Jurnal Informatika: Jurnal pengembangan IT

Vol. 10, No. 4, 2025
ISSN : 2477-5126, DOI:10.30591/jpit.v10i4.8629

Implementasi *Binary Space Partitioning* Untuk *Procedural Map Generation* Dalam Gim

Harits Ar Rosyid [1], Ahmad Adi Prasetyo[2]

[1,2] Departemen Teknik Elektro dan Informatika, Fakultas Teknik, Universitas Negeri Malang, Jl. Semarang no. 5, Malang, 65145, Indonesia

Info Artikel

Riwayat Artikel:

Received 2025-03-30
Revised 2025-08-24
Accepted 2025-08-25

Corresponding Author:

Harits Ar Rosyid

Email: harits.ar.ft@um.ac.id

Abstract *– The popularity of games as an interactive entertainment medium continues to grow, with 2D maps playing a crucial role in enhancing user experiences. However, manual map creation is time-consuming as game worlds become increasingly complex. Procedural Content Generation (PCG) offers a solution by automating map creation, increasing replayability while reducing the designer's workload. This research explores the use of the Binary Space Partitioning (BSP) algorithm to procedurally generate dungeon maps with random connections between rooms, resulting in more exploratory and dynamic layouts. The research process included the development of a map generator, the implementation of BSP with random connections, and validation using Space Syntax analysis through Visibility Graph Analysis (VGA) and Axial Line Analysis to measure connectivity, visibility, and integration. The results show that a small minimum room size configuration, such as in test set 1 with a map size of 100x100 and a minimum room size of 15x15, results in more hotspots despite lower connectivity and visibility, while a larger room configuration actually increases connectivity and visibility but reduces the number of hotspots. Thus, this study technically proves the effectiveness of BSP in varying game maps through Space Syntax-based quantitative evaluation and confirms its potential in producing varied maps with measurable spatial characteristics.*

Keywords: *Binary Space Partitioning; Game; Game Maps; Procedural Content Generation; Space Syntax Analysis*

Abstrak *– Popularitas gim sebagai media hiburan interaktif terus berkembang, dengan peta 2D yang berperan penting dalam meningkatkan pengalaman pengguna. Namun, pembuatan peta secara manual membutuhkan waktu yang lama seiring meningkatnya kompleksitas dunia gim. Procedural Content Generation (PCG) menawarkan solusi dengan mengotomasi pembuatan peta, sehingga meningkatkan replayability sekaligus mengurangi beban kerja desainer. Penelitian ini mengeksplorasi penggunaan algoritma Binary Space Partitioning (BSP) untuk menghasilkan peta dungeon secara prosedural dengan koneksi acak antar ruangan, sehingga tata letak yang dihasilkan menjadi lebih eksploratif dan dinamis. Proses penelitian mencakup pengembangan generator peta, implementasi BSP dengan penghubung acak, serta validasi menggunakan analisis Space Syntax melalui Visibility Graph Analysis (VGA) dan Axial Line Analysis untuk mengukur konektivitas, visibilitas, dan integrasi. Hasilnya menunjukkan bahwa konfigurasi ukuran minimum ruangan yang kecil, seperti pada test set 1 dengan ukuran peta 100×100 dan ruangan minimum 15×15, menghasilkan lebih banyak hot spot meskipun dengan konektivitas dan visibilitas yang lebih rendah, sementara konfigurasi ruangan yang lebih besar justru meningkatkan konektivitas dan visibilitas namun mengurangi jumlah hot spot. Dengan demikian, penelitian ini secara teknis membuktikan efektivitas BSP dalam memvariasikan peta permainan melalui evaluasi kuantitatif berbasis Space Syntax serta menegaskan potensinya dalam menghasilkan peta yang bervariasi dengan karakter spasial yang terukur.*

Kata Kunci*: Binary Space Partitioning, Game, Game Maps, Procedural Content Generation, Space Syntax Analysis*

I. PENDAHULUAN

Gim adalah salah satu media hiburan interaktif saat ini dan popularitasnya berkembang pesat. Salah satu elemen penting dalam gim yang meningkatkan pengalaman pengguna adalah penerapan peta 2 dimensi. Salah satunya adalah pada gim *rogue-like* yang mengawali sejarah peta gim dimana pemain bermain di dalam peta itu sendiri dan menggerakkan karakter di dalam gim tersebut [1]. Selain itu, peta gim juga berkontribusi pada persepsi pemain bahwa ia berada di dunia virtual dengan tambahan konteks visual dari lingkungan sekitar [2]. Hal ini menunjukkan bahwa peta merupakan bagian penting dari pengalaman pemain dalam sebuah gim.

Pembuatan peta dalam industri gim membutuhkan proses manual yang panjang, dan seiring dengan perkembangan teknologi yang terus membuat dunia gim menjadi semakin kompleks, para desainer gim membutuhkan lebih banyak waktu untuk membuat konten [3], [4]. Hal ini menunjukkan bahwa desain peta gim secara manual sulit untuk beradaptasi dengan kebutuhan yang semakin kompleks dan membutuhkan waktu yang tidak sedikit. Proses manual tidak hanya memakan waktu dan sumber daya yang banyak, tetapi juga sulit untuk menjamin kualitas yang seragam pada konten yang dibuat oleh desainer [5]. Level yang dibuat secara manual

bisa jadi seragam dan kurang bervariasi, sehingga berpotensi mempengaruhi pengalaman bermain. Oleh karena itu, dibutuhkan sebuah cara untuk menghasilkan peta yang lebih dinamis dan bervariasi dengan biaya yang lebih rendah.

Procedural Content Generation (PCG) adalah penerapan komputer untuk menghasilkan konten gim dan mampu memilih konten gim yang menarik bagi pemain [6]. Konten gim akan dihasilkan secara otomatis dengan menerapkan algoritma dan menghasilkan konten yang bervariasi seperti tekstur, medan, bahkan alur cerita dan karakter [7]. Konten yang dihasilkan secara prosedural juga membuat gim berumur lebih panjang karena pemain dapat terus menjelajahi peta dan tantangan baru setiap kali mereka bermain [8]. Hal ini menunjukkan bahwa PCG dapat meningkatkan *replayability* dari sebuah gim tanpa menambah beban kerja desainer untuk membuat setiap level secara manual. Selain itu, *Procedural Content Generation* (PCG) merupakan cara alternatif untuk membuat dunia gim yang kompleks dalam waktu yang terbatas tanpa membebani desainer gim [6]. Di lain sisi, PCG juga bisa memberi inspirasi lebih dan memperluas imajinasi seorang desainer [9]. Keuntungan lainnya adalah dapat mengurangi biaya pembuatan yang diperlukan untuk mengembangkan gim [10]. Oleh karena itu, PCG juga dapat menjadi sebuah alat eksplorasi ide-ide baru dan inovatif untuk membantu desainer dalam menciptakan konten mereka sendiri untuk gim tertentu [11]. Hal ini tidak hanya meningkatkan efisiensi dalam proses pengembangan, tetapi juga membuka peluang baru dalam penciptaan konten yang lebih dinamis.

