BAB II

LANDASAN TEORI

2.1 Designing sebuah video game

Sebuah *video game* dirancang bukan karena para *developer* ingin memecahkan sebuah masalah tetapi karena *game* itu menyenangkan untuk dimainkan (Hotz, 2012). Ketika merancang bangun sebuah *video game* diperlukan untuk mempertimbangkan beberapa hal, yaitu (Gibson, 2015).

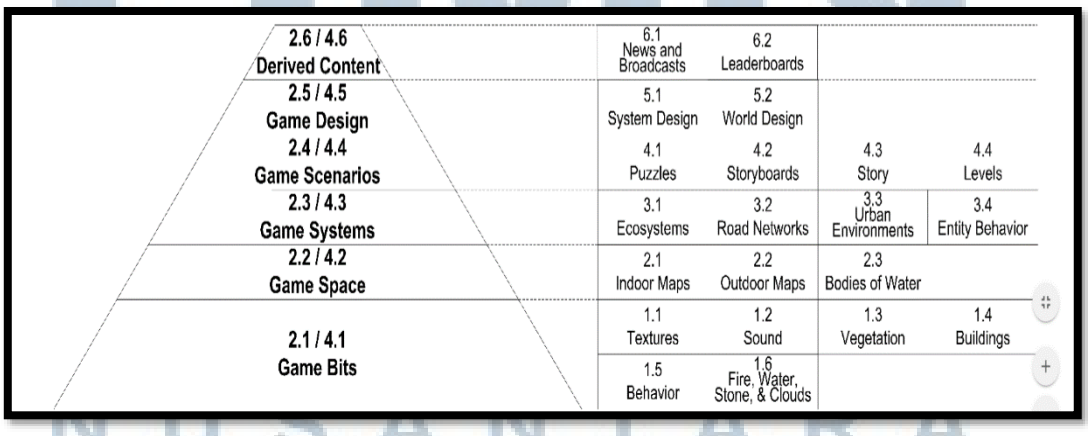
1. Siapa target pemainnya
2. *Hardware* dimana *game* akan dimainkan
3. Modal yang ada
4. Batas waktu ketika membangun *game*
5. Seberapa *skill* yang dibutuhkan
6. Apa saja hal yang bisa digunakan untuk membangun *game*
7. Kapan percobaan akan dilakukan
8. Siapa yang akan menjadi *tester*
9. Teknik macam apa yang akan digunakan untuk menilai *game*
10. Seberapa besar *team*/*developer*
11. Kapan *game* dapat dianggap selesai dirancang.

2.2 Procedural Content Generation

Procedural Content Generation (PCG*)* adalah suatu teknik pembuatan konten dengan sedikit input dari *user* atau tak ada sama sekali, dengan kata lain, *Procedural Content Generation* adalah kemampuan suatu *software* untuk membuat konten *game* secara sendiri (Togelius dkk., 2013). *Procedural Content*

Generation (PCG) terikat ke beberapa bidang seperti *computational aesthetics* dan *computational creativity* dan menjadi semakin penting dalam area *human-computer interaction* (Yannakakis & Togelius, 2011). *Games, Web 2.0, interface* dan desain *software* termasuk aplikasi populer yang menggunakan PCG (Yannakakis & Togelius, 2011). Dalam konteks *games* konten bisa berupa: *levels, maps, game rules, textures, stories, items, quest, music, weapon, vehicles, characters,* dan lain-lain namun bukan *game engine* sendiri atau *behaviour* suatu *Non-Playable Character* (Togelius dkk., 2013). Untuk membuat suatu *generator* komputer harus memiliki *skill* untuk dapat men-*generate* konten (Compton, 2016) *skill* ini berupa:

1. Enkapsulasi opsi ilmu.
2. Buat struktur.
3. *Encode* peraturan kondisional untuk opsi.
4. Buat variasi di struktur.
5. Dapat menanyakan batasannya sendiri.



Gambar 2. 1 Tipe Konten yang Bisa di-*Generate* di Game (Hendrikx dkk., 2011)

Sekarang sudah ada banyak metode untuk melakukan PCG di *game* (Hendrikx dkk., 2011). Namun belum ada teknik atau algoritma yang dapat digunakan untuk men-*generate* semua jenis konten di *game,* hampir semua algoritma PCG yang ada digunakan untuk men-*generate* satu jenis konten (Togelius dkk., 2013), untuk membuat konten *narrative* dan *game space* akan digunakan dua jenis metode. Metode pertama adalah *Grammars* untuk men-*generate* cerita atau *narrative* di dalam *game*, kemudian *Room-generating algorithm* dan *corridor-generating algorithm* untuk men-*generate game space* di dalam *game.*

2.3 Narrative and Game Space

Narrative dan *game space* merupakan konten penting di dalam sebuah *game* (Togelius dkk., 2013). *Game*, pada umumnya, akan memiliki semacam *narrative* dan *game space* yang saling mendukung satu sama lain. *Game space* membantu *game* menceritakan *narrative,* dan *narrative* membantu pemain menjelajahi *game space.* Namun, masih sedikit riset yang dilakukan dalam men-*generate narrative* dan *game space* bersamaan. Contoh riset dalam topik ini pernah dilakukan oleh Joris Dormans (2010) dan Kan Hartsook (2013).

2.4 Narrative in Video Game

Peran cerita di *video game* modern adalah untuk memunculkan motivasi dan mendorong pemain untuk melakukan aksi di *game* (Adams dan Short, 2017). Dengan definisi ini bisa diasumsikan bahwa sesederhana apapun *video game* akan mempunyai suatu cerita di dalamnya (Ostenson, 2013). Contoh dari ini adalah *game* dengan nama *Angry birds* dimana ceritanya pemain bermain sebagai burung

