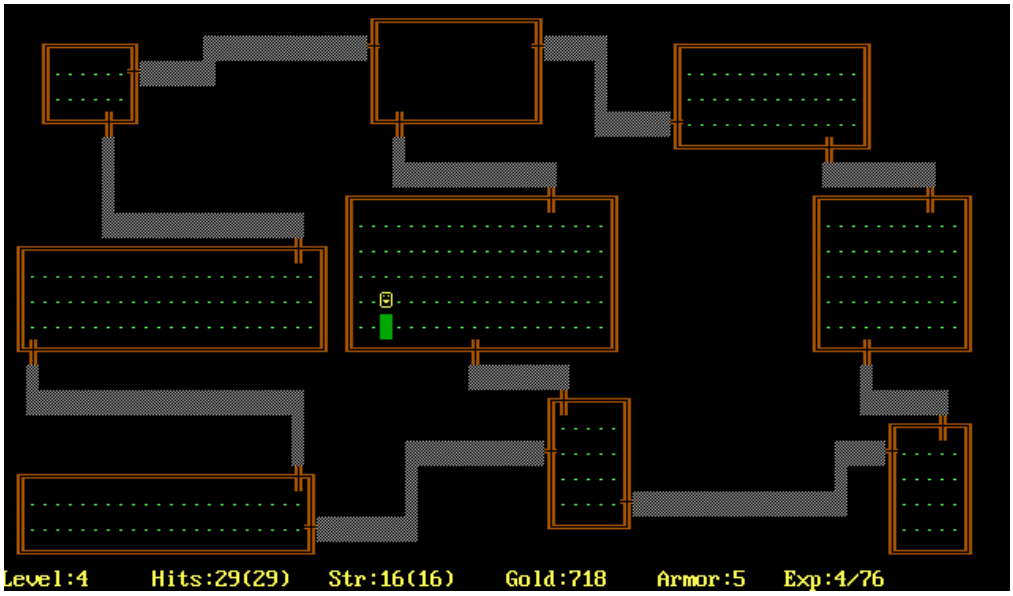
BAB 2
LANDASAN TEORI

2.1 *Procedural Content Generation* (PCG)

Procedural Content Generation atau PCG adalah suatu metode "generatif" yang diterapkan pada suatu gim yang mana gim tersebut mampu untuk membangkitkan sendiri konten di dalamnya [8]. Konten dalam konteks tersebut antara lain *level*, peta, cerita, misi, musik, *item*, dan sebagainya yang merupakan unsur pembangun dalam gim [2].

Beberapa contoh gim yang menggunakan PCG antara lain Rogue, Ancient Domains of Mystery, dan Enter The Gungeon.



Gambar 2.1 Rogue (*video game*)

Gambar 2.2 Ancient Domains of Mystery (*video game*)

Shaker, Togelius, dan Nelson, juga menyebutkan beberapa hal yang termasuk PCG pada bukunya yang berjudul "***Procedural Content Generation in Games***", antara lain:

1. Sebuah alat perangkat lunak yang digunakan untuk membangkitkan *dungeons* pada sebuah gim *action adventure* tanpa *input* dari manusia yang mana setiap kali alat tersebut dijalankan, maka sebuah *level* akan dibangkitkan.
2. Sebuah sistem yang membangkitkan senjata baru pada sebuah gim *space shooter* dengan mengadaptasi kebiasaan sejumlah pemain sehingga dapat membangkitkan sebuah senjata versi hasil evolusi yang lebih menyenangkan untuk digunakan oleh pemain.
3. Piranti tengah atau *framework* untuk sebuah *game engine* yang secara cepat memopulasikan sebuah dunia gim dengan vegetasi secara otomatis.