Salah satu penerapan PCG dalam gim adalah pembuatan peta. Ada beberapa metode untuk menghasilkan peta secara prosedural. Ramadhan & Indriyanti memanfaatkan *Perlin Noise* untuk mendesain peta gim *platformer* berbentuk gua, namun desainnya dipengaruhi oleh ukuran dunia, ukuran karakter, dan ukuran kamera, sehingga kurang sesuai untuk peta *dungeon* dengan struktur ruangan dan koridor [12]. Ashlock menggunakan *Fashion-based Cellular Automata*, pengembangan dari *Cellular Automata* (CA), untuk membuat level berbentuk gua karena mampu menirukan pola-pola yang kompleks di alam [13]. Hidayat juga menerapkan CA dengan aturan *Von Neumann Neighborhood* untuk menghasilkan level *dungeon* [14]. Namun, CA memiliki kelemahan yaitu sulit untuk dipahami secara penuh karena setiap parameter mempengaruhi banyak aspek dari peta, sehingga kurang cocok untuk membuat peta dengan kebutuhan yang spesifik [15]. Ijai menggunakan algoritma *Random Walk* atau *Drunkard's Walk* untuk membuat level *dungeon* dengan jalur acak yang menyerupai lintasan molekul dalam cairan atau gas, sehingga menghasilkan struktur yang tidak beraturan dan dinamis [16]. Namun, algoritma ini lebih cocok untuk peta berbentuk gua daripada peta dengan struktur ruangan dan koridor [17]. *Cellular Automata* dan *Random Walk* dapat menghasilkan ruangan dan koridor yang berbentuk acak, tetapi tidak memiliki kontrol penuh terhadap ukuran dan bentuknya, sehingga dibutuhkan algoritma yang lebih terstruktur dan terkendali.

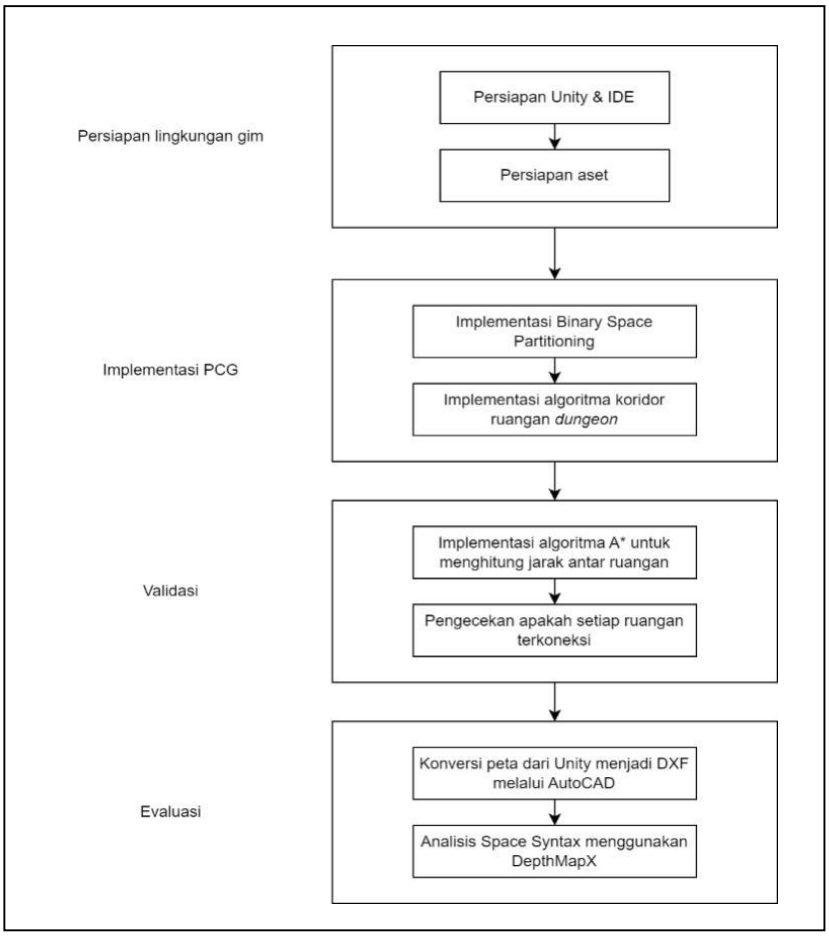
Binary Space Partitioning (BSP) adalah algoritma yang secara iteratif membagi ruang hingga mencapai kondisi akhir, memastikan ruangan tidak saling tumpang tindih dan terhubung dengan ruang induknya [18], [19]. Snodgrass & Sarkar menggabungkan *Example–Driven Binary Space Partitioning* dengan *Variational Autoencoders* [20] dan menunjukkan bahwa kombinasi struktur berbasis BSP dengan pembelajaran generatif mampu menghasilkan level yang seimbang antara integritas dan variasi. Pendekatan ini relevan dengan *procedural map generation* pada *dungeon crawler*, di mana struktur dasar ruangan dan koridor dibentuk menggunakan BSP. Algoritma ini mudah dikontrol dengan parameter seperti ukuran ruang minimum dan area peta, menghasilkan ruang-ruang terpisah yang dihubungkan oleh koridor. Dalam konteks ini, agar peta yang dihasilkan lebih eksploratif, algoritma penghubung antar ruangan pada BSP juga perlu diperbarui sehingga ruangan tidak hanya terhubung dengan ruangan terdekat, tetapi juga dapat terhubung secara acak dengan ruangan yang lebih jauh untuk menciptakan lebih banyak percabangan dan jalur alternatif.

Pembuatan peta permainan secara manual masih menyita banyak waktu dan tenaga. Metode generatif yang umum digunakan, seperti *Perlin Noise*, *Cellular Automata*, dan *Random Walk*, cenderung menghasilkan bentuk organik atau pola acak, tetapi kurang efektif dalam membangun ruangan yang saling terhubung melalui koridor. Oleh karena itu, penelitian ini bertujuan untuk mengeksplorasi penerapan algoritma *Binary Space Partitioning* dengan mekanisme penghubung ruangan acak guna menghasilkan tata letak peta 2D yang menghadirkan tata letak peta yang terstruktur, namun tetap memungkinkan eksplorasi yang lebih luas dan bervariasi. Untuk mengevaluasi kualitas peta yang dihasilkan, penelitian ini menerapkan analisis *Space Syntax*, khususnya *Visibility Graph Analysis* (VGA) dan *Axial Line Analysis*, guna mengukur aspek konektivitas, visibilitas, dan integrasi spasial.

II. METODE

A. Procedural Map Generation yang Diusulkan

Penelitian ini menerapkan algoritma *Binary Space Partitioning* dengan koridor ruangan acak untuk menghasilkan peta *dungeon*. Gambar 1 adalah sistem *map generator* yang diusulkan. Sistem ini dibagi menjadi beberapa blok yaitu persiapan lingkungan gim, implementasi peta *dungeon*, validasi, dan evaluasi.



Gambar 1. Blok Diagram *Procedural Map Generation*

Blok pertama dalam persiapan lingkungan permainan adalah menyiapkan dasar pengembangan untuk *generator* peta *dungeon*. Langkah awal yang dilakukan adalah memasang Unity sebagai *game engine* yang akan digunakan untuk pengembangan, serta Visual Studio Code sebagai *integrated development environment* (IDE) untuk menulis kode pemrograman sistem *generator* peta. Selain itu, tahap ini juga mengkonfigurasi *tilemap* yang akan digunakan untuk memvisualisasikan peta *dungeon* yang telah dibuat. *Tilemap* ini berperan penting sebagai representasi grafis dari ruangan, koridor, dan elemen-elemen lain di dalam peta *dungeon*.

Blok kedua adalah proses pembuatan peta *dungeon*, yang meliputi implementasi *Binary Space Partitioning* (BSP) dan algoritma untuk menghubungkan ruangan secara acak. Implementasi BSP adalah langkah pertama dalam pembuatan peta *dungeon*.

Beberapa variabel penting yang digunakan dalam pembangkitan peta antara lain panjang dan lebar peta, panjang dan lebar ruangan minimum, *seed*, dan ukuran ruangan. Panjang dan lebar peta berfungsi sebagai batas ruang keseluruhan yang akan dibagi, sedangkan panjang dan lebar minimum ruangan berperan sebagai kontrol untuk mencegah ruangan berukuran terlalu kecil. Parameter *seed* menentukan konsistensi hasil *generation*, sehingga peta yang sama dapat direplikasi pada percobaan berbeda. Namun, untuk variabel *seed* dan ukuran ruangan, mereka akan ditetapkan sebagai nilai konstan. Ukuran ruangan pada penelitian ini ditetapkan sebagai nilai konstan untuk menjaga konsistensi. Hasil dari tahap ini adalah terbentuknya ruangan-ruangan yang masih belum terhubung satu sama lain.