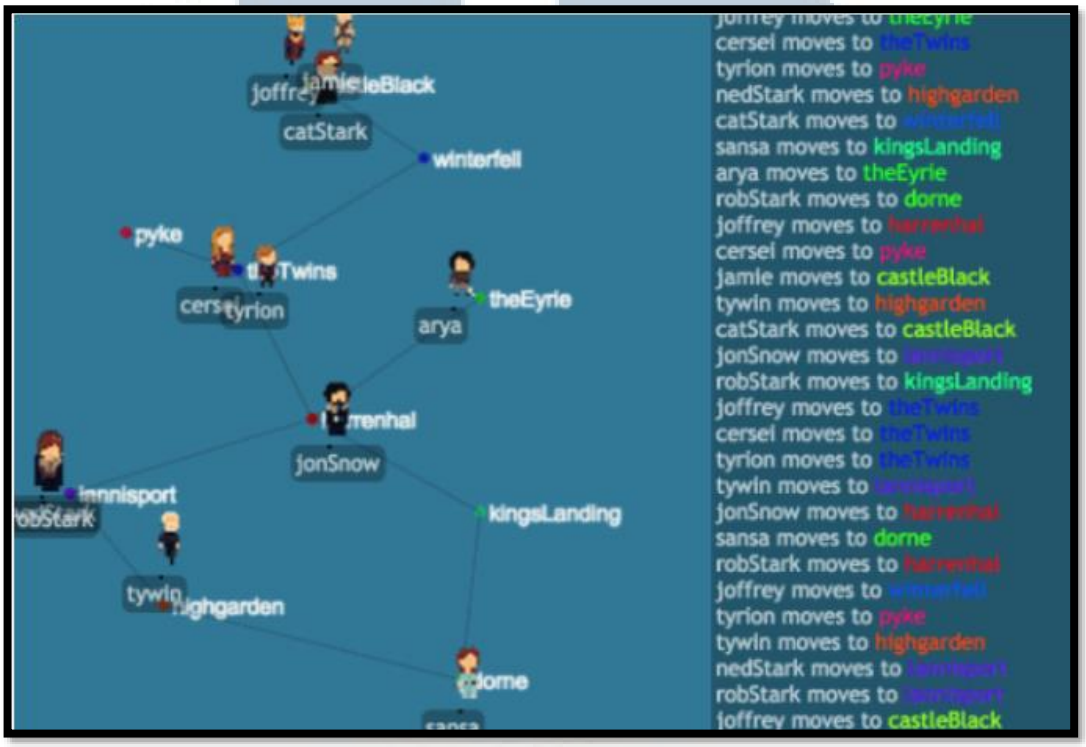
yang meluncurkan dirinya kepada sebuah struktur yang didirikan oleh kelompok babi hijau, dimana konflik dari cerita ini adalah para babi hijau ini telah mencuri telur milik burung (Ostenson, 2013). *Game* ini memiliki cerita walaupun perannya tidak krusial di dalam *game* dan hanya sekedar untuk mendorong refleks pemain untuk bermain (Ostenson, 2013).

Namun ada juga *game* yang menaruh cerita sebagai hal krusial di dalam *game*nya, contoh *game* seperti ini adalah *Dungeon and Dragon* dimana cerita terstruktur sangat kompleks dan memiliki peran penting di dalam *game* (Ostenson, 2013). Cerita di dalam *video game* harus juga sesuai dengan *game design game* dan lingkungan *game*, karena itu membuat suatu cerita dengan *generator* sangat berat dan kompleks (Adam dan Short, 2017) namun tidaklah mustahil, *Grammars* merupakan metode yang sering digunakan oleh developer ketika men-*generate* sebuah *story* atau *narrative* di dalam *game* (Adam dan Short, 2017) sebuah pernyataan yang didorong oleh riset Kart Hartsook (2011) dan Joris Dormans (2010) dimana riset menggunakan *Grammars* untuk men-*generate narrative* di dalam *game*.

2.5 Grammars

Grammars memiliki konsep bahwa suatu hal yang kompleks terdiri dari hal hal yang lain, dan hal hal lain itu bisa juga terdiri dari hal hal yang lebih kecil lagi. Contohnya seperti suatu *galaxy* mempunyai planet-planet, planet memiliki benua, benua memiliki negara, negara memiliki kota, dan seterusnya. Setiap hal memilki

sub-hal dan ketika di *"unpack"* harus memilih dari salah satu sub-halnya (Compton, 2016).



Gambar 2. 2 Gambaran Bagaimana *Grammars* Berkerja (Compton, 2016)

Grammars terbentuk dari simbol-simbol dan *rules* yang akan memutuskan bagaimana simbol-simbol akan dikonstruksi (Adam dan Short, 2017). Contoh ada tiga simbol yaitu *A, B,* dan *C* yang mengikuti ketiga *rules* ini:

1. A -> C
2. A -> B
3. B -> CC

Rules ini biasanya dipanggil dengan *rewrite rules*, dimana simbol yang berada di sebelah kiri panah adalah *pattern*, jika ada di dalam kumpulan *string* simbol maka dapat ditulis ulang dengan simbol yang berada di sebelah kanan panah. Ini akan menjadi pedoman untuk men-*generate* konten cerita di sebuah

game. Hal ini dilakukan dengan mendefinisikan simbol-simbol dengan *potential event* dari sebuah *quest* di sebuah *game* (Adam dan Short, 2017). Contoh ada simbol yang didefinisikan seperti ini [Berpetualang], [Konflik], dan [Penemuan] ini contoh dari apa yang bisa terjadi di sebuah cerita. Selanjutnya membuat *rules*-nya:

1.[Berpetualang] -> [Berpetualang][Konflik]

2.[Konflik]->[Konflik][Konflik]

3.[Konflik] -> [Penemuan][Berpetualang]

Dengan ini jika mulai dengan [Berpetualang] bisa didapatkan:

1. [Berpetualang]
2. [Berpetualang][Konflik]
3. [Berpetualang][Konflik][Konflik]
4. [Berpetualang][Konflik][Penemuan][Berpetualang]
5. [Berpetualang][Konflik][Penemuan][Berpetualang][Konflik]

Ini salah satu contoh yang dapat dihasilkan dengan *rules* yang dibuat. Tahap selanjutnya adalah mendefinisikan isi dari simbol-simbol, dengan mengambil struktur abstrak yang baru saja dibuat menjadi suatu hal yang konkrit.

1. [Berpetualang] -> [Pergi ke kota]
2. [Berpetualang] -> [Pergi ke masa depan]
3. [Konflik] -> [Bertemu dengan bandit]
4. [Konflik] -> [Bertemu dengan iblis]
5. [Penemuan] -> [Menemukan senjata tembak]
6. [Penemuan] -> [Menemukan pedang]

Jika diimplementasikan *rule* ini dengan contoh sebelumnya maka bisa mendapatkan suatu *quest* atau *narrative* yang konkrit.

Sebelumnya:

[Berpetualang][Konflik][Penemuan][Berpetualang][Konflik]

Setelah di *apply* dengan *rules* yang baru saja dirancang:

[Pergi ke masa depan][Bertemu Iblis][Menemukan Pedang][Pergi ke kota][Bertemu dengan bandit].

Salah satu keuntungan dari menggunakan *grammars* adalah metode ini gampang dimengerti dan mudah diimplementasikan untuk PCG (Adam and Short, 2017). *Grammars* juga dapat digunakan untuk men-*generate* konten seperti *game space* (Dormans, 2010). Satu kekurangan dari metode cerita yang di-*generate* akan terlalu *dependent* terhadap dirinya sendiri, dimana tidak akan ada koneksi dengan *game world* (Adam and Short, 2017).

2.6 Level Generation atau Game space

Level atau *game space* adalah lingkungan dari sebuah *game*, dimana lingkungan ini akan diisi dengan konten-konten *visual* dimana pemain juga bisa bernavigasi (Hendrikx dkk., 2011). *Game space* bisa dibagi menjadi tiga tipe yaitu *Indoor Maps, Outdoor Maps,* dan *Bodies of water*.

1. *Indoor Maps:* Struktur dan ruang yang diset menjadi suatu ruangan. Ruangan dapat terkoneksi antara satu dengan yang lain dengan koridor dan terbagi menjadi sebuah *dungeon* (Hendrikx dkk., 2011).
2. *Outdoor Maps: Outdoor Maps* biasa dipanggil sebagai *terrains.* Lingkungan yang biasanya memiliki *Indoor Maps* di dalamnya (Hendrikx dkk., 2011).