1) Kriteria Penilaian Solusi PCG

Shaker, Togalius, dan Nelson [2] juga menyebutkan terdapat beberapa kriteria yang dapat diukur dari sebuah solusi PCG, yaitu:

1. *Speed*, yaitu kecepatan pembangkitan konten, baik ketika dalam proses pengembangan maupun ketiga gim sedang berjalan.
2. *Reliability*, yaitu keandalan dalam membangkitkan konten yang sesuai dengan kualitas yang diinginkan. Hal ini bersifat relatif, contohnya sebuah *dungeon* yang tidak memiliki pintu masuk atau keluar termasuk tidak memenuhi kualitas yang diinginkan oleh pengembang gim.
3. *Controllability*, yaitu kemampuan pembangkit konten untuk menerima *input* tambahan baik dari manusia atau dari sebuah algoritma dengan tujuan pembangkit konten masih dapat dikendalikan oleh pengembang gim agar sesuai dengan spesifikasi yang diinginkan.
4. *Expressivity and Diversity*, yaitu kemampuan pembangkit konten untuk menghasilkan konten yang secara relatif tidak terkesan repetitif agar gim tidak membosankan.
5. *Creativity and Believability*, yaitu kriteria konten yang dihasilkan oleh pembangkit konten agar terlihat alami atau dibuat oleh manusia, bukan seperti dibangkitkan oleh suatu algoritma.

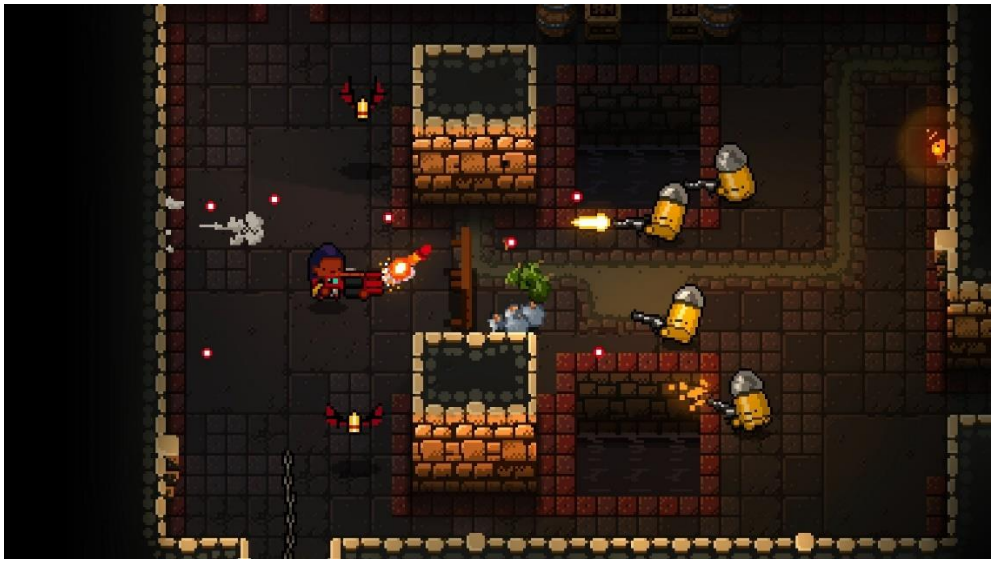
2) Online vs Offline

Menurut Shaker et al [2], PCG dibedakan berdasarkan waktu pembangkitan konten di dalamnya yaitu *Offline* dan *Online*. *Offline* berarti pembangkitan konten terjadi ketika fase pengembangan atau sebelum gim dimulai, sedangkan *Online* berarti pembangkit konten terjadi ketika fase gim sedang berjalan atau dimainkan.

2.2 *Roguelike*

Roguelike adalah sebuah genre gim yang muncul setelah gim berjudul Rogue rilis. Rogue sendiri adalah gim pertama yang menggunakan PCG sebagai pembangkit konten berupa *level* yang mana konten tersebut dibangkitkan secara *random*. Setelah gim Rogue rilis, banyak gim serupa yang memanfaatkan PCG

untuk membangkitkan *level* sehingga terlahirlah genre gim *Roguelike*. Genre ini memiliki ciri khas pada konten yang dibangkitkan secara prosedural, *permanent death*, dan karakter yang bersifat progresif [9].