TABEL 1
ALGORITMA *BINARY SPACE PARTITIONING*

Algoritma *Binary Space Partitioning*	
1:	**Require**: $Level$ → sebuah *grid* dengan tinggi dan lebar h×w
2:	**Require**: min → ukuran minimum sebuah *section*
3:	Bagi *section* tersebut sepanjang garis vertikal atau horizontal
4:	Pilih salah satu dari dua *section* yang telah dibuat, *s*
5:	**if** $s_{height} \geq 2 \times min$ or $s_{width} \geq 2 \times min$ **then**
6:	Pergi ke baris 3 dengan *s*
7:	**else**
8:	Pilih *section* lain dan pergi ke baris 5
9:	**end if**
10:	**for all** s **do**
11:	Isi *section s*
12:	**end for**

Proses kedua adalah menghubungkan ruangan-ruangan menggunakan koridor. Penelitian sebelumnya oleh Biyik & Sürer [19] menghubungkan dua ruangan yang memiliki jarak paling dekat dibandingkan dengan kombinasi ruangan lainnya. Namun, pada penelitian ini, ruangan-ruangan tersebut akan dihubungkan secara acak, sehingga memberikan kesempatan bagi ruangan tertentu untuk memiliki beberapa cabang koridor yang menghubungkan ke satu ruangan.

TABEL 2
ALGORITMA KORIDOR RUANGAN

Algoritma Koridor Ruangan	
1:	**Require**: $Rooms$ ← daftar titik pusat ruangan (x, y)
2:	**Require**: $current\ room$ ← pilih ruangan awal secara acak
3:	Hapus $current\ room$ dari daftar $Rooms$
4:	**while** $banyak\ rooms > 0$
5:	Pilih ruangan secara acak dari $Rooms$, $choosen\ room$
6:	Hapus $choosen\ room$ dari $Rooms$
7:	Hubungkan $current\ room$ dan $choosen\ room$ dengan sebuah koridor
8:	Ubah $current\ room$ menjadi $choosen\ room$

Pendekatan ini diharapkan dapat menghasilkan variasi yang lebih dinamis dalam struktur *dungeon* yang dihasilkan. Selain itu, kondisi tumpang tindih antara koridor yang ada dan ruangan baru dengan koridor yang sedang dibuat akan diabaikan dalam proses ini, sehingga memungkinkan peta yang lebih kompleks dan beragam.

Blok ketiga dari pembuatan peta adalah validasi. Hal ini dilakukan untuk memastikan bahwa peta yang dihasilkan dapat dinavigasikan, dengan catatan bahwa setiap ruangan memiliki jalur yang dapat dilalui untuk berpindah dari satu ruangan ke ruangan lainnya. Untuk melakukan validasi ini, algoritma A* digunakan untuk menemukan rute antara setiap kombinasi ruangan yang berbeda. Algoritma ini dipilih karena menggunakan pendekatan *best-first heuristic search*, di mana sebuah fungsi f digunakan untuk menentukan *node* mana yang akan diperiksa selanjutnya, sehingga mempercepat proses pencarian rute terbaik [21]. Jumlah kombinasi ruangan yang unik dapat dihitung dengan persamaan berikut:

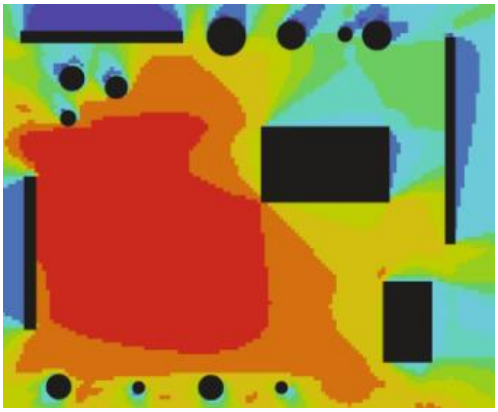
$$Total\ kombinasi\ ruangan = \frac{banyak\ ruangan \times (banyak\ ruangan - 1)}{2} \quad (1)$$

Dari setiap kombinasi, akan ditemukan rute terdekat antara dua ruangan pada peta yang telah dibuat. Jika jarak antara dua ruangan lebih besar dari nol, maka dapat dikatakan bahwa ada jalan yang dapat dilalui antara kedua ruangan tersebut. Sebaliknya, jika jaraknya nol, maka tidak ada jalan di antara kedua ruangan tersebut. Oleh karena itu, jika tidak ada kombinasi ruangan yang memiliki jarak nol, maka dapat dipastikan bahwa semua ruangan di dalam peta dapat dijelajahi.

Blok keempat adalah evaluasi. Peta yang telah divalidasi dan dapat dieksplorasi akan dianalisis menggunakan *Space Syntax* untuk mengetahui tingkat integrasi dan konektivitas antar ruangan dalam peta. *Space syntax* merupakan sebuah metode yang digunakan untuk menganalisis bagaimana sebuah lingkungan atau ruang yang dibangun terhubung satu sama lain. Metode ini membantu untuk memahami pilihan-pilihan yang mungkin diambil oleh pengguna dalam menjelajahi ruang tersebut, serta menilai bagaimana penataan ruang bisa mempengaruhi interaksi di dalamnya [19]. Analisis ini diperlukan untuk mengevaluasi seberapa eksploratif peta yang dihasilkan. Pada tahap ini, peta yang dihasilkan di Unity akan dikonversi dengan melakukan *plotting* melalui AutoCAD dan diekspor dalam format berkas DXF. Berkas DXF atau *Drawing Exchange File* di

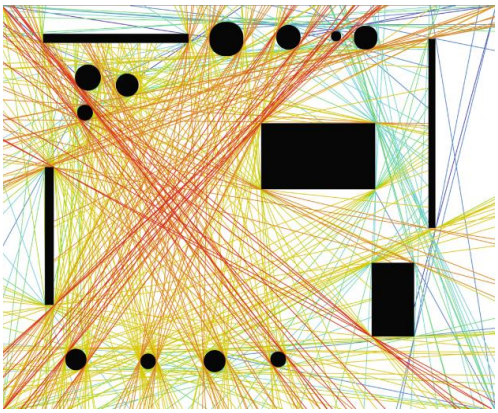
AutoCAD merupakan format berkas grafis yang umum digunakan dan memiliki banyak penerapan di berbagai bidang [22]. Setelah proses konversi selesai dan diimpor ke berkas DXF, berkas tersebut diimpor ke dalam DepthMapX, dimana akan dilakukan dua analisis, yaitu *Visibility Graph Analysis* (VGA) dan *Axial Line Analysis*.

Visibility Graph Analysis (VGA) dan *axial line analysis* adalah metode untuk menganalisis visibilitas dan konektivitas dalam ruang perkotaan. VGA membagi area ke dalam *grid* dan menilai sejauh mana setiap sel terlihat dari sel lainnya, dengan hambatan memengaruhi kedalaman topologis antar sel. Sementara itu, *axial line analysis*, khususnya *all-line axial map*, menggambarkan semua garis pandang yang menghubungkan titik-titik yang saling terlihat dalam suatu ruang. Jika VGA fokus pada hubungan antar titik visibilitas, *axial line analysis* berfokus pada keterkaitan garis pandang dalam sistem ruang secara keseluruhan [23].



Gambar 2. Contoh *Visibility Graph Analysis*. Diambil dari Van Nes & Yamu (2021).