3. *Bodies of Water*: Seperti halnya danau, perairan, dan lautan biasanya digunakan sebagai rintangan di *game* (Hendrikx dkk,2011).

Untuk *game* yang akan dikonstruksi akan lebih fokus kepada *game space* tipe *Indoor Maps.* Banyak sekali cara untuk men-*generate* suatu *Indoor Maps,* namun teknik yang sering digunakan untuk men-*generate* suatu *Indoor Maps* adalah *Pseudo-Random Number Generators* (Pullen, 2011).

Tapi untuk *game* yang akan dirancang bangun *Indoor Maps* atau *Dungeon* akan di-*generate* dengan proses atau teknik Jessica R. Baron (2017), dimana terdapat lima algoritma yang bisa digunakan untuk membuat suatu *Procedural generated dungeon.* Algoritma ini dibagi menjadi *room-generating* dan *corridor-generating* dimana akan digabung menjadi lima jenis *map generating algorithm pairs. Room-generating algorithm* berupa *Random Room Placement* dan *Binary Space Partitioning (BSP) Room placement*, sementara *corridor-generating* berupa *Random Point Connect, Drunkard Walk*, dan *BSP Corridors (*Baron, 2017*).*

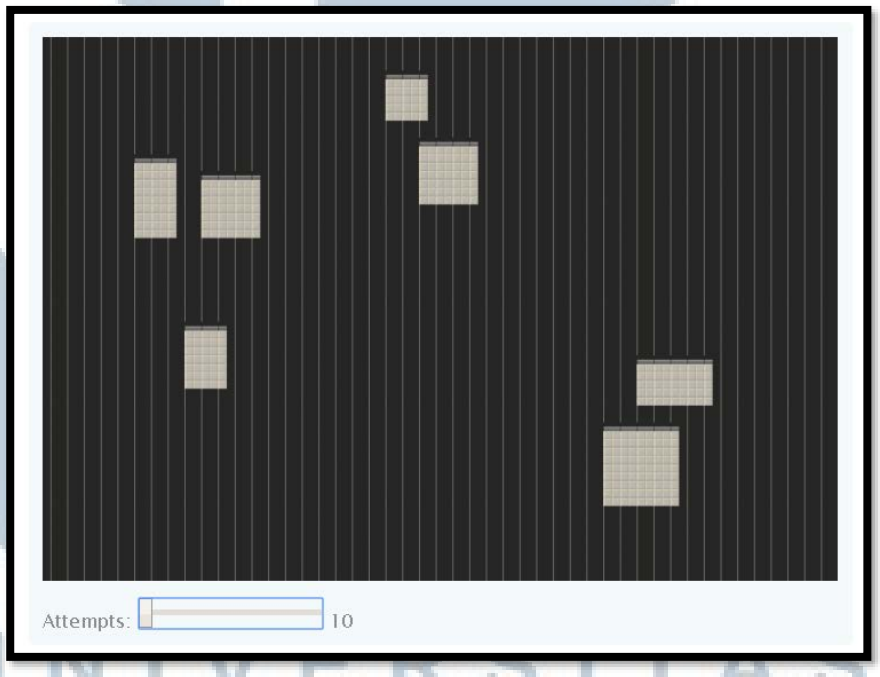
2.6.1 Room-Generating Algorithm

Game yang memiliki *game space Indoor* akan terdiri atas ruangan-ruangan yang tersambung dengan koridor. *Room-Generating Algorithm* adalah algoritma/teknik yang digunakan untuk membuat dan membangun ruangan di dalam *game* secara *procedural generated* (Baron, 2017). Sebelum sebuah *corridors* dibentuk di-*generate* dulu ruangan-ruangan, terdapat dua cara untuk melakukan ini (Baron, 2017).

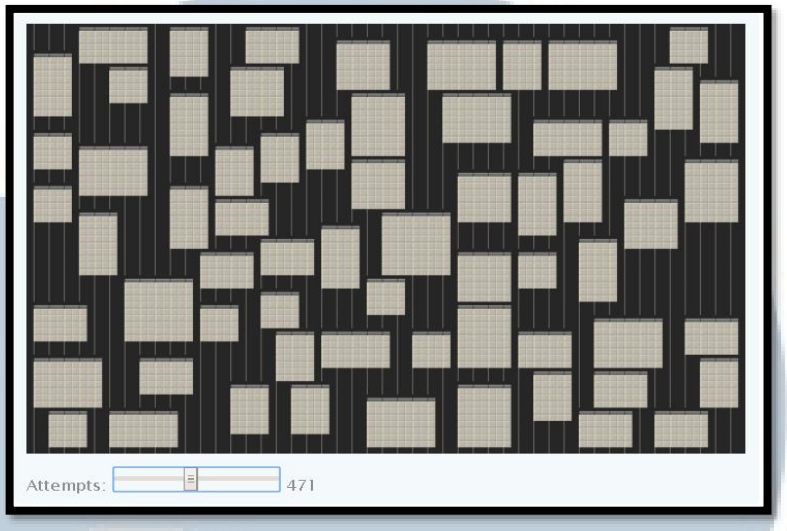
1. *Random Room Placement*. Sebuah algoritma *brute force* untuk peletakan ruangan dimana setiap ruangan dibuat dengan lebar dan tinggi *random* (yang

sudah dibatasi lebar dan tingginya), kemudian dipilih *vertex* x dan y secara *random* pada *graph* dimana nanti dicek mana saja yang bertabrakan. Koordinat peletakan ruangan berada di kiri atas ruangan dan jika ruangan yang ingin diletakan bertabrakan dengan ruangan lain maka ruangan yang sedang dipegang akan mencoba mencari *vertex* lain sampai batas *attempt* atau sampai mendapat koordinat yang tidak bertabrakan dengan ruangan lain. Proses ini akan terus berjalan secara *loop* sampai jumlah ruangan yang diinginkan tercapai (Baron, 2017).

Namun, Bob Nystrom (2014) memodifikasi algoritma ini dengan menggunakan *variable attempt* untuk mengakhiri *loop*.

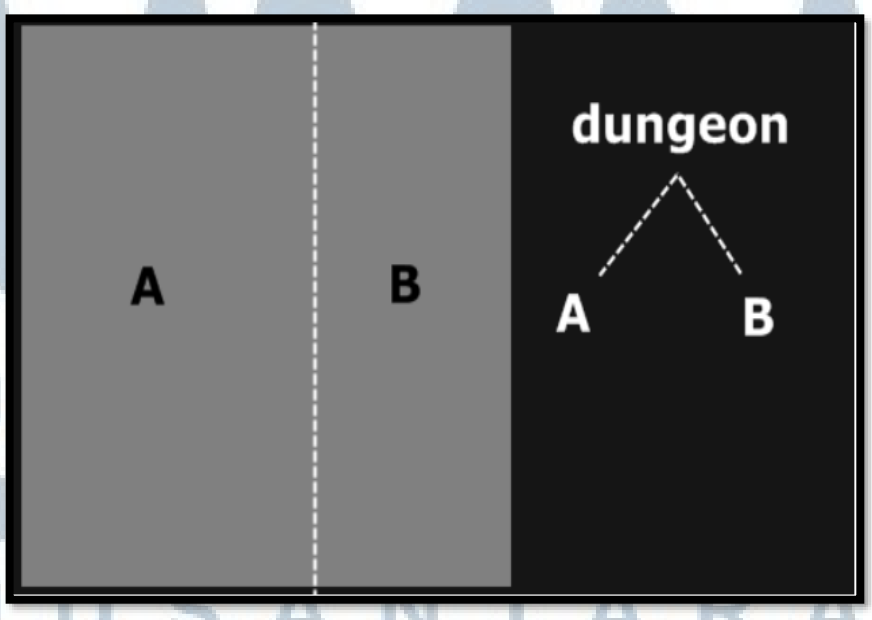


Gambar 2. 3 *Attempt* Penempatan Ruangan Sebanyak 10 Kali Nystrom (2014)



