JURNAL NUANSA INFORMATIKA
Volume 12 Nomor 2, Juli 2018
p-ISSN : 1858-3911, e-ISSN : 2614-5405
https://journal.uniku.ac.id/index.php/ilkom

IMPLEMENTASI MODEL PENGEMBANGAN SISTEM *GDLC* DAN ALGORITMA *LINEAR CONGRUENTIAL GENERATOR* PADA GAME PUZZLE

Rio Andriyat Krisdiawan
Fakultas Ilmu Komputer Universitas Kuningan
Jalan Tjut Nyak Dhien No. 36 A Cijoho Kuningan Jawa Barat 45513Telepon (0232) 2875097
rioandriyat@uniku.com/ rioandriyat@gmail.com

Games puzzle, masih banyak diminati oleh para gamers, karena pemecahan teka-teki, perhitungan matematika, dan bahkan termasuk kedalam games edukasi. Dalam pembuatannya game puzzle menitik beratkan bagaimana membuat game puzzle yang menantang untuk menyelesaikannya agar tidak mudah ditebak/membosankan dengan pola yag sama ketika dimainkan berulang-ulang. Untuk menjawab pertanyaan dalam pembuatan game tersebut, peneliti melakukan penelitian, bagaimana membangu game dengan model/metode pengembangan game yang sesuai dan bagaimana menerapkan sebuah algoritma dalam sebuah game puzzle, agar game menjadi lebih menantang untuk diselesaikan oleh gamers. Dalam penelitian ini, peneliti menggunakan model pengembangan *GDLC (Game Development Life Cycle)*, perancangan dengan menggunakan *UML*, serta Implementasi Algoritma *LCG (Linear Congruental Generator)* untuk pembangkit acak puzzle pada game. Adapun hasil dari penelitian ini yaitu model prototype pengembangan game dengan *GDLC* dan implementasi algoritma *LCG* untuk pembangkit puzzle secara optimal.

Kata Kunci : *Games Puzzle, GDLC, UML, LCG.*

Puzzle games, are still in great demand by gamers, because of puzzle solving, mathematical calculations, and even included in educational games. In making the puzzle game, focus on how to make a challenging puzzle game to solve it so that it is not easily guessed / boring with the same pattern when played repeatedly. To answer the question in making the game, researchers conducted research, how to build games with appropriate game development models / methods and how to implement an algorithm in a puzzle game, so that the game becomes more challenging for gamers to solve. In this study, researchers used the GDLC (Game Development Life Cycle) development model, design using UML, and the Implementation of the LCG (Linear Congruental Generator) Algorithm for puzzle random generation in the game. The results of this study are the game development prototype model with GDLC and the implementation of the LCG algorithm for the puzzle generator optimally.
Keywords: Games Puzzle, GDLC, UML, LCG.

1. PENDAHULUAN

Puzzle adalah sebuah *game* yang menampilkan potongan – potongan gambar dimana potongan gambar tersebut teracak susunannya sehingga memberikan tantangan tersendiri untuk menyusunnya.

Games puzzle, masih banyak diminati oleh para gamers, karena pemecahan teka-teki, perhitungan matematika, dan bahkan termasuk kedalam games edukasi. Dalam pembuatannya game puzzle menitik beratkan bagaimana membuat game puzzle yang menantang untuk menyelesaikannya agar tidak mudah ditebak/membosankan dengan pola yag sama ketika dimainkan berulang-ulang.

p-ISSN : 1858-3911, e-ISSN : 2614-5405
https://journal.uniku.ac.id/index.php/ilkom

Dalam pembuatan sebuah game diperlukan model pengembangan system yang sesuai, agar pegembangan game dapat berjalan dengan optimal seperti yang diinginkan. Selain dari model pengembangan game, untuk pembuatan game bergenre puzzle, diperlukan sebuah algoritma, agar game puzzle yang dibuat, memiliki teka-teki dan tantangan yang menantang untuk selalu dimainkan.