Gambar 2.3 Enter The Gungeon (*video game*)

Beberapa gim terbaru yang termasuk dalam genre *Roguelike* antara lain Enter The Gungeon, Dead Cells, The Binding of Isaac. Minecraft juga termasuk dalam genre *Roguelike*, namun karena memiliki elemen lain yang lebih kuat, maka *Roguelike* hanya menjadi pelengkap atau sebagai subgenre saja, atau masuk ke dalam kategori lain yang lebih cocok [9].

2.3 *Drunkard's Walk Algorithm*

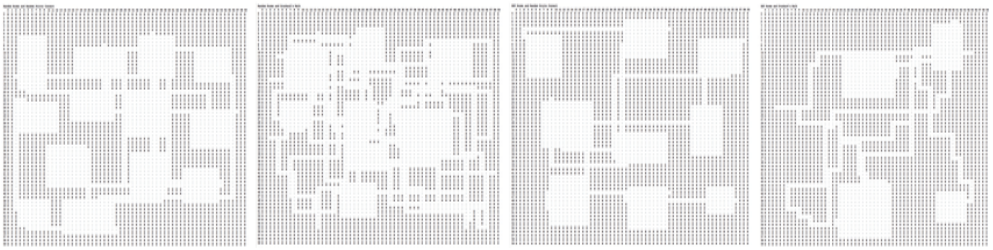
Drunkard's Walk Algorithm atau *random walk* merupakan salah satu algoritma yang digunakan untuk melakukan *generate dungeon* atau *maze*. *Drunkard's Walk Algorithm* bekerja dengan cara memilih satu *point* awal di dalam *grid* dan mengisi *grid* tersebut secara satu arah hingga algoritmanya memilih untuk bergerak menuju arah lain, proses ini diulang terus menurut hingga satu *level* yang diinginkan terbentuk [10]. *Drunkard's Walk* juga bisa digunakan dalam koneksi antar ruang, proses ini bisa dilakukan dengan menyambungkan *vertex border* antar 2 (dua) ruangan yang *random* [11].

$$S_n = \sum_{i=1}^{n} X_i + \cdots + X_n \quad \text{(Rumus 2.1)}$$

Rumus di atas merupakan gambaran proses yang dilakukan oleh *Drunkard's Walk* di mana S_n merupakan jumlah dari *random step*, berturut-turut X_i. Lalu S_n membentuk suatu *random walk* berdasarkan hasil yang didapatkan. *Drunkard's Walk* mendapatkan namanya dari konsep masalah pemabuk berjalan, di mana arah yang ditempuh tidak akan jelas dan meninggalkan jejak yang menghasilkan 2D (dua dimensi) *random walk*.

$$S_{t+i} = S_t + W_t \quad \text{(Rumus 2.2)}$$

Rumus di atas merupakan gambaran *Drunkard's Walk* di mana S_t merupakan posisi saat ini berdasarkan t yang akan ditambahkan dengan probabilitas pergerakan S_t selanjutnya digambarkan dengan W_t. Hasil dari implementasi *Drunkard's Walk* akan berupa *level* berbentuk layaknya *dungeon*, *room*, maupun *hallway* dengan bentuk yang berbeda setiap saat.



Gambar 2.4 PCG Generated Dungeon (*Drunkard's Walk* ke-2 dari kiri)

Gambar di atas merupakan beberapa gambaran implementasi beberapa algoritma PCG dalam *dungeon generation* dalam suatu gim. *Drunkard's Walk* memiliki struktur yang lebih berantakan karena sifatnya yang lebih *random*, dibandingkan dengan beberapa algoritma lain yang digunakan sebagai *generator*.

Pseudocode merupakan cara penulisan program informal yang dapat dibuat dengan kaidah yang ditentukan sendiri. Pseudocode berfungsi sebagai *outline* untuk memahami alur dan logika program sebelum diubah menjadi bahasa pemrograman tertentu. Berikut merupakan pseudocode dari algoritma *Drunkards's Walk*.