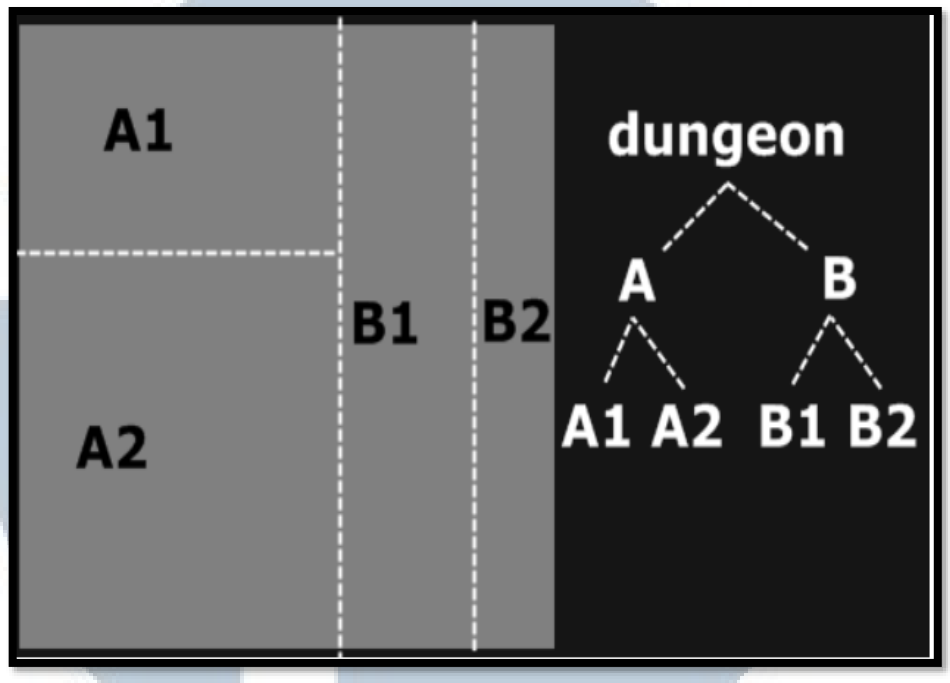
Gambar 2. 4 *Attempt* Penempatan Ruangan Sebanyak 471 kali Nystrom (2014)

2. *BSP Rooms* mengimplementasikan *binary space partitioning,* dimana *graph* akan berubah atau dianggap sebagai daun, dimana masing masing akan menyimpan struktur *tree* dengan seluruh *graph* sebagai *root*. Tiap *leaf* kemudian membelah dirinya sendiri secara vertikal atau horizontal (Baron, 2017).



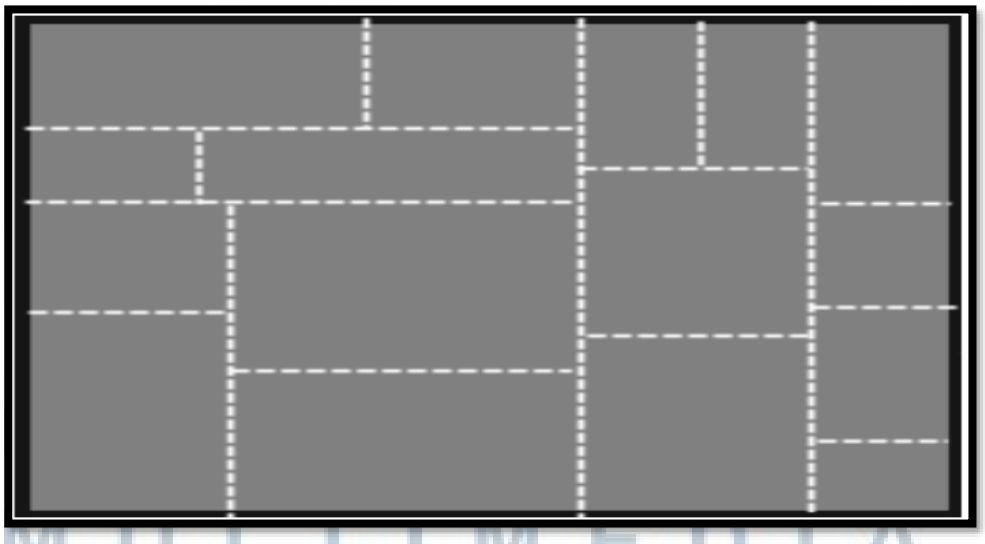
Gambar 2. 5 Tahap Awal BSP Algorithm Roguebasin (2012)

Kemudian melakukan hal yang sama ke *sub child* A dan B.