Dalam *Visibility Graph Analysis* seperti pada Gambar 2, area dengan tingkat visibilitas yang tinggi dan kemampuan melihat yang jelas biasanya berada dalam posisi yang sama. Hal ini menunjukkan bahwa lokasi-lokasi tersebut merupakan *hot spot* untuk orientasi dan navigasi. Dalam analisis yang dilakukan, area merah menandakan lokasi tersebut paling terintegrasi, sementara area biru menunjukkan lokasi yang paling terpisah [23].



Gambar 3. Contoh *Axial Line Analysis*. Diambil dari Van Nes & Yamu (2021).

Axial line analysis pada Gambar 3 menunjukkan bahwa area di belakang bangunan dan dinding memiliki nilai visibilitas yang rendah. Sementara itu, area terbuka yang luas adalah tempat yang paling terlihat dan terintegrasi. Garis pandang berwarna merah menunjukkan garis *axial* yang paling terintegrasi, sedangkan garis pandang berwarna biru menunjukkan yang paling terpisah. Analisis ini memberikan gambaran tentang seberapa terintegrasinya semua garis pandang yang mungkin di suatu area dan sangat berguna di tempat-tempat yang ruang dan batasnya ditentukan oleh jaringan jalan [23].

B. Desain Eksperimen

Penelitian ini akan menggunakan beberapa set pengujian dengan parameter yang berbeda. Berikut ini adalah set pengujian yang digunakan:

TABEL 3
KONFIGURASI PARAMETER TIAP PENGUJIAN

Test set	Konfigurasi	
	Luas peta	Ukuran minimum ruangan
1	100×100	15×15
2	100×100	20×20
3	100×100	25×25

Untuk setiap set pengujian, tiga peta permainan yang berbeda akan dibuat secara prosedural dengan menggunakan nilai *seed* 0 sebagai *test set* (a), 40 sebagai *test set* (b), dan -60 sebagai *test set* (c). Dari ketiga set pengujian ini, perbandingan akan dibuat berdasarkan dua tahap analisis:

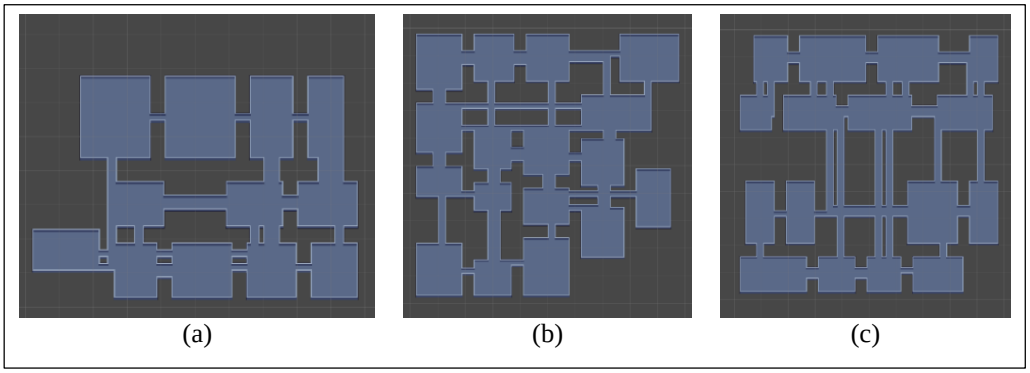
1. *Visibility Graph Analysis* akan digunakan untuk melakukan analisis konektivitas, menghasilkan berbagai posisi yang memungkinkan untuk elemen-elemen permainan.
2. *Axial Line Analysis* menggunakan parameter integrasi dan menampilkan letak sumbu kritis (*critical axes*).

Dalam analisis *space syntax*, kode warna digunakan untuk menunjukkan jangkauan suatu nilai. Dalam penelitian ini, area dan garis berwarna biru disebut *cold spot*, yang berarti interaksi diperkirakan terbatas dan tingkat kesulitannya rendah. Sebaliknya, area dan garis berwarna merah disebut sebagai *hot spot* yang diperkirakan memiliki interaksi yang lebih tinggi dengan tingkat kesulitan yang lebih tinggi.

III. HASIL DAN PEMBAHASAN

A. Hasil Implementasi PCG

Sesuai dengan desain eksperimen yang disebutkan pada bab sebelumnya, berikut ini adalah hasil implementasi untuk setiap *test set*.



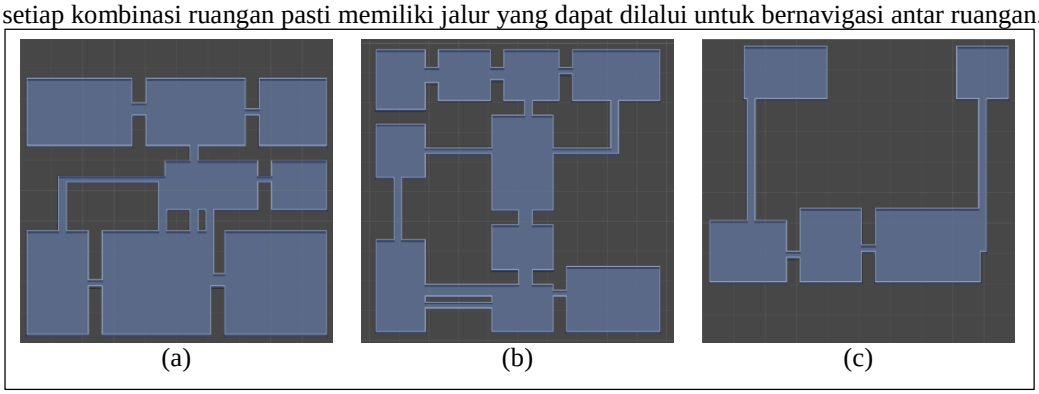
Gambar 4. Hasil PCG *Test Set* 1

Hasil PCG untuk set pengujian pertama dengan ukuran peta 100×100 unit dengan parameter luas ruangan minimum 15×15 unit ditunjukkan pada Gambar 4. Pada Gambar 4(a), dihasilkan 12 ruangan, pada Gambar 4(b) dihasilkan 17 ruangan, dan pada Gambar 4(c) dihasilkan 16 ruangan.

TABEL 4
VALIDASI PETA *TEST SET* 1

Test set	Origin	Destination	Distance
1(a)	(72, 56)	(88, 56)	17
	(91, 10)	(72, 10)	20
	(52, 56)	(72, 56)	21
1(b)	(30, 38)	(30, 56)	19
	(50, 35)	(50, 54)	20
	(30, 89)	(50, 89)	21
1(c)	(10, 32)	(24, 32)	15
	(74, 32)	(92, 32)	19
	(38, 9)	(56, 9)	19

Berdasarkan Tabel 4, semua peta yang dihasilkan pada set uji coba 1 adalah valid. Hal ini dikarenakan dari 3 jarak terkecil dari setiap kombinasi ruangan pada setiap peta, semua nilai jaraknya lebih besar dari 0, yang berarti setiap kombinasi ruangan pasti memiliki jalur yang dapat dilalui untuk bernavigasi antar ruangan.



(a) (b) (c)

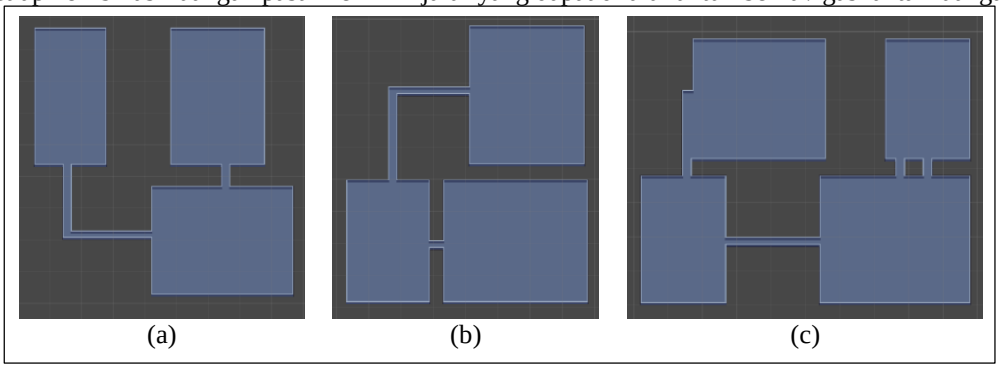
Gambar 5. Hasil PCG *Test Set* 2

Hasil PCG untuk set pengujian kedua dengan ukuran peta 100×100 unit dengan parameter luas ruangan minimum 20×20 unit ditunjukkan pada Gambar 5. Pada Gambar 5(a), dihasilkan 8 ruangan, pada Gambar 5(b) dihasilkan 10 ruangan, dan pada Gambar 5(c) dihasilkan 5 ruangan.

