut:

1. Diharapkan pada penelitian selanjutnya dapat menggunakan titik selain titik tengah ruangan sebagai titik pivot untuk perhitungan jarak terdekat dan pembentukan koridor.
2. Diharapkan pada penelitian selanjutnya dapat menggunakan metode partisi lainnya dalam membentuk dungeon.

DAFTAR PUSTAKA

Alvarez, A., Dahlskog, S., Font, J. and Togelius, J., 2019, August. Empowering quality diversity in dungeon design with interactive constrained map-elites. In *2019 IEEE Conference on Games (CoG)* (pp. 1-8). IEEE.

Baldwin, A., Dahlskog, S., Font, J.M. and Holmberg, J., 2017, August. Towards pattern-based mixed-initiative dungeon generation. In *Proceedings of the 12th International Conference on the Foundations of Digital Games* (pp. 1-10).

Baron, J.R., 2017, April. Procedural dungeon generation analysis and adaptation. In *Proceedings of the SouthEast Conference* (pp. 168-171).

Dahlskog, S., Björk, S. and Togelius, J., 2015. Patterns, dungeons and generators. In *Foundations of Digital Games Conference, FDG, Pacific Grove, USA (2015)*. Foundations of Digital Games.

De Carli, D.M., Bevilacqua, F., Pozzer, C.T. and d'Ornellas, M.C., 2011, November. A survey of procedural content generation techniques suitable to game development. In *2011 Brazilian symposium on games and digital entertainment* (pp. 26-35). IEEE.

Gill, S. and Hooda, M., 2021. The Design Perspective of the Structures Based on kd Tree. In *Rising Threats in Expert Applications and Solutions: Proceedings of FICR-TEAS 2020* (pp. 515-524). Springer Singapore.

Hamilton, W., 2019. *Procedural Generation of Three-Dimensional Game Levels with Interior Architecture* (Doctoral dissertation, Carleton University).

Liu, J., Snodgrass, S., Khalifa, A., Risi, S., Yannakakis, G.N. and Togelius, J., 2021. Deep learning for procedural content generation. *Neural Computing and Applications*, *33*(1), pp.19-37.

Merrell, P. and Manocha, D., 2010. Model synthesis: A general procedural modeling algorithm. *IEEE transactions on visualization and computer graphics*, *17*(6), pp.715-728.

Otair, D.M., 2013. Approximate k-nearest neighbour based spatial clustering using kd tree. *arXiv preprint arXiv:1303.1951*.

Putra, P.A., Tarigan, J.T. and Zamzami, E.M., 2023, October. Procedural 2D Dungeon Generation Using Binary Space Partition Algorithm And L-Systems. In *2023*

International Conference on Computer, Control, Informatics and its Applications (IC3INA) (pp. 365-369). IEEE.

Shen, Z., 2022. Procedural Generation in Games: Focusing on Dungeons. In *SHS Web of Conferences* (Vol. 144, p. 02005). EDP Sciences.

Ullrich, T., Schinko, C. and Fellner, D.W., 2010. Procedural modeling in theory and practice.

Viana, B.M. and dos Santos, S.R., 2019, October. A survey of procedural dungeon generation. In *2019 18th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)* (pp. 29-38). IEEE.

Viana, B.M. and dos Santos, S.R., 2021. Procedural dungeon generation: A survey. *Journal on Interactive Systems*, *12*(1), pp.83-101.

Weeks, M. and Davis, J., 2022, April. Procedural dungeon generation for a 2D top-down game. In *Proceedings of the 2022 ACM Southeast Conference* (pp. 60-66).

Dog Knight PBR Polyart. 2020. Versi 1.2. Dungeon Mason. Diambil dari https://assetstore.unity.com/packages/3d/characters/animals/dog-knight-pbr-polyart-135227

Rock Textures - 4K. 2020. Versi 1.0. Texture Haven. Diambil dari https://assetstore.unity.com/packages/2d/textures-materials/rock-textures-4k-179128