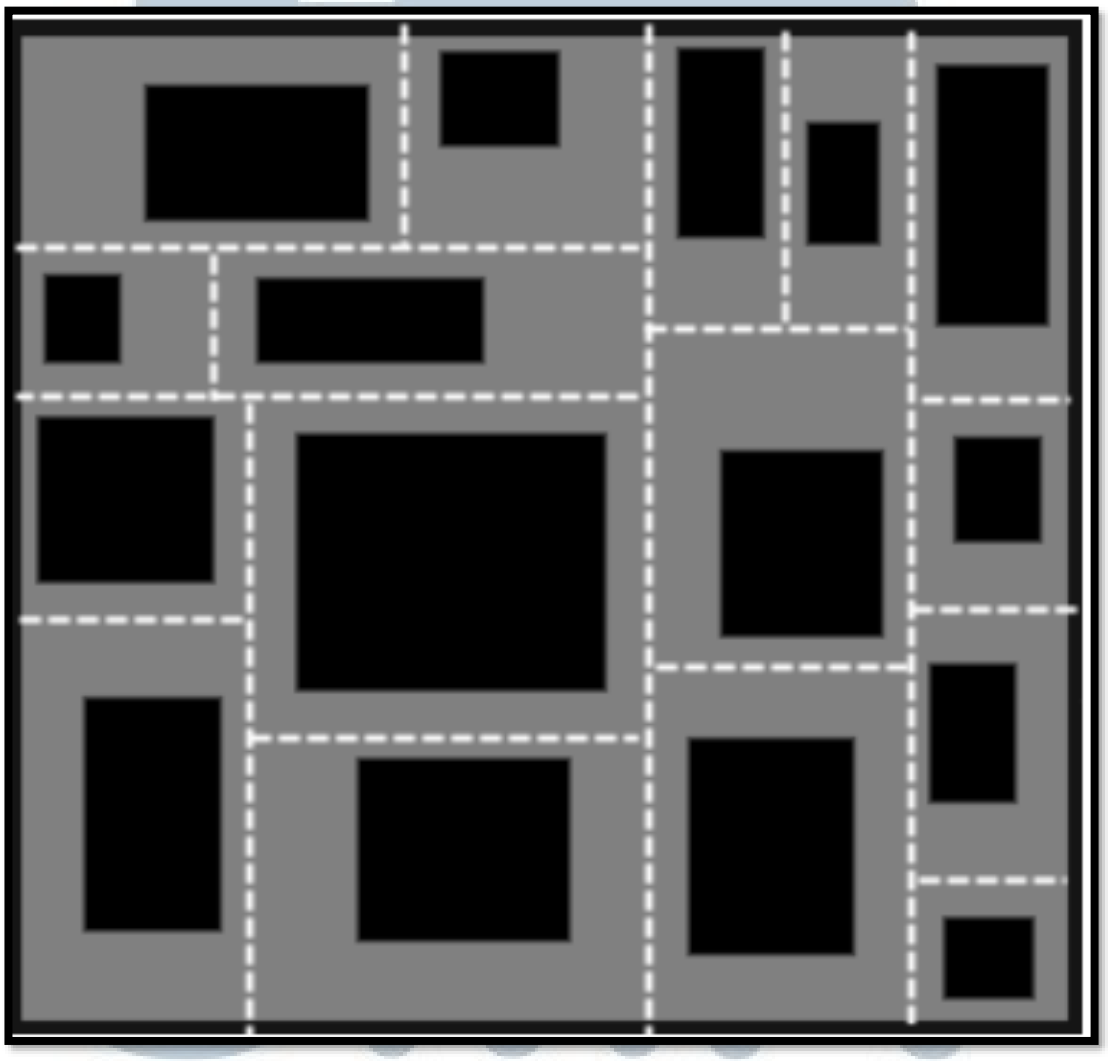
Gambar 2. 6 Tahap Kedua BSP Roguebasin (2012)

Dimana proses ini akan terus berjalan sampai batas *sub child* yang ditentukan.



Gambar 2. 7 Loop Terakhir BSP Roguebasin (2012)

Lalu setelah penempatan ruangan selesai dipilih baru dibuatlah ruangan. Ruangan yang dibuat akan berada di setiap *cell* dan ukuran ruangan akan dipilih secara *random* dengan konstrain tidak boleh melebihi ukuran *cell* dan minimum ukuran sesuai dengan apa yang telah dibuat (Baron, 2017).



Gambar 2. 8 Hasil Akhir BSP Roguebasin (2012).

2.6.2 Corridor-Generating Algorithms

Setelah *graph* yang berisi ruangan dibuat maka akan dibuatlah *corridors* untuk menyambungkan satu ruangan dengan ruangan lain (Baron, 2017) .

1. *Random Point Connect.* Algoritma ini mengakses *list* yang diperoleh pembuatan ruangan sebelumnya dan mengkoneksikan dua ruangan satu per satu. Algoritma ini mengkoneksikan dua ruangan secara *brute force*. Tergantung bagaimana *array* dibuat di algoritma pembuatan ruangan maka hasil dari algoritma juga berbeda. Untuk algoritma ruangan *Random Room Placement* dimana hasil *array* dari algoritma ini adalah ruangan yang berhasil ditempati di *graph* maka penempatan koridor ada kemungkinan akan bertabrakan dengan ruangan lain, jika ini terjadi maka ruangan tersebut akan di-*override* sehingga jalur melalui ruangan yang ditabrak. Namun untuk BSP yang menggunakan struktur *tree* tak akan terjadi tabrakan karena setiap kumpulan *cell* di array pastinya berdempetan dengan satu yang lain (Baron, 2017).
2. *Drunkard's Walk.* Mirip dengan *Random Point Connect* namun algoritma ini mulai dari *vertex* pertama kemudian mencari jalan menuju *vertex* kedua, tak peduli jaraknya negatif atau positif. Algoritma ini mengakses *array* ruangan dan menyambungkan dua pasangan ruangan, *border* untuk setiap ruangan akan di-pilih secara *random*, setelah ruangan dipilih *border*nya maka *corridors* akan mulai dibuat dari ruangan pertama kemudian berjalan ke *border* ruangan kedua sesuai dengan target x dan y, secara inkremental (dimana inkremen bisa sejumlah 2 atau 6 kali) masing masing secara horizontal atau vertikal. Jika koridor menabrak ujung dari *graph* maka jalur *corridors* akan secara otomatis di *invert*.

3. *BSP Corridors*. Algoritma ini harus implementasikan dengan algoritma *BSP Room placement*. Mirip dengan algoritma *corridors* lainnya namun setiap pasangan ruangan dipilih, diambil dari *leaf object* terendah.

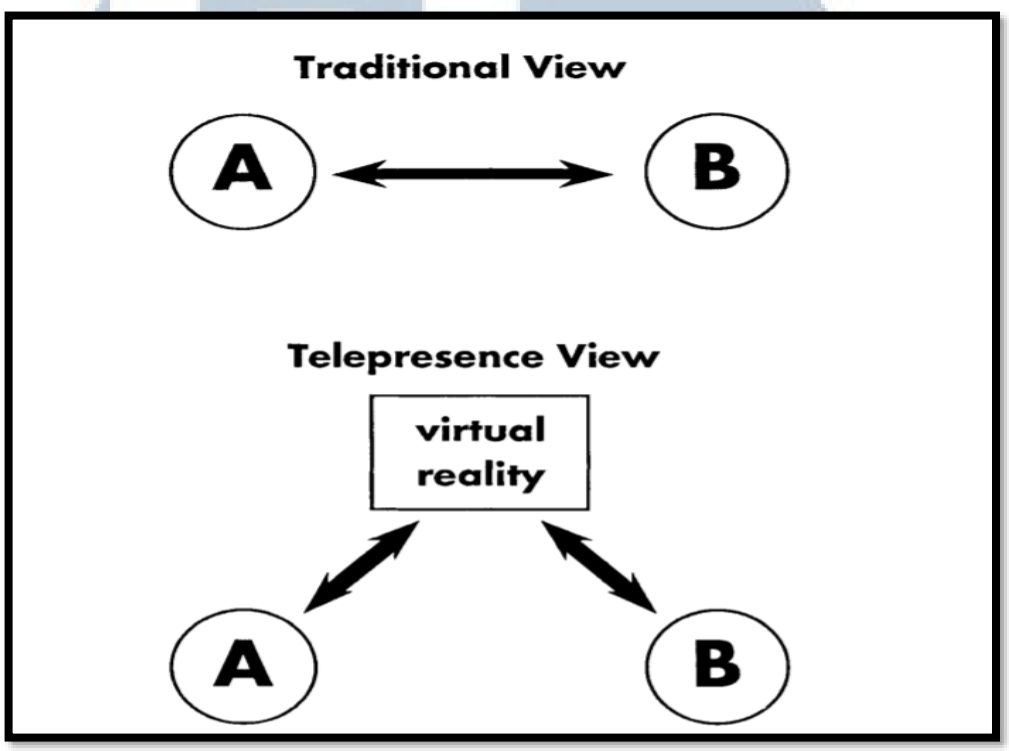
2.7 Virtual Reality

Virtual reality adalah cara menggabungkan dunia virtual dengan dunia asli dimana dunia virtual terasa seakan-akan nyata. Hal ini bisa dicapai dengan adanya teknologi *virtual reality* seperti Oculus Rift dan HTC Vive dan dengan bantuan, *motion capture*, *wireless controller*, dan lain lain untuk lebih *immerse* pemain ke dalam dunia virtual. Steuer (1992) mengatakan banyak sekali definisi mengenai apa itu *virtual reality*, Steuer sendiri mendefinisikan *virtual reality* sebagai *presence*.