$Algoritma\ Random\ Walk$
$\boldsymbol{Input}$: $Jaringan\ kesamaan\ G = (V, E)$;
$Mulai\ node\ n$;
$Restart\ probabilitas\ \alpha$;
$\boldsymbol{Output}$: $Vektor\ stasioner\ Drunkard's\ Walk\ dimulai\ dari\ n$;
(1) $Misalkan\ s\ menjadi\ vektor\ restart\ dengan\ semua\ entry\ diinisiaisasi\ ke\ 0\ kecuali\ 1\ untuk\ entri\ dilambangkan\ dengan\ n$
(2) $Misalkan\ P\ menjadi\ matriks\ adjacency\ (transisi)\ yang\ dinormalisasi\ baris\ yang\ didefinisikan\ oleh\ G$;
(3) $Menginisialisasi\ x \coloneqq s$;
(4) $while\ (x\ belum\ menemukan\ titik\ temu)$
(5) $x \coloneqq \alpha s + (1 - \alpha)P^T x$;
(6) $Output\ x$;

(Algoritma Drunkard's Walk 2-3)

Adapun daftar penjelasan notasi yang digunakan pada pseudocode tersebut yaitu sebagai berikut:

Tabel 2.1 Daftar Notasi Pseudocode Algoritma *Drunkard's Walk*

No.	Simbol	Penjelasan
1.	G	Tidak terarah, sebuah grafik yang berbobot
2.	V	Simpul dalam grafik
3.	E	Tepi dalam grafik

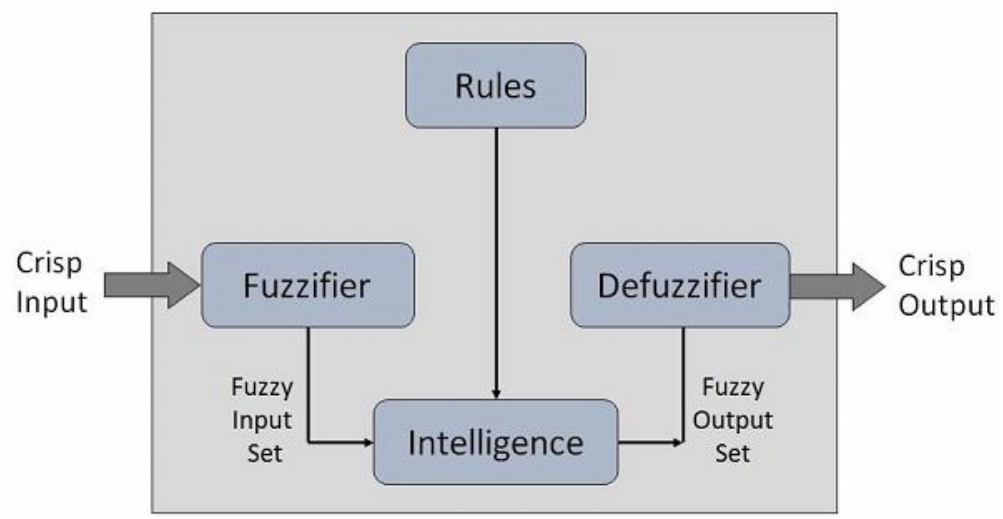
No.	Simbol	Penjelasan
4.	P	Matriks transisi untuk grafik
5.	C	Vektor yang terdiri dari sekelompoke *node*
6.	α	*Restart* probabilitas *Drunkard's Walk*
7.	s	*Restart* vektor untuk sebuah *node*
8.	x	Vektor stasioner dari sebuah *node Drunkard's Walk*

2.4 *Fuzzy Logic*

Fuzzy Logic merupakan bentuk logika bernilai banyak yang memiliki nilai kebenaran variabel dalam bilangan riil antara nol dan satu [12]. *Fuzzy Logic* dikembangkan berdasarkan bahasa manusia (bahasa alami). Tujuannya untuk menjembatani bahasa mesin yang presisi dengan bahasa manusia yang menekankan pada makna atau arti (*significance*). *Fuzzy Logic* umumnya diterapkan pada masalah-masalah yang mengandung unsur ketidakpastian (*uncertainly*), tidak tepat (*imprecise*), *noisy*, dan sebagainya [13].