TABEL 5
VALIDASI PETA *TEST SET* 2

Test set	Origin	Destination	Distance
2 (a)	(89, 52)	(61, 52)	29
	(56, 76)	(62, 52)	30
	(56, 76)	(87, 76)	32
2 (b)	(52, 10)	(52, 31)	22
	(32, 89)	(54, 89)	23
	(10, 88)	(32, 89)	24
2 (c)	(14, 30)	(41, 32)	30
	(14, 32)	(72, 32)	32
	(14, 30)	(72, 32)	61

Berdasarkan Tabel 5, semua peta yang dihasilkan pada set uji coba 2 adalah valid. Hal ini dikarenakan dari 3 jarak terkecil dari setiap kombinasi ruangan pada setiap peta, semua nilai jaraknya lebih besar dari 0, yang berarti setiap kombinasi ruangan pasti memiliki jalur yang dapat dilalui untuk bernavigasi antar ruangan.



(a) (b) (c)

Gambar 6. Hasil PCG *Test Set* 3

Hasil PCG untuk set pengujian ketiga dengan ukuran peta 100×100 unit dengan parameter luas ruangan minimum 25×25 unit ditunjukkan pada Gambar 6. Pada Gambar 6(a), dibangkitkan 3 ruangan, pada Gambar 6(b) dibangkitkan 3 ruangan, dan pada Gambar 6(c) dibangkitkan 4 ruangan.

TABEL 6
VALIDASI PETA *TEST SET* 3

Test set	Origin	Destination	Distance
3 (a)	(64, 20)	(63, 65)	47
	(64, 20)	(16, 65)	94
	(16, 65)	(63, 65)	136
3 (b)	(36, 30)	(76, 30)	41
	(36, 30)	(79, 76)	90
	(76, 30)	(79, 76)	126
3 (c)	(86, 80)	(76, 39)	52
	(14, 39)	(76, 39)	63
	(14, 39)	(36, 80)	64

Berdasarkan Tabel 6, semua peta yang dihasilkan pada set uji coba 3 adalah valid. Hal ini dikarenakan dari 3 jarak terkecil dari setiap kombinasi ruangan pada setiap peta, semua nilai jaraknya lebih besar dari 0, yang berarti setiap kombinasi ruangan pasti memiliki jalur yang dapat dilalui untuk bernavigasi antar ruangan.

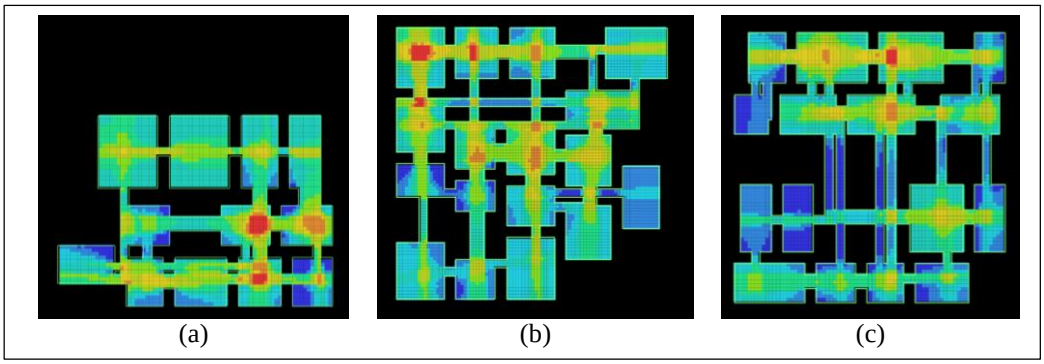
TABEL 7
PERBANDINGAN BANYAK RUANGAN DAN TOTAL WAKTU *GENERATION* TIAP *TEST SET*

Test set	Konfigurasi		Banyak ruangan per *test set*			Total Waktu (ms) *Generation* per *test set*			
	Luas peta	Ukuran minimum ruangan	(a)	(b)	(c)	(a)	(b)	(c)	Rata-rata
1	100×100	15×15	12	17	16	72	121	119	104
2	100×100	20×20	8	10	5	78	86	49	71
3	100×100	25×25	3	3	4	49	54	61	54

Test set 1 menghasilkan ruangan paling banyak dari tiga *test set*, dengan jumlah paling sedikit 12 dan paling banyak 17. Namun, set pengujian 3 menghasilkan ruangan yang lebih luas meskipun berukuran kecil. Hal ini sejalan dengan penelitian Biyik & Sürer, yang menunjukkan bahwa konfigurasi panjang dan lebar yang lebih kecil menghasilkan lebih banyak ruangan tetapi area ruangan yang lebih sempit [16]. Hal ini disebabkan oleh algoritma *Binary Space Partitioning*, yang menentukan kapan nilai ukuran ruangan minimum tercapai. Nilai yang lebih kecil menghasilkan lebih banyak ruangan tetapi ruangan yang dihasilkan lebih kecil, sedangkan nilai yang lebih besar menghasilkan lebih sedikit ruangan dengan area yang lebih luas. Selain itu, hasil pengukuran *runtime* juga mendukung temuan ini, di mana *test set* 1 membutuhkan waktu generasi lebih lama karena banyaknya partisi dan koneksi yang terbentuk, sedangkan *test set* 3 lebih efisien secara komputasi karena jumlah partisi yang lebih sedikit dan struktur BSP yang lebih sederhana.

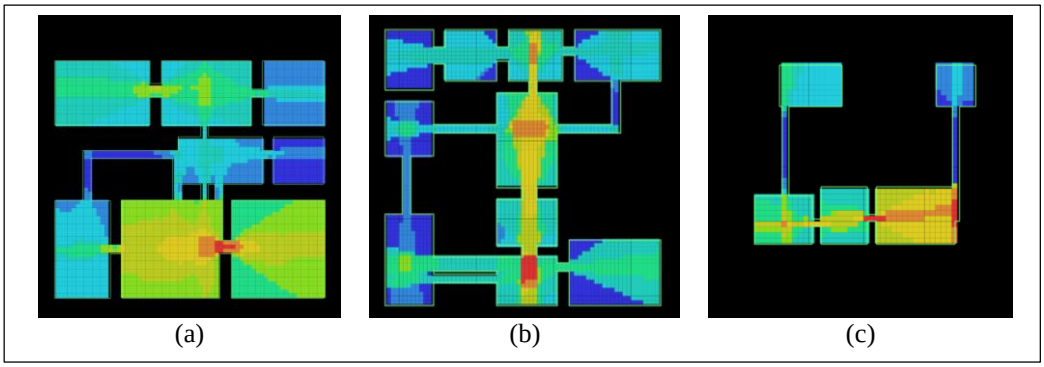
B. Visibility Graph Analysis

Peta yang telah dibuat dan divalidasi di Unity kemudian diplot melalui AutoCAD. Hasil peta dari AutoCAD kemudian diimpor ke dalam DepthMapX untuk analisis *Space Syntax*, yaitu *visibility graph analysis*. Analisis ini dilakukan untuk mengetahui tingkat konektivitas dan visibilitas peta.