Presence adalah suatu kondisi dimana seseorang merasa ia berada di sebuah lingkungan. Banyak hal yang berkontribusi kepada hal ini, dari input indra, persepsi, mental proses, dan pengalaman masa lalu (Steuer, 1992). Ketika seseorang menggunakan *virtual reality* orang itu akan dipaksa untuk merasakan dua lingkungan yang berbeda, yaitu lingkungan nyata dan lingkungan virtual. Ketika menggunakan sebuah teknologi *virtual reality* seseorang akan lebih mempersepsikan dunia virtual dibanding dengan dunia nyata, fenomena ini dinamakan sebagai *telepresence.*

Telepresence didefinisikan sebagai kondisi dimana seseorang lebih merasa berada di dalam suatu lingkungan yang dibuat (*virtual)* dibanding dengan linkungan nyata. Jadi dengan menggunakan konsep *telepresence* bisa didefinisikan

bahwa *virtual reality* adalah sebuah lingkungan buatan dimana seseorang mengalami *telepresence* (Steuer, 1992).



Gambar 2. 9 Gambaran *Telepresence* (Steuer, 1992)

2.8 Game User Experience Satisfaction Scale

GUESS adalah skala *game* yang divalidasi secara psikologis dan komprehensif yang memiliki sembilan *subscales* dan 55 *items* totalnya (Phan, 2017). *GUESS* dimaksudkan untuk segala *playtesting* dan evaluasi *game*. *GUESS* dirancang dan divalidasi berdasarkan penilaian dari 450 *video game* unik dari segala jenis. Jadi dapat aplikasikan ke banyak tipe *video game* untuk melihat aspek apa saja yang berkontribusi terhadap kepuasan pemain.

GUESS didasarkan oleh tujuh *point* skala Likert dengan *respond anchor* pada setiap point (contoh 1 = *Strongly Disagree*, 5 = *Somewhat Agree*, 7 = *Strongly*

Agree). GUESS memiliki 55 *statements* atau *items* dan sembilan *subscales* atau *dimension* yaitu: Usability/Playability, Narratives, Play Engrossment, Enjoyment, Creative Freedom, Audi Aesthetics, Personal Gratification, Social Connectivity, dan Visual Aesthetics.

Nilai dari semua pertanyaan di dimensi yang sama akan dihitung nilai rata-ratanya untuk mendapatkan nilai *subscales* untuk setiap responden dan nilai tingkat kepuasan dapat diperoleh dengan memghitung nilai rata-rata dari semua *sub-scale* Contoh hasil dari penggunaannya pernah dilakukan oleh Mikki Phan (2016) sendiri dengan Metodologi *exploratory factor analysis* (EFA), yaitu sebuah model struktur terdiri dari satu atau lebih variabel, dengan 629 peserta dan *confirmatory factor analysis* (CFA), sebuah instrumen pengujian, dengan 771 peserta.

Variable	EFA (%)	CFA (%)
Last time played		
Today	25.4	25.7
Yesterday	25.9	31.6
Last week	30.0	27.4
Last month	14.3	11.7
About 2–3 months ago	4.3	3.6
Total time spent playing		
10–19 hr	14.8	13.2
20–39 hr	21.0	20.0
40–79 hr	20.8	19.1
80–120 hr	12.1	10.9
More than 120 hr	31.3	36.8
Gaming device used		
Computer device (e.g., laptop, desktop)	36.6	42.5
Console device (e.g., Xbox 360, Nintendo Wii)	48.5	41.1
Handheld gaming device (e.g., Game Boy Advance)	5.1	4.0
Mobile device (e.g., smartphone, tablet)	9.4	12.3
Other (e.g., arcade)	0.5	0.0
Overall satisfaction level		
Extremely dissatisfied	0.5	0.3
Dissatisfied	0.3	0.4
Somewhat dissatisfied	0.8	0.3
Neither satisfied nor dissatisfied	1.3	1.0
Somewhat satisfied	6.2	4.7
Satisfied	43.6	44.9
Extremely satisfied	47.4	48.5

Note. EFA = exploratory factor analysis; CFA = confirmatory factor analysis.

Gambar 2. 10 Pengujian *GUESS* dari *The Development and Validation of the GUESS (*Phan, 2016)

	Statement	Strongly Disagree	Disagree	Somewhat Disagree	Neither Agree nor Disagree	Somewhat Agree	Agree	Strongly Agree	N/A
1	I think it is easy to learn how to play the game.								
2	I cannot tell that I am getting tired while playing the game.								
3	I am captivated by the game's story from the beginning.								
4	I am in suspense about whether I will succeed in the game.								
5	I feel the game allows me to be imaginative.								
6	I think the game is fun.								
7	I enjoy the sound effects in the game.								
8	I find the controls of the game to be straightforward.								
9	I think the characters in the game are well developed.								
10	I find the game supports social interaction (e.g., chat) between players.								

Gambar 2. 11 Kuesioner *GUESS* Oleh Mikki Phan (Phan, 2017)

GUESS dibuat dengan tujuan untuk evaluasi sebuah *game,* melihat nilai aspek yang berkontribusi terhadap *user satisfaction* dan sebagai alat untuk meneliti *gaming experience* pemain. Sembilan *scales* sudah divalidasi mengefek nilai dari kepuasan pemain ketika memainkan *game* (Phan, 2016). Namun jika sebuah *game* tidak memiliki salah satu aspek dari 9 *subscales* ini, maka *subscales* tersebut boleh dihilangkan (Phan, 2016).