1) Arsitektur Sistem *Fuzzy Logic*

Pada dasarnya, ada 4 (empat) bagian dalam arsitektur sistem *fuzzy logic*, yaitu sebagai berikut:



Gambar 2.5 Arsitektur *Fuzzy Logic*

A. *Rule Base*

Rule Base berisi semua aturan dan kondisi "if-else" untuk mengontrol pengambilan keputusan. Namun, seiring perkembangan modern, jumlah aturan dalam *rule-base* yang digunakan *fuzzy logic* telah banyak berkurang.

B. *Fuzzifier / Fuzzification*

Fuzzifier atau *Fuzzification* adalah komponen kedua dalam arsitektur *fuzzy logic* dan berguna untuk membantu mengubah *input*. Komponen ini membantu dalam mengonversi angka ekstrem ke himpunan *fuzzy*. Masukan yang ekstrem diukur oleh sensor dan diteruskan ke sistem kontrol untuk diproses. Pada bagian ini digunakan untuk mengubah *input* sistem dan juga membantu dalam membagi sinyal *input* menjadi 5 (lima) *state*:

a) *Large positive*
b) *Medium positive*
c) *Small*
d) *Medium negative*
e) *Large negative*

C. *Intelligence*

Intelligence atau *Inference Engine* membantu dalam menentukan tingkat kecocokan antara *input fuzzy* dan aturan *fuzzy*. Berdasarkan persentase itu

diputuskan aturan mana yang perlu diterapkan. Setelah itu, untuk mengembangkan tindakan kontrol, aturan yang diterapkan akan digabungkan.

D. *Defuzzifier*

Defuzzifier merupakan kebalikan dari proses *Fuzzifier* atau *Fuzzification*. Pada proses *Defuzzifier*, nilai *fuzzy* diubah menjadi nilai ekstrem melalui pemetaan (*mapping*). Ada beberapa metode *Defuzzifier* yang dapat dilakukan, tetapi pemilihan metode yang terbaik didasarkan sesuai *input*.

2) Kelebihan dan Kekurangan *Fuzzy Logic*

Berikut merupakan kelebihan dan kekurangan algoritma *fuzzy logic*:

A. Kelebihan

a) Sistem yang dibangun dengan *fuzzy logic* dapat bekerja dengan berbagai jenis *input*, baik yang tidak presisi, terdistorsi, maupun mengandung banyak informasi yang *noise*.
b) Rancangan sistem *fuzzy logic* cukup mudah dan dapat dimengerti.
c) *Fuzzy logic* hadir dengan konsep matematika dari teori himpunan dan penalaran yang cukup sederhana.
d) Dapat memberikan solusi yang sangat efisien untuk masalah kompleks di semua bidang kehidupan karena menyerupai penalaran dan pengambil keputusan manusia.
e) *Fuzzy logic* dapat dikodekan menggunakan lebih sedikit data, sehingga tidak menempati ruang memori yang besar.
f) Algoritma ini fleksibel dan aturannya dapat dimodifikasi.

2.5 *Game User Satisfaction Scale* (GUESS)

GUESS merupakan suatu metode yang digunakan untuk menjadi tolak ukur kepuasan pemain dalam bermain *video game* dengan cara mengadakan 55 (lima puluh lima) pertanyaan dengan 9 (sembilan) *construct* sebagai penilaiannya. Berikut merupakan 9 (sembilan) *construct* yang terdapat pada GUESS [14].

***Usability / Playability*:** Seberapa mudah gim bisa dimainkan dengan benar tanpa adanya suatu halangan fisik berupa kontrol maupun halangan visual berupa *User Interface* (UI) yang berantakan.

a) ***Narrative*:** Secara lateral merupakan cerita yang ada di dalam gim. Cerita diukur dengan cara melihat kemampuannya untuk menggugah minat dan membawa emosi pemain selama bermain gim.

b) ***Play Engrossment*:** Tolak ukur sejauh mana gim dapat terus menarik perhatian dan minat pemain untuk memainkan gim tersebut.

c) ***Enjoyment*:** Berbeda dengan *play engrossment*, *enjoyment* merupakan seberapa menyenangkan gim yang dimainkan menurut pemain.

d) ***Creative Freedom*:** Seberapa besar kreativitas dan kebebasan yang diberikan kepada pemain.

e) ***Audio Aesthetics*:** Seberapa memadai audio yang digunakan pada gim, seperti *sound effect* dan *soundtrack* yang digunakan.

f) ***Personal Gratification*:** Aspek pada gim yang memberikan rasa kepuasan dan motivasi kepada pemain.

g) ***Social Connectivity*:** Pengukuran fasilitas pada gim berupa *tools* atau fitur yang mendukung adanya interaksi antar pemain.

h) ***Visual Aesthetics*:** Seberapa bagus sebuah grafik yang digunakan pada gim menurut para pemain.