Untuk menjawab pertanyaan dalam pembuatan game tersebut, peneliti melakukan penelitian, bagaimana membangu game dengan model/metode pengembangan game yang sesuai dan bagaimana menerapkan sebuah algoritma dalam sebuah game puzzle, agar game menjadi lebih menantang untuk diselesaikan oleh gamers. Dalam penelitian ini, peneliti menggunakan model pengembangan *GDLC (Game Development Life Cycle)*, perancangan dengan menggunakan *UML*, serta Implementasi Algoritma *LCG (Linear Congruental Generator)* untuk pembangkit acak puzzle pada game. Adapun hasil dari penelitian ini yaitu model prototype pengembangan game dengan *GDLC* dan implementasi algoritma *LCG* untuk pembangkit puzzle secara optimal

Dari latarbelakang diatas, maka dapat diketahui rumusan masalah sebagai berikut :

1. Bagaimana membangun game dengan model pengembangan game *GDLC*.
2. Bagaimana mengimplementasikan algoritma *LCG* pada game bergenre puzzle.
3. Bagaimana cara kerja algoritma *LCG* dalam mengacak/pembangkit susunan *puzzle games.*

Batasan masalah dalam penelitian ini adalah:

1. Model *prototype* game puzzle adalah *puzzle slide*.
2. Model pengembangan game yang digunakan adalah model *GDLC*.
3. Potongan puzzle akan menggunakan ukuran 2 x 2 untuk level yang paling mudah, 3x3 untuk level sedang dan potongan ukuran 4 x 4 untuk level yang paling sulit.
4. Algoritma *LCG* digunakan untuk pembangkit dalam pengacakan susunan puzzle.

Maksud dan tujuan dari penelitian ini adalah bagaimana menerapkan model pengembangan game *GDLC* pada game puzzle dengan Pembangkit urutan acak puzzle dengan algoritma *LCG*.

.

2. METODELOGI PENELITIAN

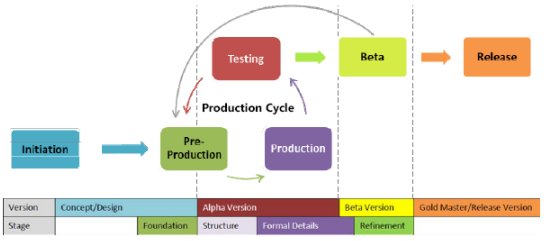
2.2 Metode Pengembangan Sistem

Metodologi yang digunakan oleh penulis dalam melakukan penelitian ini adalah *GDLC (Game Development Life Cycle)*. *GDLC* adalah suatu proses pengembangan sebuah game yang menerapkan pendekatan iteratif yang terdiri dari 6 fase pengembangan, dimulai dari fase inisialisasi/pembuatan konsep, preproduction, production, testing, beta dan realease.

Dari 6 fase tersebut dapat dikelompokkan menjadi 3 proses utama yaitu:

1. Proses Inisialisasi yang terdiri dari konsep dan design,
2. Proses produksi terdiri dari Pra Produksi, Produksi, dan Pengujian (Alpha dan Beta)
3. Realease

Fase dan Proses GDLC Guidelines dapat dilihat pada gambar dibawh ini :



Gambar 1. Fase dan Proses GDLC

A. Inisiasi

Adalah proses awal yang berupa pembuatan konsep kasar dari game, mulai dari menentukan game seperti apa

p-ISSN : 1858-3911, e-ISSN : 2614-5405
https://journal.uniku.ac.id/index.php/ilkom

yang akan dibuat, mulai dari indentifikasi dari trending, topik,, target user dari game yang akan dibuat. Output dari tahap initation adalah konsep game dan deskripsi permainan yang sangat sederhana.