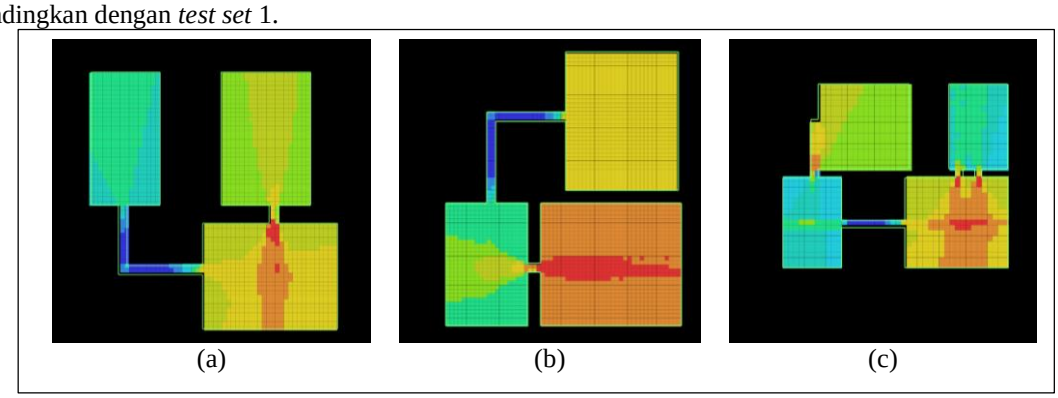
Gambar 7. Hasil Konektivitas *Visibility Graph Analysis Test Set* 1

Pada Gambar 7, dari ketiga peta tersebut terdapat beberapa *hot spot* yang dihasilkan. Terdapat 3 *hot spot* yang dihasilkan pada Gambar 7(a), 7 *hot spot* yang dihasilkan pada Gambar 7(b), dan 2 *hot spot* yang dihasilkan pada Gambar 7(c). Secara visual, warna dari ketiga peta tersebut relatif sama dan didominasi oleh warna biru muda hingga hijau. Hal ini mengindikasikan tingkat konektivitas dan jarak pandang yang seragam antar ruangan. Selain itu, terdapat banyak area berwarna kuning dengan tingkat konektivitas bernilai sedang yang berada di tengah ruangan yang terhubung dengan lebih dari 1 ruangan.



Gambar 8. Hasil Konektivitas *Visibility Graph Analysis Test Set* 2

Pada Gambar 8, dari ketiga peta tersebut, *hot spot* yang dihasilkan lebih sedikit jika dibandingkan dengan Gambar 7. Terdapat 1 *hot spot* yang dihasilkan pada Gambar 8(a), 3 *hot spot* yang dihasilkan pada Gambar 8(b), dan 1 *hot spot* yang dihasilkan pada Gambar 8(c). *Test set* 2 memiliki area kuning yang lebih sedikit jika dibandingkan dengan *test set* 1.



Gambar 9. Hasil Konektivitas *Visibility Graph Analysis Test Set* 3

Pada Gambar 9, dari ketiga peta, 1 *hot spot* dihasilkan pada setiap peta, tetapi area *hot spot* lebih luas jika dibandingkan dengan *test set* 1 dan 2. Selain itu, perbedaan warna antar ruangan lebih mencolok jika dibandingkan dengan set uji 1 dan 2. Hal ini mengindikasikan tingkat konektivitas yang lebih beragam.

TABEL 8
PERBANDINGAN BANYAK *HOT SPOT* TIAP TEST SET DENGAN *VISIBILITY GRAPH ANALYSIS*

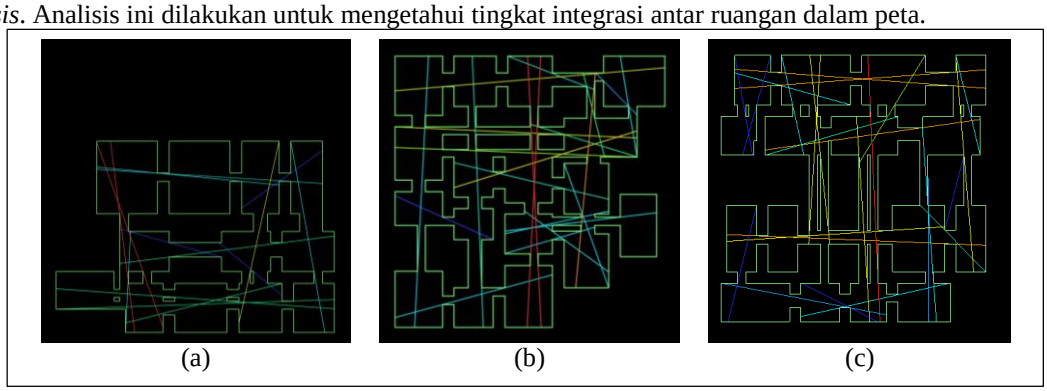
Test set	Konfigurasi		Banyak *hot spot* per *test set*		
	Luas peta	Ukuran minimum ruangan	(a)	(b)	(c)
1	100×100	15×15	3	7	2
2	100×100	20×20	1	3	1
3	100×100	25×25	1	1	1

Test set 1 dengan konfigurasi ukuran ruangan minimum 15×15 menghasilkan peta dengan konektivitas dan visibilitas yang lebih rendah jika dibandingkan dengan *test set* 2 dan 3. Sementara itu, *test set* 3 dengan konfigurasi ruangan 25×25 menghasilkan peta dengan visibilitas dan konektivitas yang lebih tinggi. Hal ini ditunjukkan dengan meningkatnya nilai rata-rata konektivitas seiring dengan bertambahnya luas ruangan. Namun, *test set* 1 memiliki lebih banyak *hot spot* dan lebih tersebar jika dibandingkan dengan *test set* 2 dan 3. Sedangkan, *test set* 3 memiliki *hot spot* yang sangat sedikit namun dengan area yang sangat luas. Dapat disimpulkan bahwa semakin besar ruang minimum, semakin tinggi konektivitas dan visibilitas peta, tetapi dengan *hot spot* yang lebih sedikit dan area yang lebih luas. Semakin sedikit *hot spot*, semakin sedikit titik untuk berinteraksi dengan peta. Kebalikannya juga berlaku, semakin kecil ukuran ruang minimum, konektivitas dan visibilitas menurun, tetapi jumlah *hot spot* meningkat dengan area yang lebih sedikit. Hal ini menyebabkan lebih banyak interaksi yang terjadi.

Jika dibandingkan dengan penelitian Biyik & Sürer [19], terdapat perbedaan yang cukup signifikan pada struktur hasil ruangannya. Pada penelitian mereka, struktur ruangan cenderung linear, di mana satu ruangan biasanya hanya terkoneksi dengan satu ruangan lain atau paling banyak dua hingga tiga ruangan. Sebaliknya, pada penelitian ini, ruangan dapat terhubung dengan lebih dari dua ruangan sekaligus, sehingga menghasilkan struktur jaringan yang lebih bercabang dan kompleks. Perbedaan ini berimplikasi pada pengalaman eksplorasi: struktur linear cenderung memberikan jalur eksplorasi yang terarah, sedangkan struktur bercabang memungkinkan eksplorasi yang lebih bebas dengan pilihan jalur yang lebih beragam. Selain itu, distribusi *hot spot* juga menunjukkan perbedaan yang penting. Pada penelitian Biyik & Sürer, *hot spot* cenderung terpusat hanya pada satu ruangan terbesar dalam peta, sehingga interaksi pemain lebih terkonsentrasi pada area tertentu. Sebaliknya, pada penelitian ini *hot spot* tersebar di beberapa ruangan, yang memberikan lebih banyak titik pilihan untuk interaksi dan menghasilkan pola eksplorasi yang lebih dinamis.