2.9 Evaluating PCG

Berikut adalah alasan-alasan mengapa PCG perlu divalidasi, menurut Noor Shaker, dkk (2016) :

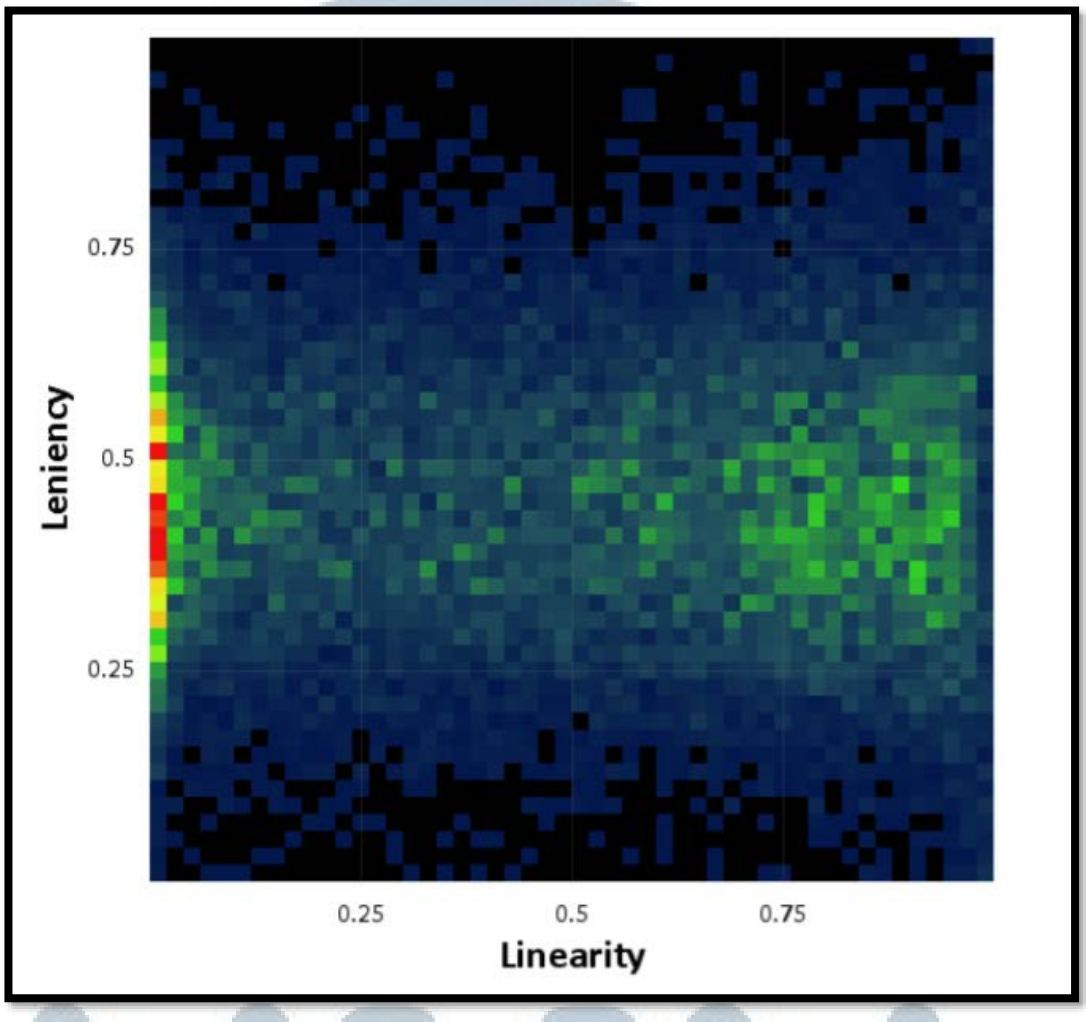
1. Untuk lebih memahami kapabilitas PCG.
2. Untuk memastikan konten ter-*generate.*
3. Untuk lebih mempermudah perubahan/modifikasi dalam iterasi selanjutnya
4. Dapat membandingkan konten *generator* dengan satu sama lain.

Salah satu cara mengevaluasi konten *generator* adalah dengan *Bottom-up evaluation via players* (Shaker dkk., 2016) yaitu dengan menanyakan pemain secara langsung. Salah satu kuesioner yang dapat digunakan untuk evaluasi PCG di dalam *game* adalah *The Game Experience Questionnaire* (Shaker dkk., 2016) oleh Ijsselsteijn de Kort dan Poels (2013). Kuesioner ini menggunakan skala dan memiliki lima *anchor* untuk *respond* dan memiliki komponen berupa *Immersion, Flow, Competence, Positive,* dan *Negative Affect*, dimana nilai dari masing masing komponen adalah rata-rata dari total skala untuk masing masing pertanyaan yang berhubungan dengan komponen. *Game Experience Questionnaire* sudah pernah di-aplikasikan ke beberapa riset yang meng-investigasi pengukuran implisit dan objektif *gaming experience* (Phan, 2016).

Cara lain untuk mengevaluasi PCG adalah dengan *Top-down evaluation via expressivity measures* dimana untuk melakukan evaluasi kualitas sebuah PCG adalah dengan sekedar melihat konten yang di-*generate* dan mengevaluasinya. Namun jika konten yang di-*generate* berjumlah banyak maka tak mungkin untuk melihatnya semua, solusinya adalah dengan melakukan evaluasi *expressive range*

dari sebuah *generator* (Shaker dkk., 2016). *Expressive range* adalah sebuah ruang potensi *level* sebuah *generator* dapat men-*generate* termasuk seberapa *bias generator* terhadap pembuatan suatu konten tertentu (Whitehead, 2011). Evaluasi ini dapat dilakukan dengan memilih metrik dan konten mana yang bisa dievaluasi dan menggunakan metrik tersebut sebagai pedoman untuk men-*generate space* kemungkinan pembuatan konten (Shaker dkk., 2016). Kuantitas konten yang banyak kemudian di-*generate* dan dievaluasi sesuai dengan metrik yang didefinisikan kemudian di *plotted* di sebuah *heatmap. Heatmap* ini dapat memperlihat seberapa *bias* hasil konten di sebuah *generator* dan komperasi tiap set *heatmap* dapat menyimpulkan se-kontrol apa *generator* (Shaker dkk., 2016).

Evaluasi dengan menggunakan *Bottom-up evaluation* dan *Top-down evaluation* boleh dilakukan (Shaker dkk., 2016). Untuk evaluasi konten *game space* akan digunakan teknik *Top-down evaluation* karena konten yang di-*generate* sudah dalam *graph* sehingga bisa ditentukan metrik evaluasinya kemudian mengubah set konten yang dibuatnya menjadi *heatmap* kemudian meng-*evaluasinya*. Untuk metrik yang akan digunakan akan tergantung kepada konten yang di-*generate*. Gambar 2.12 adalah contoh *heatmap* sebuah konten *2D platforming levels* yang menggunakan metrik *linearity* dan *leniency* dalam evaluasinya, namun kedua metrik ini tidak tentu dapat digunakan untuk konten lain (Shaker dkk., 2016).