B. Pra-produksi

Pra-produksi adalah salah satu fase yang penting dalam siklus produksi. Pra-produksi melibatkan penciptaan dan revisi desain game dan pembuatan prototipe permainan. Desain game berfokus pada mendefinisikan genre permainan, gameplay, game mekanik/konfensional, alur cerita, karakter, tantangan, faktor kesenangan, aspek teknis, dan dokumentasi elemennya dalam **Dokumen Desain Game (GDD).**

Pra-produksi berakhir ketika revisi atau perubahan desain game telah disetujui dan didokumentasikan di GDD.

C. Produksi

Produksi adalah proses inti yang berputar di sekitar penciptaan aset, pembuatan kode sumber, dan integrasi kedua elemen. Prototipe terkait dalam fase ini adalah perincian dan penyempurnaan formal.

Rincian Formal adalah struktur yang disempurnakan dengan mekanika dan aset yang lebih lengkap. Kegiatan produksi yang terkait dengan penciptaan dan penyempurnaan detail formal adalah menyeimbangkan permainan (terkait dengan kriteria kualitas yang seimbang), menambahkan fitur baru, meningkatkan kinerja secara keseluruhan, dan memperbaiki bug (terkait dengan kriteria kualitas fungsional dan internal yang lengkap).

Penyeimbangan permainan yaitu penyesuaian yang terkait dengan kesulitan permainan untuk membuat kesulitan game yang tepat (Leveling).

Refinement adalah prototipe lengkap yang merupakan subjek dari permainan. Kriteria kualitas terkait game fun dan dapat diakses. Kegiatan selama penyempurnaan diarahkan untuk membuat permainan lebih menyenangkan, menantang, dan lebih mudah dipahami. Hanya perubahan kecil yang diizinkan dalam fase ini.

D. Pengujian

Pengujian dalam konteks ini berarti pengujian internal dilakukan untuk menguji kegunaan permainan dan pemutaran. Metode pengujian khusus untuk setiap tahap prototipe.

Perincian Formal Pengujian dilakukan menggunakan playtest untuk menilai fungsionalitas fitur dan kesulitan permainan (terkait dengan keseimbangan).

Metode untuk menguji kriteria kualitas fungsional adalah melalui fitur playtesting. Untuk menguji kriteria kualitas internal yang lengkap, dapat dilakukan melalui playtesting bersamaan dengan uji fungsi.

Ketika tester menemukan bug, celah, atau kegagalan selama playtesting, penyebab dan skenario untuk mereproduksi kesalahan perlu didokumentasikan dan dianalisis. Untuk menguji kriteria kualitas yang seimbang, bermain dengan beberapa perawatan yang berbeda digunakan untuk mengkategorikan apakah perawatan terlalu sulit, terlalu mudah, atau baik-baik saja.

Perbaikan Pengujian terkait dengan menyenangkan dan kriteria kualitas aksesibilitas. Dalam penyempurnaan pengujian, kesenangan diuji melalui playtest dan umpan balik langsung dari sesama pengembang, apakah itu membosankan, membuat frustrasi, menantang, dll. Aksesibilitas dapat diuji melalui pengamatan perilaku penguji. Jika tester merasa sulit untuk bermain dan memahami permainan, itu berarti bahwa game tersebut tidak cukup dapat diakses. Output dari pengujian adalah laporan bug, permintaan perubahan, dan

p-ISSN : 1858-3911, e-ISSN : 2614-5405
https://journal.uniku.ac.id/index.php/ilkom

keputusan pengembangan. Hasilnya akan memutuskan apakah sudah waktunya untuk maju ke fase berikutnya (Beta) atau mengulangi siklus produksi.

