# IMPLEMENTASI ALGORITMA MULTIPLICATIVE CONGRUENTIAL GENERATOR PADA GAME PANAHAN

# IMPLEMENTATION OF MULTIPLICATIVE CONGRUENTIAL GENERATOR ALGORITHM IN ARCERY GAME

**Muhamad Rihan Yunisa[1], Roswan Latuconsina[2], Anton Siswo Raharjo Ansori[3]**
[1]Prodi S1 Teknik Komputer, Fakultas Teknik, Universitas Telkom
[2]Prodi S1 Teknik Komputer, Fakultas Teknik, Universitas Telkom
[3]Prodi S1 Teknik Komputer, Fakultas Teknik, Universitas Telkom
[1]**rihanyunisa@student.telkomuniversity.ac.id**, [2]**roswan78@gmail.com**, [3]**raharjotelu@gmail.com**

**Abstrak**

Ketika seseorang sedang dilanda kejenuhan akibat akitivitas pekerjaan yang cukup padat tentu akan menimbulkan stress dan akan menghambat aktivitas kedepannya. Maka dari itu, orang-orang melampiaskannya dengan memainkan sebuah permainan yang ada pada perangkat keras seperti komputer maupun gawai. penulis mengembangkan permainan panahan (Archery Game) sebagai prosedur pemrograman dalam bentuk skrip yang ditulis dalam perangkat lunak game development yaitu Unity3D. Algoritma yang digunakan pada penilitian ini adalah Multiplicative Congruential Generators yaitu implementasi dari metode Pseudorandom Number Generator (PRNG) yang menggabungkan dua atau lebih sebuah generator linear kongruensial. Gabungan generator linear kongruensial juga memiliki sebuah algoritma khusus di dalamnya, di mana beberapa variabelnya mendeklarasikan modulus-modulus dari MCG dan nilai acaknya. Target statis maupun dinamis dapat bergerak random yang di dapat dari PRNG melalui algoritma MCG.

Kata Kunci: *game, Multiplicative Congruential Generator (MCG), Pseudorandom Number Generator (PRNG).*

**Abstract**

When someone is hit by boredom due to work activities that are quite dense it will certainly cause stress and will hinder future activities. Therefore, people take it out by playing a game that is on hardware such as computers or devices. In this research, the author develop an archery game (Archery Game) as a programming procedure in the form of scripts written in game development software, Unity3D. The algorithm used in this research is Multiplicative Congruential Generators, which is the implementation of the Pseudorandom Number Generator (PRNG) method that combines two or more congruential linear generators. Congruential linear generator compounds also have a special algorithm in them, in which some of the variables declare modulus of MCG and its random value. Static and dynamic targets can move randomly obtained from PRNG through the MCG algorithm.

Keywords: *game, Multiplicative Congruential Generator (MCG), Pseudorandom Number Generator (PRNG).*

## 1. Pendahuluan

Perkembangan aplikasi permainan pada zaman sekarang ada berbagai macam, mulai dari grafis yang makin ditingkatkan, berbagai macam *genre* dan ditujukan usia tertentu. Ini menjadikan para *developer* aplikasi permainan yang diharuskan memiliki tantangan dan tingkat kesulitan yang menyesuaikan dengan kemampuan pemain. Pemain itu ada dua jenis yaitu *Hardcore Gamers* yang menyukai tantangan pada permainan dan ingin menyelesaikannya dan *Casual Gamers* yang cenderung mudah menyerah ketika permainannya terlalu sulit.

Teknik/metode yang digunakan pada penilitian ini adalah dengan gabungan metode *Pseudorandom Number Generator (PRNG)* dan algoritma *Multiplicative Congruential Generators*. Definisi dari metoda yang digunakan yaitu implementasi dari *Pseudorandom Number Generator (PRNG)* yang menggabungkan dua atau lebih sebuah generator linear kongruensial. Gabungan generator multiplikatif kongruensial juga memiliki sebuah algoritma khusus di dalamnya, di mana beberapa variabelnya mendeklarasikan modulus-modulus dari algoritma dan nilai acaknya[1]. Tujuannya memberikan kesan atraktif seperti mengacak objek-objek yang dijelaskan sebelumnya, yaitu untuk membuat *game* lebih menarik atau tidak monoton dan memiliki tantangan bagi pemainnya.