Pada kenyataannya tidak semua orang akan rela untuk melakukan pengisian terhadap 55 (lima puluh lima) kuesioner, oleh karena itu diadakan perbaikan pada GUESS-18 yang hanya memiliki 18 (delapan belas) utilitas pertanyaan berdasarkan 9 (sembilan) *construct* GUESS versi sebelumnya [14]. GUESS versi pertama membutuhkan setidaknya 10-15 menit untuk menyelesaikan sedangkan GUESS-18 membutuhkan setidaknya 3-5 menit untuk menyelesaikannya. GUESS-18 memberikan hasil yang cukup kuat dan tidak kalah akurat terhadap GUESS versi pertama yang digunakan sebagai analisis data. Berikut merupakan 7 (tujuh) *level* penilaian *Likert* yang digunakan dalam GUESS:

Tabel 2.2 Penilaian *Likert*

Penilaian *Likert*	**Nilai**
Sangat Setuju	7
Setuju	6
Cukup Setuju	5
Netral	4
Cukup Tidak Setuju	3
Tidak Setuju	2
Sangat Tidak Setuju	1

Setiap pertanyaan mempunyai jumlah berdasarkan dari nilai yang dipilih oleh responden, yang nilai tersebut dapat dihitung menjadi nilai rata-rata dengan rumus sebagai berikut:

$$Rata - rata\ (\%) = ((jumlah\ sangat\ tidak\ setuju * 1) + (jumlah\ tidak\ setuju * 2) + (jumlah\ cukup\ tidak\ setuju * 3) + (jumlah\ netral * 4) + (jumlah\ cukup\ setuju * 5) + (jumlah\ setuju * 6) + (jumlah\ sangat\ setuju * 7))$$

$$\frac{}{(jumlah\ responden * skala\ tertinggi)} * 100\%$$

(Rumus Rata-rata 2.3)

Kategori predikat penilaian nilai persentase GUESS-18 yang didapatkan dari setiap pertanyaan, maupun yang didapatkan dari rata-rata keseluruhan bisa dibagi menjadi beberapa tingkatan yaitu sebagai berikut:

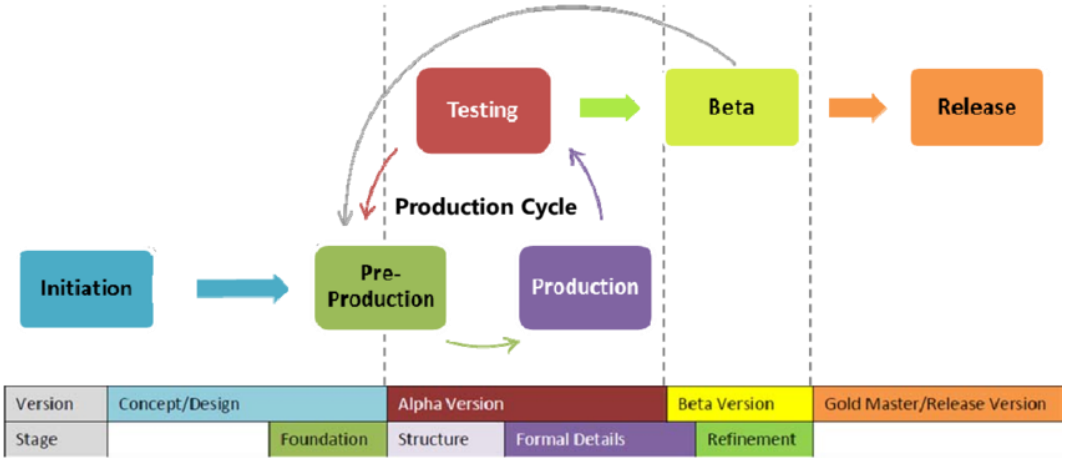
Tabel 2.3 Predikat Nilai dan Persentase GUESS-18