E. Beta

Beta adalah fase untuk melakukan pengujian pihak ketiga atau eksternal yang disebut pengujian beta. Pengujian beta masih menggunakan metode pengujian yang sama dengan metode pengujian sebelumnya, karena prototipe terkait dalam pengujian beta adalah perincian dan penyempurnaan formal. Metode pemilihan tester datang dalam dua jenis: beta tertutup dan beta terbuka. Ditahap beta hanya memungkinkan individu yang diundang untuk menjadi peserta, sementara beta terbuka memungkinkan siapa saja yang mendaftar menjadi peserta. Kriteria kualitas dalam beta terkait erat dengan tahap prototipe saat ini. Dalam pengujian detail resmi, penguji diminta untuk menemukan bug (terkait dengan kriteria kualitas fungsional dan internal yang lengkap). Dalam penyempurnaan pengujian, penguji diberi lebih banyak kebebasan untuk menikmati permainan, karena sasaran lebih diarahkan untuk mendapatkan umpan balik (terkait dengan kriteria kualitas aksesibilitas dan menyenangkan). Output dari pengujian beta adalah laporan bug dan masukan pengguna. Sesi Beta ditutup terutama karena 2 alasan, baik jangka beta berakhir atau jumlah penguji beta yang ditentukan telah memberikan laporan uji mereka. Dari sini, dapat menyebabkan siklus produksi lagi untuk memperbaiki produk atau terus merilis game jika hasilnya memuaskan.

F. Rilis

Sudah saatnya build game telah mencapai tahap akhir dan siap untuk dirilis ke publik. Rilis melibatkan peluncuran produk, dokumentasi proyek, berbagi pengetahuan, post-mortems, dan perencanaan untuk pemeliharaan dan ekspansi permainan.

("Rido Ramadan and Yani Widyani; Game Development Life Cycle")

2.2.1 Algoritma Linear Congruential Generator

Metode *Linear Congruential Generator* digunakan untuk membangkitkan bilangan acak *x1, x2, ..., xn* yang bernilai [*0,m*] dengan memanfaatkan nilai sebelumnya. Untuk membangkitkan bilangan acak dengan metode *Linear Congruential Generator*, didefinisikan

Xn = (a(Xn-1) + b) mod m

dimana a,b dan m dinamakan nilai pembangkit, Xn-1 adalah bilangan acak sebelumnya Xn dinamakan bilangan deret ke-n dari deretnya X0 dinamakan nilai awal, biasanya nilai ini yang di-gunakan dalam proses *randomize* (mengacak di awal atau state awal).

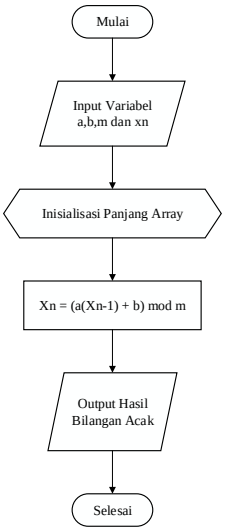
Secara umum, langkah - langkah dari penerapan metode *Linear Congruential Generator* pada pengacakan adalah sebagai berikut :

1. Menentukan nilai pembangkit yaitu **a** ,**b** dan **m**, **m** adalah nilai acak atau jumlah yang di acak.
2. Mendefinisikan nilai awal atau state awal yaitu **X0**.
3. Proses pembangkitan bilangan acak dengan **Xn = (a(Xn-1) + b) mod m**
4. Menampilkan hasil bilangan acak.

Ciri khas dari *Linear Congruential Generator* adalah terjadi pengulangan pada periode waktu tertentu atau setelah sekian kali pembangkitan.

Flowchart Penyelesaian Prosedur *Linear Congruential Generator* adalah sebagai berikut :

p-ISSN : 1858-3911, e-ISSN : 2614-5405
https://journal.uniku.ac.id/index.php/ilkom



Gambar 2. *Flowchart* Metode *Linear Congruential Generator*

Langkah-langkah yang dikerjakan di dalam Metode *Linear Congruential Generator* ini yaitu :

1. Masukan variable dari a,b,m dan Xn, dimana Xn adalah bilangan acak ke n, a dan b adalah konstanta *Linear Congruential Generator*, M adalah batas maksimum bilangan acak.
2. Setelah memasukkan nilai dari *variable* - *variable*, yaitu menginisialisasi panjang array.
3. Melakukan perhitungan pada rumus **Xn = (a(Xn-1) + b) mod m**
4. Setelah melakukan perhitungan kita mendapatkan nilai dari bilangan acak tersebut.