Pada proses *scripting*, menggunakan metode *Pseudorandom Number Generator (PRNG)* melalui algoritma *Multiplicative Congruential Generators (MCG)*. Pada MCG, diketahui bahwa suatu angka memiliki periode tertentu, dimana angka akan berulang pada iterasi tertentu. Metoda ini dipilih karena pada bagian target di dalam permainan menerapkan penggunaan

*scripting*, di mana semua bagian konten dapat dihasilkan dengan tulisan kode yang terstruktur dengan baik dan benar[2]. Alasan yang memungkinkan dari penggunaan metode ini dilihat dari tingkat acak *(random)* dari kontn yang dihasilkannya. Mengusahakan agar pemain nantinya merasakan permainan yang menantang. Dengan adanya *Multiplicative Congruential Generators*, pemain beradaptasi dengan target yang tidak monoton sebelumnya dengan berupa pergerakan target statis (kiri-kanan) dan pergerakan binatang yang bergerak secara acak sehingga tingkat kesulitan semakin tinggi.

## 2. Perancangan Sistem

Dalam penelitian ini diterapkan salah satu pembuatan target secara otomatis dengan menggunakan *Pseudorandom Number Generator (*PRNG) sebagai indeks untuk *spawn* suatu target secara *random*. PRNG merupakan *Random Number Generator* yang menghasilkan angka *random* berdasarkan *seed* yang diberikan. PRNG yang digunakan adalah *Multiplicative Congruential Generator*. Dengan menggunakan *Multiplicative Congruential Generator* maka angka *random* yang digunakan akan selalu berbeda berdasarkan *seed*, akan tetapi jika menggunakan *seed* yang sama akan menghasilkan pola angka yang sama. Untuk menyelesaikan masalah tersebut akan digunakan *random number* bawaan sebagai *seeder* untuk menghasilkan *output* yang lebih *random* lagi. Setiap target yang akan di *spawn* akan di indeks di dalam sebuah *array*. Untuk setiap target akan di *spawn* secara *random*. Target yang di *spawn* akan di acak peletakannya melalui properti *transform* pada *Unity3d* dengan *range* yang sudah ditentukan, agar tidak melebihi dari luas dari suatu *arena*. Target yang dibuat berbentuk burung dan kelinci dimana burung bergerak sesuai dengan *waypoint* yang tersedia ,sedangkan kelinci bergerak secara acak sesuai dengan bentuk *arena*. Perlu diingat bahwa *seed* merupakan nilai yang diberikan untuk membuat suatu angka acak. Dengan begitu *random number generator* akan menciptakan angka yang lebih acak lagi. *Range* dari *chance* setiap target ditentukan oleh *developer* dan akan dilakukan perhitungan ulang oleh sistem.

### 2.1 Algoritma Multiplicative Congruential Generator

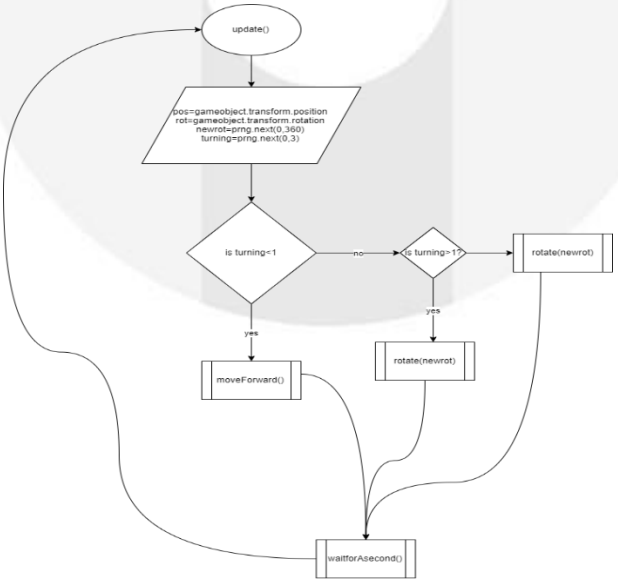
*Multiplicative Congruential Generator* adalah suatu algoritma yang digunakan untuk menghasilkan urutan-urutan (*sequence)* dari angka-angka sebagai hasil perhitungan dengan komputer yang diketahui distribusinya sehingga angka tersebut muncul secara random dan digunakan terus menerus. Pengertian *random* menunjukkan bahwa algoritma tersebut akan menghasilkan suatu angka yang akan berperan dalam pemunculan angka yang akan keluar dalam proses di komputer. Suatu angka yang diperoleh merupakan angka penentu bagi angka random berikutnya, demikian seterusnya. Meskipun begitu, angka-angka yang muncul dapat berlain lainan. Berikut adalah persamaan dari algoritma tersebut:

$$X_{K+1} = a \cdot X_K \bmod m$$

### 2.2 *Pseudorandom Number Generator*

*Pseudorandom Number Generator* (PRNG) merupakan algoritma yang menghasilkan deret bit yang unik yang berasal dari nilai awal yang di sebut dengan *seed* [12]. *Pseudorandom number generator* digunakan dalam banyak hal seperti, permainan, robotik, peningkatan performa, modeling dan simulasi, algoritma genetik dan masih banyak lagi. Dengan menggunakan PRNG angka dapat di *generate* secara acak berdasarkan *seed* yang diberikan. Sebuah PRNG harus melalui beberapa tes untuk menguji keacakan dan performa dari *random number* tersebut. TestU01 merupakan tes yang digunakan untuk menguji *Empirical Statistik Testing* pada PRNG jenis *Uniform* [6]. *Empirical Statistik Testing* merupakan suatu tes yang digunakan untuk menentukan apakah *Random Number Generator* memproduksi angka acak yang berprilaku sebagaimana angka acak berprilaku.