Predikat Nilai	Persentase
Sangat Buruk	0% – 14%
Buruk	15% – 28%
Cukup Buruk	29% – 42%
Netral	43% – 56%
Cukup Baik	57% – 70%
Baik	71% – 84%
Sangat Baik	≥ 85%

Hasil interval predikat 14% didapatkan dari hasil perhitungan 100% dibagi dengan skala terbesar.

2.6 Game Development Life Cycle (GDLC)

Game Development Life Cycle (GDLC) diajukan untuk menjawab tiga pertanyaan penelitian: langkah-langkah apa yang diperlukan untuk mengembangkan sebuah gim, kriteria kualitas apa yang harus saya pertimbangkan dalam setiap langkah, dan bagaimana cara membuat gim yang berkualitas [7]. GDLC yang diusulkan terdiri dari 6 (enam) fase pengembangan seperti pada Gambar 2.1 di bawah.



Gambar 1.1 Game Development Life Cycle

a) Initiation

Langkah pertama yang harus dilakukan dalam membuat sebuah gim adalah dengan melakukan studi literatur sebagai tahapan pendalaman materi, perumusan masalah dan teori yang berkaitan dengan permasalahan serta untuk mendalami materi yang terkait dengan *Procedural Content Generation* (PCG) menggunakan algoritma *Drunkard's Walk* dengan *Fuzzy Logic* pada gim bergenre *Roguelike*.

b) Pre-production

Pre-production atau Pra-produksi adalah salah satu fase utama dan terpenting dalam siklus produksi. Pra-produksi melibatkan pembuatan dan revisi desain gim dan pembuatan *prototype* gim. Fokus pada desain gim menentukan *genre* gim, *gameplay*, *mechanic*, alur cerita, karakter, tantangan, faktor kesenangan, aspek teknis, dan dokumentasi elemennya dalam dokumen desain gim. Setelah dokumen desain gim dibuat, bentuk *prototype* dibuat untuk menilai desain gim dan keseluruhan idenya. Pada iterasi pertama siklus produksi, yang dibuat *prototype* adalah fondasi dan struktur, sedangkan iterasi berikutnya, *prototype* terkait yang akan disempurnakan adalah detail dan perbaikan formal.

Dasar dari *prototype* pertama, terkait dengan kriteria kualitas kesenangan. Dasar ini digunakan untuk menunjukkan *mockup gameplay* inti dan kemampuan gim. Kriteria kualitas kesenangan diuji melalui kuesioner atau diskusi. Struktur dalam perbaikan dasar, akan terkait dengan kriteria kualitas kesenangan dan

fungsional pada gim. Ini merupakan ciri utama dari struktur yang dapat menunjukkan baik itu *gameplay* inti dari gim dan mekanika seperti aritmetika, logika, dan aturan di dalam gim. Kuesioner dan diskusi digunakan untuk menguji kriteria kualitas kesenangan. Kemudian kriteria kualitas fungsional diuji melalui *playtesting*, di mana penguji diberikan beberapa tugas dan tujuan yang ingin dicapai menurut pengujian skenario. Pra-produksi berakhir saat revisi atau perubahan desain gim telah disetujui dan didokumentasikan.

c) Production

Production adalah proses inti yang membahas seputar pembuatan aset, pembuatan program, dan integrasi kedua elemen. Terkait *prototype* dalam fase ini adalah detail formal dan perbaikan. Detail formal adalah struktur yang disempurnakan dengan lebih banyak mekanik dan aset yang lengkap. *Production* merupakan kegiatan yang terkait dengan pembuatan dan penyempurnaan detail formal menyeimbangkan, menambahkan fitur yang baru, meningkatkan kinerja, dan memperbaiki *bug* (terkait dengan fungsional dan penyelesaian internal kriteria kualitas).