Penentuan konstanta *Linear Congruential Generator* (a, b dan m) sangat menentukan baik tidaknya bilangan acak yang diperoleh dalam arti memperoleh bilangan acak yang seakan-akan tidak terjadi pengulangan.

Tabel 1. Contoh *puzzle* 9 kotak yang belum teracak

1	**2**	**3**
4	**5**	**6**
7	**8**	**0**

Contoh pengacakan untuk 9 kotak *puzzle* dengan ketentuan : a=4, b=5, m=9, dan x(0)=12, dimana nilai x(0) didapat dari pembangkitan bilangan acak $1 \geq i \leq 100$. Maka:

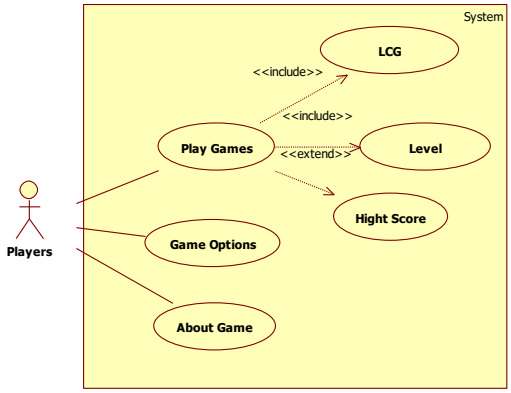
x(i) = (a * x(i-1) + b) mod m :
x(1) = (4*12+5) mod 9 = 8
x(2) = (4*8+5) mod 9 = 1
x(3) = (4*1+5) mod 9 = 0
x(4) = (4*0+5) mod 9 = 5
x(5) = (4*5+5) mod 9 = 7
x(6) = (4*7+5) mod 9 = 6
x(7) = (4*6+5) mod 9 = 2
x(8) = (4*2+5) mod 9 = 4
x(9) = (4*4+5) mod 9 = 3

Dari pengujian di atas didapat urutan kemunculan: 8, 1, 0, 5, 7, 6, 2, 4, 3. Dan dapat dilihat bahwa penentuan nilai konstanta *Linear Congruential Generator* cukup tepat pada *puzzle* 9 kotak ini dengan nilai maksimal pengacakan 9 sesuai dengan jumlah kotak dimana terlihat seakan-akan urutan kemunculan tidak terjadi perulangan.

Tabel 2. Contoh *puzzle* 9 kotak yang sudah teracak

8	**1**	**0**
5	**7**	**6**
2	**4**	**3**

3.2 Perancangan *Use Case Diagram*



Gambar 3. *Use case* diagram sistem

p-ISSN : 1858-3911, e-ISSN : 2614-5405
https://journal.uniku.ac.id/index.php/ilkom

Pada gambar 3 *use case* diagram akan di deskripsikan sebagai berikut :

a. Play Game

Usecase : Play Game
Aktor : *Players*
Deskripsi : Player dapat mengakses permainan dengan Play Game dan memilih level sesuai pencapaian level yang sudah dicapai sebelumnya.
Kondisi Sebelum : Halaman utama
Kondisi Setelah : Halaman *leve*l

Tabel 3. Deskripsi *use case* Play Game

Aktor	Game
1. Player mengakses play game	2. Menampilkan level
3. Memilih Level	4. Mengacak potongan *puzzle* menggunakan algoritma *LCG*
5. Bermain menyusun gambar *puzzle*	6. Menghitung jumlah pergerakan puzzle
	7. Mengkalkulasikan pergerakan puzzle dengan nilai rata-rata pergerakan puzzle di level yang sama
8. Menggunakan *button solve*	9. Menyusun potongan *puzzle*
	10. *Validasi* susunan *puzzle*
	11. Hitung skor
	12. Menyimpan skor dalam *database*

b. Game Options

Usecase : Game Options
Aktor : *Players*
Deskripsi : Players memilih *Game option*, untuk melakukan pengaturan permainan dan melihat intruksi permainan.
Kondisi sebelum : Halaman Utama
Kondisi setelah : Halaman *Game Options*