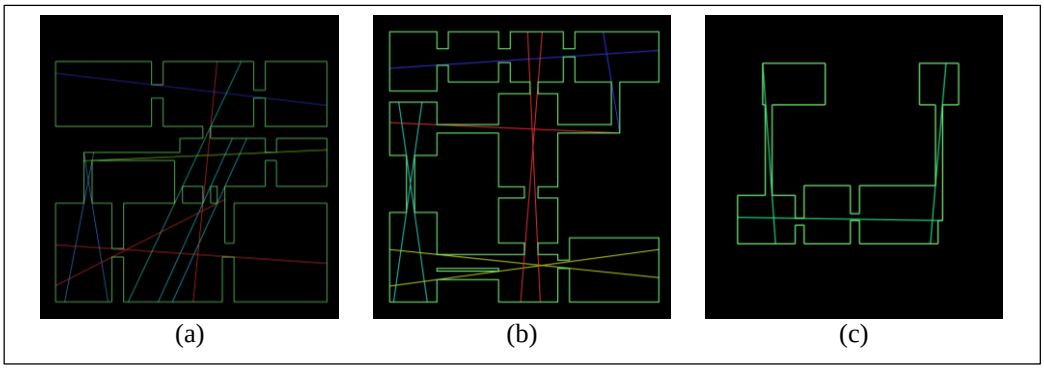
C. Axial Line Analysis

Peta yang telah dibuat dan divalidasi di Unity kemudian diplot melalui AutoCAD. Hasil peta dari AutoCAD kemudian diimpor ke dalam DepthMapX untuk dilakukan analisis *Space Syntax*, yaitu *axial line analysis*. Analisis ini dilakukan untuk mengetahui tingkat integrasi antar ruangan dalam peta.



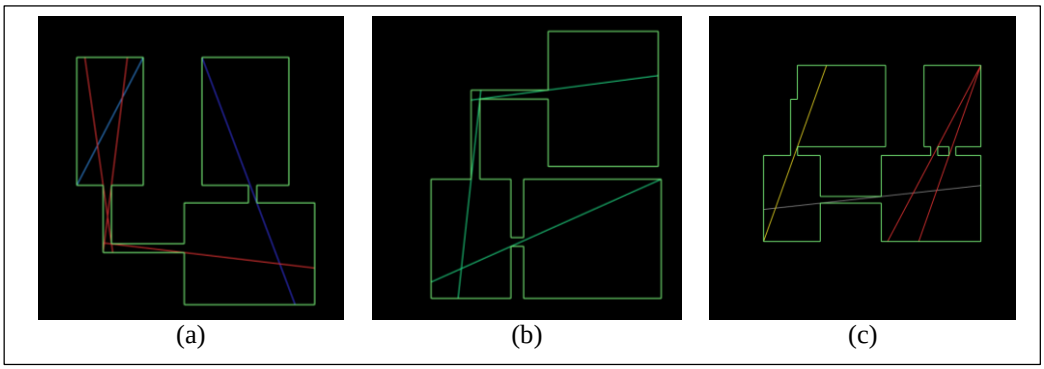
Gambar 10. Hasil Integrasi *Axial Line Analysis Test Set* 1

Pada Gambar 10, dari ketiga peta tersebut terdapat beberapa *hot spot* yang dihasilkan. *Hot spot* ditunjukkan dengan garis berwarna oranye dan merah. Terdapat 2 *hot spot* yang dihasilkan pada Gambar 10(a), 3 *hot spot* yang dihasilkan pada Gambar 10(b), dan 5 *hot spot* yang dihasilkan pada Gambar 10(c).



Gambar 11. Hasil Integrasi *Axial Line Analysis Test Set* 2

Pada Gambar 11, dari ketiga peta tersebut terdapat beberapa *hot spot* yang dihasilkan. Terdapat 3 *hot spot* yang dihasilkan pada Gambar 11(a), 3 *hot spot* yang dihasilkan pada Gambar 11(b). Namun tidak ada *hot spot* yang dihasilkan pada Gambar 11(c) karena warna semua garis aksial sama.



Gambar 12. Hasil Integrasi *Axial Line Analysis Test Set* 3

Pada Gambar 12, dari ketiga peta tersebut terdapat beberapa *hot spot* yang dihasilkan. Terdapat 3 *hot spot* yang dihasilkan pada Gambar 12(a) dan 2 *hot spot* yang dihasilkan pada Gambar 12(c). Namun tidak ada *hot spot* yang dihasilkan pada Gambar 12(b) karena warna semua garis aksial sama.

TABEL 9
PERBANDINGAN BANYAK *HOT SPOT* TIAP *TEST SET* DENGAN *AXIAL LINE ANALYSIS*

Test set	Konfigurasi		Banyak *hot spot* per *test set*		
	Luas peta	Ukuran minimum ruangan	(a)	(b)	(c)
1	100×100	15×15	2	3	5
2	100×100	20×20	3	3	0
3	100×100	25×25	3	0	2

Test set 1 dengan konfigurasi ukuran ruangan minimum 15×15 menghasilkan peta dengan integrasi yang lebih rendah jika dibandingkan dengan *test set* 2 dan 3. Sementara itu, *test set* 3 dengan konfigurasi ruangan 25×25 menghasilkan peta dengan integrasi yang lebih tinggi. Dapat disimpulkan bahwa semakin kecil ukuran ruangan minimum akan menghasilkan peta dengan integrasi yang lebih rendah dan banyak ruangan yang tersegregasi. Selain itu, semakin kecil ukuran ruangan minimum akan menghasilkan lebih banyak garis aksial yang berarti lebih banyak pergerakan yang dibutuhkan untuk mengakses satu area ke area lainnya. Hal ini juga menunjukkan bahwa ukuran ruangan minimum yang lebih kecil akan menghasilkan peta dengan jaringan yang lebih kompleks.

IV. SIMPULAN

Berdasarkan penelitian yang telah dilakukan, *Procedural Content Generation* dengan algoritma *Binary Space Partitioning* dapat diterapkan untuk menghasilkan peta permainan yang dinamis dan variatif karena ukuran ruangan yang dibangkitkan secara acak dan koridor yang menghubungkan ruangan-ruangan secara acak. Kualitas peta game dapat dievaluasi dan diukur dengan menggunakan metode *Space Syntax* yaitu *Visibility*

Graph Analysis dan *Axial Line Analysis*. *Visibility Graph Analysis* dapat digunakan untuk menentukan tingkat konektivitas dan visibilitas dan *Axial Line Analysis* untuk menentukan tingkat integrasi dalam peta. Visibilitas, konektivitas, dan integrasi peta permainan menurun seiring dengan semakin kecilnya konfigurasi ukuran ruangan karena ruangan yang dihasilkan semakin banyak dan terpisah-pisah dengan jaringan yang kompleks. Namun, lebih banyak titik pilihan untuk interaksi ditandai dengan lebih banyak *hot spot*.

Meskipun demikian, penelitian ini masih memiliki keterbatasan. Evaluasi yang dilakukan hanya mencakup analisis struktur spasial menggunakan *Space Syntax*, sehingga aspek lain seperti pengalaman pemain, waktu komputasi, atau tingkat kesulitan navigasi belum diperhitungkan. Selain itu, penelitian ini berfokus pada penggunaan algoritma BSP secara tunggal tanpa pembandingan kuantitatif dengan metode generatif lain seperti Cellular Automata, Random Walk, atau Perlin Noise. Oleh karena itu, penelitian selanjutnya disarankan untuk menambahkan uji komparatif dengan algoritma lain serta memperluas metrik evaluasi. Lebih jauh lagi, pendekatan ini berpotensi dikembangkan ke ranah *procedural city generation* yang membutuhkan struktur spasial lebih kompleks, serta diintegrasikan dengan metode pembelajaran mesin untuk melakukan tuning parameter BSP secara otomatis agar hasil peta lebih adaptif terhadap kebutuhan desain permainan.