Penyeimbangan gim berarti penyesuaian terkait dengan kesulitan gim untuk membuat gim tersebut memiliki kesulitan yang sesuai. Perbaikan adalah *prototype* lengkap yang merupakan subjek memoleskan gim. Kriteria kualitas pada tahap ini terkait gim harus menyenangkan dan mudah diakses. Kegiatan selama penyempurnaan diarahkan untuk membuat gim lebih menyenangkan, menantang, dan lebih mudah dipahami. Hanya perubahan kecil diperbolehkan dalam fase ini.

d) Testing

Testing dalam konteks ini berarti pengujian internal yang dilakukan untuk menguji fungsi operasional dan kemampuan bermain gim. Metode pengujian khusus untuk setiap tahap *prototype*. *Testing* detail formal dilakukan dengan menggunakan *playtest* untuk menilai fungsionalitas fitur dan kesulitan pemain (terkait dengan keseimbangan gim).

Metode untuk pengujian kriteria kualitas fungsional melalui fitur *playtesting* untuk menguji kualitas lengkap secara internal bisa dilakukan melalui *playtesting* secara bersamaan dengan uji fungsionalitas. Saat penguji menemukan *bug*, celah, atau jalan buntu selama *playtesting*, penyebabnya dan skenario untuk mereproduksi kesalahan yang diperlukan didokumentasikan dan dianalisis. Untuk menguji keseimbangan kriteria kualitas, *playtesting* dengan beberapa perbedaan perbaikan digunakan untuk mengategorikan apakah suatu perbaikan terlalu sulit, terlalu mudah, atau tidak diperlukan.

Pengujian perbaikan berhubungan dengan kualitas kesenangan dan kriteria kualitas aksesibilitas. Dalam pengujian perbaikan, kesenangan diuji melalui tes bermain dan umpan balik langsung dari sesama pengembang, baik itu membosankan, membuat frustrasi, dan lain-lain. Aksesibilitas dapat diuji melalui mengamati perilaku penguji. Jika penguji merasa kesulitan untuk memainkan dan memahami gim, itu berarti bahwa gim belum cukup mudah diakses. *Output* dari pengujian adalah laporan *bug*, permintaan perubahan, dan keputusan pembangunan. Hasilnya akan menentukan apakah sudah waktunya untuk maju ke fase berikutnya (*Beta*) atau mengulangi siklus produksi.

e) Beta

Beta adalah fase untuk melakukan pengujian oleh pihak ketiga atau eksternal yang disebut pengujian beta atau *beta testing*. Pengujian beta masih menggunakan metode pengujian yang sama dengan metode pengujian sebelumnya, karena *prototype* terkait dalam pengujian hadir dalam dua jenis: beta tertutup dan beta terbuka. Beta tertutup hanya mengizinkan individu yang diundang untuk menjadi peserta, sedangkan beta terbuka memungkinkan siapa saja yang mendaftar menjadi peserta. Kriteria kualitas dalam versi beta terkait erat dengan tahap *prototype* saat ini.

Dalam pengujian detail formal, penguji diminta untuk menemukan *bug* (terkait dengan kriteria kualitas fungsional dan lengkap secara internal). Dalam uji perbaikan, penguji diberikan kebebasan lebih untuk menikmati gim, karena tujuannya lebih diarahkan untuk mendapatkan umpan balik (terkait kriteria kualitas

kesenangan dan aksesibilitas). Keluaran dari pengujian beta adalah laporan *bug* dan masukan pengguna. Sesi beta ditutup terutama karena ada dua alasan, baik istilah beta berakhir atau jumlah penguji beta yang ditentukan telah memberikan laporan pengujian mereka. Dari sini, dapat mengarah ke siklus produksi lagi untuk menyempurnakan produk atau terus merilis gim jika hasilnya memuaskan.

f) Release

Release adalah fase di mana pengembang gim telah mencapai tahap akhir dan siap dirilis ke publik. Rilis melibatkan peluncuran produk, dokumentasi proyek, berbagi pengetahuan, *post-mortem*, dan perencanaan untuk pemeliharaan dan perluasan gim.