Tabel 4. Deskripsi *use case Game Options*

Aktor	Game
1. Mengakses halaman game options	2. Menampilkan halaman option game
3. Memilih pengaturan dan intruksi permainan	4. Menampilkan halaman pengaturan dan intruksi permainan

c. About Game

Usecase : About Game
Aktor : *Players*
Deskripsi : players mengakses menu ke about games.
Kondisi sebelum : Halaman utama
Kondisi setelah : Halaman About Games

Tabel 5. Deskripsi About Games

Aktor	Game
1. Menjalankan game	2. Halaman Utama
3. Memilih menu about games	4. Menampilkan about games

d. Hight Skor

Usecase : Hight skor
Aktor : *User*
Deskripsi : player dapat mengakses hight score sebelum dan sesudah bermain games
Halaman sebelum : Halaman play games
Halaman setelah : Halaman hight score

Tabel 6. Deskripsi *use case* Hight Score

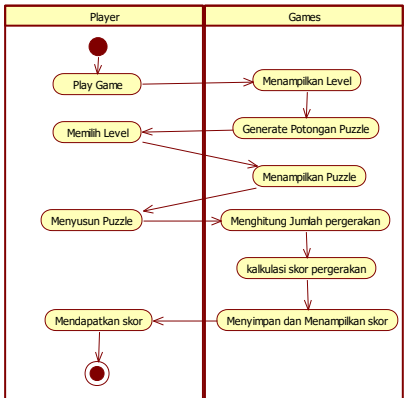
Aktor	Game
1. Mengakses halaman skor,	2. Menampilkan halaman hight score

3.3 Activity Diagram

Diagram *activity* akan menampilkan aliran kerja atau aktivitas dari sebuah sistem yang ada pada aplikasi *game* berbentuk *puzzle*. Di mana aktivitas ini akan menggambarkan aktivitas sistem dan bukan yang di lakukan oleh aktor terhadap aplikasi, jadi *activity* diagram ini akan menampilkan aktivitas-aktivitas yang dilakukan oleh *system*.

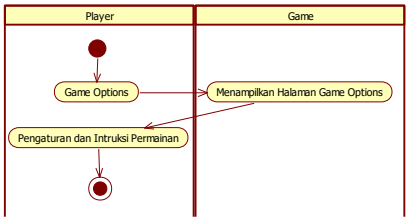
p-ISSN : 1858-3911, e-ISSN : 2614-5405
https://journal.uniku.ac.id/index.php/ilkom

***a. Activity* Diagram Play Game**



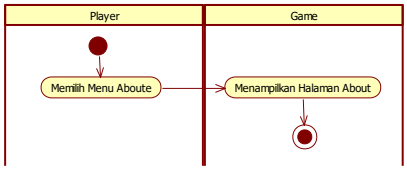
Gambar 4. *Activity* diagram play Game

***b. Activity* Diagram Game Option**



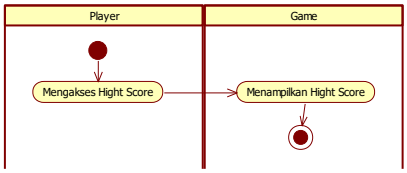
Gambar 5. *Activity* diagram Game Option