UCAPAN TERIMAKASIH

Penulis mengungkapkan rasa terima kasih yang mendalam kepada KBK *Game* dan Teknologi Digital Cerdas dari Departemen Teknik Elektro dan Informatika, Universitas Negeri Malang atas pemberian hibah publikasi penelitian dengan nomor 21.2.51/UN32/KP/2025. Hibah ini tidak hanya memberikan bantuan finansial, tetapi juga membuka peluang untuk melaksanakan penelitian yang telah dirancang. Keberhasilan dan perkembangan penelitian ini tidak terlepas dari kontribusi serta dukungan berharga yang diberikan oleh Universitas Negeri Malang.

DAFTAR PUSTAKA

[1] P. O. Toups Dugas, N. Lalone, S. A. Alharthi, H. N. Sharma, dan A. M. Webb, "Making Maps Available for Play: Analyzing the Design of Game Cartography Interfaces," *ACM Trans. Comput.-Hum. Interact.*, vol. 26, no. 5, hlm. 1–43, Okt 2019, doi: 10.1145/3336144.

[2] T. Nordvig Møller, J. Billeskov, dan G. Palamas, "Expanding Wave Function Collapse with Growing Grids for Procedural Map Generation," dalam *International Conference on the Foundations of Digital Games*, Bugibba Malta: ACM, Sep 2020, hlm. 1–4. doi: 10.1145/3402942.3402987.

[3] W. L. Raffe, "Personalized procedural map generation in games via evolutionary algorithms," *SIGEVOlution*, vol. 7, no. 1, hlm. 27–28, Agu 2014, doi: 10.1145/2661735.2661739.

[4] M. Prachyabrued, T. E. Roden, dan R. G. Benton, "Procedural generation of stylized 2D maps," dalam *Proceedings of the international conference on Advances in computer entertainment technology*, Salzburg Austria: ACM, Jun 2007, hlm. 147–150. doi: 10.1145/1255047.1255077.

[5] O. Davoodi, M. Ashtiani, dan M. Rajabi, "An Approach for the Evaluation and Correction of Manually Designed Video Game Levels Using Deep Neural Networks," *The Computer Journal*, vol. 65, no. 3, hlm. 495–515, Mar 2022, doi: 10.1093/comjnl/bxaa071.

[6] M. Hendrikx, S. Meijer, J. Van Der Velden, dan A. Iosup, "Procedural content generation for games: A survey," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 9, no. 1, hlm. 1–22, Feb 2013, doi: 10.1145/2422956.2422957.

[7] J. Roberts dan K. Chen, "Learning-Based Procedural Content Generation," 9 Oktober 2013, *arXiv*: arXiv:1308.6415. Diakses: 17 September 2024. [Daring]. Tersedia pada: http://arxiv.org/abs/1308.6415

[8] J. Togelius, M. Preuss, dan G. N. Yannakakis, "Towards multiobjective procedural map generation," dalam *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, Monterey California: ACM, Jun 2010, hlm. 1–8. doi: 10.1145/1814256.1814259.

[9] J. Togelius, G. N. Yannakakis, K. O. Stanley, dan C. Browne, "Search-Based Procedural Content Generation: A Taxonomy and Survey," *IEEE Trans. Comput. Intell. AI Games*, vol. 3, no. 3, hlm. 172–186, Sep 2011, doi: 10.1109/tciaig.2011.2148116.

[10] W. M. P. Reis, L. H. S. Lelis, dan Y. K. Gal, "Human computation for procedural content generation in platform games," dalam *2015 IEEE Conference on Computational Intelligence and Games (CIG)*, Tainan, Taiwan: IEEE, Agu 2015, hlm. 99–106. doi: 10.1109/cig.2015.7317906.

[11] A. B. Moghadam dan M. K. Rafsanjani, "A genetic approach in procedural content generation for platformer games level creation," dalam *2017 2nd Conference on Swarm Intelligence and Evolutionary Computation (CSIEC)*, Kerman, Iran: IEEE, Mar 2017. doi: 10.1109/csiec.2017.7940160.

[12] D. A. Ramadhan dan A. D. Indriyanti, "Procedural Content Generation pada Game World Exploration Sandbox Menggunakan Alogoritma Perlin Noise," *JINACS*, vol. 4, no. 01, hlm. 86–91, Jul 2022, doi: 10.26740/jinacs.v4n01.p86-91.

[13] D. Ashlock, "Evolvable fashion-based cellular automata for generating cavern systems," dalam *2015 IEEE Conference on Computational Intelligence and Games (CIG)*, Tainan, Taiwan: IEEE, Agu 2015, hlm. 306–313. doi: 10.1109/CIG.2015.7317958.

[14] E. W. Hidayat, E. N. F. Dewi, dan I. S. Ramadhan, "APPLICATION OF PROCEDURAL CONTENT GENERATION SYSTEM IN FORMING DUNGEON LEVEL IN DUNGEON DIVER GAME," 2024.

[15] R. Van Der Linden, R. Lopes, dan R. Bidarra, "Procedural Generation of Dungeons," *IEEE Trans. Comput. Intell. AI Games*, vol. 6, no. 1, hlm. 78–89, Mar 2014, doi: 10.1109/tciaig.2013.2290371.

[16] M. Ijai, A. Arbansyah, dan S. H. Suryawan, "Penerapan Algoritma Random Walk Untuk Procedural Map Pada Game 'The Last Hope' 2D," *CoSciTech*, vol. 4, no. 3, hlm. 754–762, Jan 2024, doi: 10.37859/coscitech.v4i3.6420.

[17] J. R. Baron, "Procedural Dungeon Generation Analysis and Adaptation," dalam *Proceedings of the SouthEast Conference*, Kennesaw GA USA: ACM, Apr 2017, hlm. 168–171. doi: 10.1145/3077286.3077566.

[18] S. Snodgrass, "Levels from Sketches with Example-Driven Binary Space Partition," *AIIDE*, vol. 15, no. 1, hlm. 73–79, Okt 2019, doi: 10.1609/aiide.v15i1.5227.

[19] E. A. Biyik dan E. Sürer, "DEVELOPING A SPACE SYNTAX-BASED EVALUATION METHOD FOR PROCEDURALLY GENERATED GAME LEVELS," *Mugla Journal of Science and Technology*, vol. 6, no. 2, hlm. 79–88, Des 2020, doi: 10.22531/muglajsci.706895.

[20] S. Snodgrass dan A. Sarkar, "Multi-Domain Level Generation and Blending with Sketches via Example-Driven BSP and Variational Autoencoders," dalam *International Conference on the Foundations of Digital Games*, Bugibba Malta: ACM, Sep 2020, hlm. 1–11. doi: 10.1145/3402942.3402948.

[21] L. H. O. Rios dan L. Chaimowicz, "A Survey and Classification of A* Based Best-First Heuristic Search Algorithms," dalam *Advances in Artificial Intelligence – SBIA 2010*, vol. 6404, A. C. Da Rocha Costa, R. M. Vicari, dan F. Tonidandel, Ed., dalam Lecture Notes in Computer Science, vol. 6404. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, hlm. 253–262. doi: 10.1007/978-3-642-16138-4_26.

[22] W. Shang, J. Zhong, dan Q. Yan, "Analysis of DXF file with an application to 3D graphic display," dalam *2012 IEEE International Conference on Information and Automation*, Shenyang, China: IEEE, Jun 2012, hlm. 611–615. doi: 10.1109/ICInfA.2012.6246886.

[23] A. Van Nes dan C. Yamu, "Orientation and Wayfinding: Measuring Visibility," dalam *Introduction to Space Syntax in Urban Studies*, Cham: Springer International Publishing, 2021, hlm. 87–111. doi: 10.1007/978-3-030-59140-3_3.