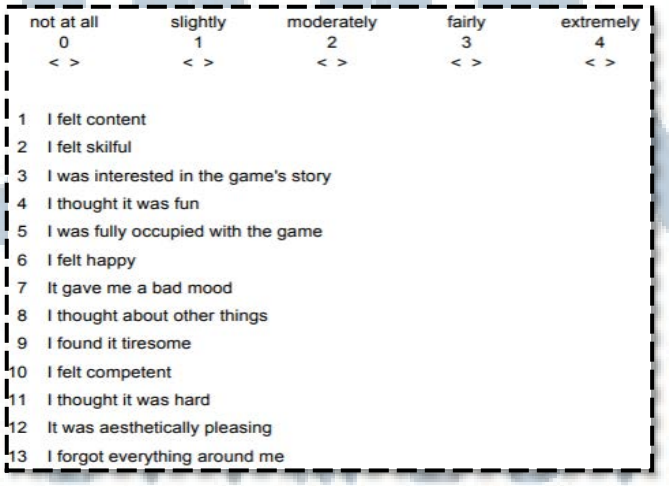
Gambar 2.12 *Expressive range* dari sebuah *Launchpad level generator* (Shaker, 2016)

Shaker, dkk (2016) mengatakan bahwa peraturan yang penting ketika memiliki metrik yang akan digunakan adalah memilih metrik yang jauh dari input parameter sistemnya. Tujuan dilakukannya *expressive range* untuk evaluasi adalah untuk mengerti *properties* dari sebuah *generative* sistem. Memilih metrik yang sama dengan input parameter *generator* hanya akan menunjukkan bahwa algoritma yang dipakai bekerja sesuai ekspetasi, sehingga tidak dapat memberi pencerahan kepada *unexpected behaviour* atau *output* yang mengejutkan. Metriks yang akan

digunakan untuk menghitung tingkat kualitas PCG dari *game space* berupa *density* dan *leniency*.

1. *Leniency* adalah nilai untuk melihat seberapa mudah *game* (Shaker, 2016). Untuk konteks PCG *game space, leniency* menghitung seberapa mudahnya bernavigasi di dalam *maze.* Ini dapat dilihat dari seberapa banyak *corridor* yang dipunyai oleh suatu ruangan. Lebih banyak pilihan, lebih sulit bernavigasi, karena pemain harus memilih jalan dan jalan yang dipilih bisa saja salah.
2. *Density* untuk konteks *game space maze* adalah seberapa banyak ruangan yang akan digunakan oleh *generator* untuk membuat suatu *maze* dari luas ruangan yang diberikan. Britton Horn (2014) dalam penelitian PCGnya, menggunakan nilai *density* untuk mengavaluasi PCG dari *game* 2D *platform* buatannya.

Untuk konten *narrative* akan dievaluasi menggunakan *Game Experience Questionnaire* yang memiliki tujuh komponen, nilai dari setiap komponen adalah hasil rata-rata dari nilai setiap pertanyaan yang berhubungan dengan komponen tersebut.



Gambar 2.13 Skala dan Beberapa Pertanyaan *Gaming Experience Questionnaire* (Ijsselsteijn, 2013)

Game Experience Questionnaire sudah pernah digunakan untuk berbagai riset, salah satunya oleh Tom Gross, dkk (2009) dalam buku *Human-Computer-INTERACT* 2009 untuk mengukur pengalaman pemain dalam *board game* yang di *balanced* dan di *unbalanced.*

Kuesioner ini memiliki 3 modul yaitu *Core Questionnaire, Social Presence,* dan *Post-game* setiap modul ini memiliki fungsi masing masing *Core Questionnaire* bertujuan untuk menilai *experience game* melalui tujuh askpek yaitu: *Immersion, Flow, Competence, Positive and Negative Affect, Tension,* dan *Challenge* (Ijsselsteijn, 2013). Modul *Social Presence* untuk menilai interaksi antara pemain dan modul *Post-game* untuk melihat dampak setelah bermain (Ijsselsteijn, 2013). Modul *Core Questionnaire* aspek Immersion akan digunakan untuk meng-evaluasi *PCG Narrative* dikarenakan cerita pada *game* masuk kedalam aspek *Immersion* (Ijsselestenijn, 2013).

2.10 Formal Elements and Dramatic Elements

Formal elements membantu dalam perancangan struktur *game*, hubungan antara *formal elements* inilah yang membuat suatu *game* (Nacke, 2014). Berikut adalah *formal elements* dari *game* menurut Lennart Nacke (2014).

1. *Players: Player* atau pemain adalah peserta dari *game* yang berinteraksi dengan *game*. Tiap *game* memiliki cara berinteraksi yang berbeda, interaksi inilah yang menentukan bagaimana pemain memainkan *game* dan menentukan peran pemain di dalam *game.*
2. *Objectives: Objectives* atau objektif adalah masalah yang dapat diselesaikan oleh pemain di dalam *game* sehingga mendorong motivasi pemain untuk

bermain *game*. Contoh dari objektif adalah mendapatkan *point* terbanyak pada akhir permainan atau berhasil menjawab semua soal pada *game* kuis.

3. *Procedures: Procedures* atau prosedur adalah aksi atau metode bermain yang diperbolehkan oleh *game's rules.* Prosedur bisa disebut sebagai instruksi yang bisa dilakukan ketika *game* dimainkan.
4. *Rules: Rules* atau peraturan adalah objek dan konsep dari *game*. Peraturan memberitahu apa yang bisa dilakukan dan tidak bisa dilakukan oleh pemain.
5. *Resources: Resources* adalah objek *game* yang mempunyai nilai untuk pemain dalam usaha menyelesaikan suatu objektif. *Resources* membantu dalam meraih tujuan untuk pemain.
6. *Conflict: Conflict* atau konflik adalah sesuatu yang muncul dari hasil *procedures* dan *rules* di dalam *game* yang mencegah pemain dalam meraih tujuan akhir. Objektif biasanya memandu pemain kepada konflik ini.
7. *Boundaries:* Limitasi dari *game* biasanya terkait dengan *game space* suatu *game.*
8. *Outcome: Outcome* adalah resolusi dari *game* biasanya dapat dihitung (seperti halnya *point*) dan tidak rata (seperti halnya pemain menang).

Formal elements dari *game* dapat membuat suatu *game* yang berfungsi, namun tak menarik dengan sendirinya. *Dramatic elements* adalah kumpulan *elements* yang bertujuan untuk menarik perhatian pemain dan memberikan suatu pengalaman yang dapat mengikat ke pemain (Milanetti, 2013). Berikut adalah elemen-elemen dari *Dramatic elements* menurut Lennart Nacke (2014).

1. *Challenge:* Berhubungan dengan keahlian pemain, *Challenge* merupakan rasa kepuasan atau kesenangan atas prestasi yang berhasil didapatkan di dalam *game.*
2. *Play:* Suatu aktivitas yang dibuat untuk meraih rasa kesenangan dan kepuasan. *Play* sendiri memiliki tipe-tipe yang berbeda. Aktivitas ini bisa berupa bermain dengan improvisasi atau bermain berdasarkan peraturan.
3. *Characters: Characters* adalah aktor yang menjalankan cerita pada *game*. Mereka adalah benda yang dapat menimbulkan emosi pada diri pemain.
4. *Premise: Premise* adalah *setting* dari cerita di dalam *game*, *premise*lah yang mendirikan apa yang akan terjadi dan mempersiapkan pemain untuk apa yang akan terjadi dalam *game* sehingga dapat memotivasi pemain.
5. *Story: Story* atau *narrative* adalah cerita pada *game*. *Story* menceritakan apa yang terjadi pada awal *game* sampai dengan akhir *game* dimana pemain memiliki peran dalam cerita dan bertanggung jawab untuk menyelesaikan cerita.
6. *World Building: World building* adalah deskripsi atau pengetahuan *game space* dimana pemain berada. Seperti halnya desain dan karakteristik dari dunia yang berada di dalam *game.*