***c. Activity* Diagram About Game**



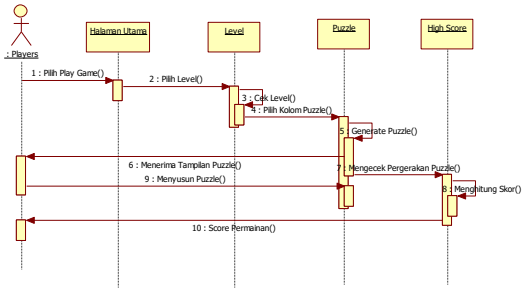
Gambar 6. *Activity* diagram about game

***d. Activity* Diagram Hight Score**



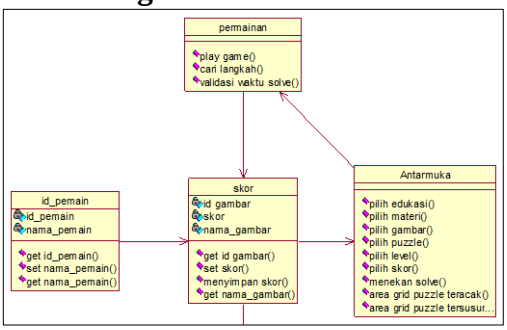
Gambar 7. *Activity* diagram Hight Score

3.4 Sequence Diagram



Gambar 8. *Sequence* diagram Play Game

3.5 Class Diagram



Gambar 9. *Class* diagram aplikasi *game puzzle*

3.6 Prototype Interface

1. Halaman Utama Aplikasi

Gambar 10. Halaman Utama Game

2. Halaman Utama Permainan Puzlle

Gambar 11. Tampilan Utama Menu Games Puzlle.

p-ISSN : 1858-3911, e-ISSN : 2614-5405
https://journal.uniku.ac.id/index.php/ilkom

Prose Generate pengacakan puzzle tersebut menggunakan algoritma LCG.

3. Halaman Hight Score

Gambar 12. Halaman Hight Score

3.7 Pengujian Game (Unit Testing)

Pengujian *Game* perangkat lunak adalah bagian terpenting guna mencari kesalahan-kesalahan yang ada dalam perangkat lunak yang di uji, pengujian ini dimaksudkan untuk mengetahui perangkat lunak yang di buat sudah memenuhi kriteria yang sesuai dengan perancangan awal perangkat lunak tersebut. Pengujian perangkat lunak menggunakan pengujian *black box* dan pengujian *white box.*

3.8 Pengujian *Black Box*

Pengujian *black box* adalah proses pengujian aspek fundamental aplikasi tanpa memperhatikan struktur logika internal perangkat lunak. Proses pengujian ini dilakukan untuk mengetahui apakah aplikasi perangkat lunak dapat berjalan dan berfungsi dengan benar.

Tabel 7.Hasil Pengujian *Black Box*

No	Fungsi yang di uji	Cara Menguji	Hasil yang diharapkan	Hasil yang keluar
1	Cek menu Level	Memilih menu level	Tampil menu level	Sesuai dengan harapan *Valid*
2	Cek menu play games	memilih menu play game	Tampil pemilihan level yang nanti diacak	Sesuai dengan harapan *Valid*
			menjadi puzlle sesuai dengan level	

3.9 Pengujian *White Box*

Pengujian *white box* merupakan sebuah pengujian yang dilakukan dengan melihat kedalam modul dan kode-kode yang ada didalam aplikasi. Tujuannya adalah sebagai petunjuk untuk mendapatkan program yang benar secara menyeluruh sehingga sistem yang dirancang mampu menghasilkan *interface* dan *output* yang sesuai dengan kebutuhan. Secara sekilas dapat diambil kesimpulan *white box testing* merupakan petunjuk untuk mendapatkan program yang benar secara 100%.

Pengujian *White Box Testing* dapat dilakukan sebagai berikut :

a. V(G) = E – N + 2 hasilnya sama dengan V(G) = P + 1
b. Flowgraph mempunyai region yang sama dengan jumlah V(G)maka sistem dikatakan sudah terbukti efektif dan efisien.

Tabel 8. Source Code Pengujian

Line/node	*Source Code*
1	`public void lcg (int[] lc, int width, int height){`

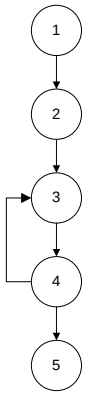
2	`int[] lcg = new int[width * height];`
	`int n = 0;`
	`int idxFinish = 0;`

3	`for (int i = lc.length-1; i >0;i--){`

4	`int index = random.nextInt(i);`
	`int tmp = lc[index];`
	`lc[index] = lc[i];`
	`lc[i] = tmp;`
	`lcg[n] = lcg[i];`
	`idxFinish = lcg[index];`
	`n++;`
	`}`

5	`lcg[lc.length-1] = idxFinish;`
	`lc = lcg;`
	`}`

p-ISSN : 1858-3911, e-ISSN : 2614-5405
https://journal.uniku.ac.id/index.php/ilkom



Gambar 14. Perhitungan Flowgraph

Dari diagram alir *tracking maker*, dapat dihitung *cyclomatic complexity* yaitu :

V(G) = 5 – 5 + 2

V(G) = 2

Dari perhitungan *cyclomatic complexity independent* path yaitu :

Path1 = 1-2-3-4-3-5

Path2 = 1-2-3-5

4. KESIMPULAN

Berdasarkan hasil implementasi dan pembahasan yang telah dilakukan, maka dapat ditarik kesimpulan sebagai berikut.

1. Metode *Linear Congruential Generator* mampu mengacak kotak puzzle dengan sangat bias karena setiap kotak berpindah tempat sesuai yang diharapkan oleh penulis.
2. Penggunaan Metode *Linear Congruential Generator* sangat efektif dalam pengacakan susunan puzzle setelah dibandingkan dengan aplikasi yang sedang berjalan.
3. *GDLC (Game Development Life Cycle)* sangat cocok dalam model pengembangan sebuah Game.

5. Saran

Sesuai dengan permasalahan yang ada dan setelah perancangan game ini selesai, maka diberikan beberapa saran yang dapat digunakan dalam pengembangan game di masa mendatang. Adapun saran yang ingin disampaikan yaitu sebagai berikut.

1. Dibutuhkan algoritma untuk penilaian dalam menyelesaikan potongan puzzle.
2. Perlu dikembangkan untuk system operasi selain android.
3. Tahapan pengujian sebaiknya dilakukan sesuai model pengembangan yaitu alfa dan beta testing.

Daftar Pustaka

[1]. Coad, Peter and Yourdon, Edward. *Object-Oriented Analysis*, Yourdon Press, 1991.

[2]. Coad, Peter and Yourdon, Edward. *Object-Oriented Design Second Edition*, Yourdon Press. 1991.

[3]. Fowler, Martin.*UML Distiled: Panduan Singkat Bahasa Pemodelan Object Standar, Edisi 3*, Penerbit Andi: Yogyakarta. 2004.

[4]. Kruchten, Philippe, *The Rational Unified Process An Introduction, Second Edition*, John Wiley and Son Ltd. 2006.

[5]. Pressman, Roger S. (2007). Rekayasa Perangkat Lunak: pendekatan praktisi (Buku1). Beizer, B. (1995). *Black-Box Testing*, Wiley. Yogyakarta: Andi.

[6]. A.S, Rosa dan M. Salahuddin. (2015). *Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek*. Bandung : Informatika Bandung

[7]. A.S, Rosa. (2011). *Modul Pembelajaran Rekayasa Perangkat Lunak*. Bandung : Modula.

[8]. Jeffry. (2010). *Rancangan Bangun Aplikasi Edukasi Puzzle Pengenalan Tokoh Sejarah Berbasis Android Dengan Metode Linear Congruential Generator*. Diakses 02 Februari, dari situs http://www.pelita-informatika.com

[9]. Ramadhan, Karli, Astuti Lastri Widya dan Verano Dwi Asa. (2015). *Game Edukasi Tebak Gambar Bendera Negara Menggunakan Metode Linear Congruential Generator (LCG) Berbasis Android*. Diakses 07 Februari, dari situs ejournal.uigm.ac.id.

[10].*Rido Ramadan and Yani Widyani; Game Development Life Cycle.(Jurnal* ICACSIS 2013 ISBN: 978-979-1421-19-5)