### 2.3 Flowchart pergerakan dinamis (binatang)




Gambar 2. *Flowchart* pergerakan dinamis

Pada proses prediksi dilakukan dengan beberapa langkah berikut :

1. Setiap ai wanderer akan memiliki gambaran pada update kurang lebih seperti pada gambar di atas, nilai prng next 0,360 akan menghasilkan nilai dari 0 sampai 359 derajat. Akan di tentukan kemana akan berputar berdasarkan nilai turning. Nilai turning berada pada 0,1 dan 2 dimana 0 tidak berbelok 1 berbelok ke kanan dan 2 berbelok ke kiri.
2. Nilai rotasi akan diterima melalui fungsi update. Nilai arah akan dikalikan dengan -1 apabila berbelok ke kiri dan 1 apabila berbelok ke kanan
3. Untuk move forward atau jalan ke depan akan di jalankan setelah berbelok. Apabila di depan dan di kiri ada tembok dan ai akan wanderer akan bergerak ke kanan. Apabila ada tembok di sebelah kanan maka sebaliknya. Nilai hit akan didapatkan melalui sensor dari unity berupa raycast. Minimal terdapat dua raycast untuk deteksi tembok pada ai wanderer.

## 3. Pengujian sistem

### *Bird*

Pada pengujian ini akan dilakukan 74 kali pengacakan, dengan *seed* mulai dari 1,27 dan 83 dengan vektor x, y dan z didapatkan 74 data sehingga didaptkan nilai *Mean, Standard error, Median, Standard deviation, Sample variance.*

Tabel 1. *Bird*

| **koordinat** | ***Mean*** | ***Standard error*** | ***Median*** | ***Standard deviation*** | ***Sample variance*** |
|---|---|---|---|---|---|
| X | -0.8903 | 0.29089 | -0.36571 | 21.52583 | 469.7089 |
| Y | 11.51075 | 0.053624 | 9.35798 | 3.995275 | 15.96222 |
| Z | -3.12031 | 1.811742 | -7.99395 | 134.9841 | 18220.7 |

### *Position Rabbit*

Pada pengujian ini akan dilakukan 27.729 kali pengacakan, dengan *seed* mulai dari 0 sampai 27 dengan vektor x dan z dan didapatkan 55.289 data sehingga didaptkan nilai *Mean, Standard error, Median, Standard deviation, Sample variance*

Tabel 2. *Position rabbit*

| **koordinat** | ***Mean*** | ***Standard error*** | ***Median*** | ***Standard deviation*** | ***Sample variance*** |
|---|---|---|---|---|---|
| X | 74.917 | 0.006441 | 92.63962 | 178.6029 | 31899.01 |
| Z | -109.362 | 0.006551 | -81.3563 | 181.6532 | 32997.89 |

### *Rotation rabbit*

Pada pengujian ini akan dilakukan 34.429 kali pengacakan, dengan *seed* mulai dari 0 sampai 27 dengan vektor y dan didapatkan 27.728 data sehingga didaptkan nilai *Mean, Standard error, Median, Standard deviation, Sample variance*

Tabel 3. *Rotation rabbit*

| **koordinat** | ***Mean*** | ***Standard error*** | ***Median*** | ***Standard deviation*** | ***Sample variance*** |
|---|---|---|---|---|---|
| Y | 182.2541 | 0.003207 | 177.2877 | 110.4271 | 12194.15 |

**Daftar Pustaka:**

[1] Wong .H, 2011, *A Study of The Video Game Industry In U.S Metropolitan Areas Using Occupational Analysis*,USA, University of Massachusetts Amherst.

[2] Larry Katz, James Parker, Hugh Tyreman, Gail Kopp, Richard Levy, Ernie Chang, 2006, *VIRTUAL REALITY in SPORT and WELLNESS: PROMISE and REALITY*, Canada, International Journal of Computer Science in Sport – Volume 4/Edition 1.

[3] Sasha Azad, Carl Saldanha, Cheng Hann Gan, and Mark O. Riedl, *Mixed Reality Meets Procedural Content Generation in Video Games*,USA, Experimental AI in *Game*s: Papers from the AIIDE Workshop AAAI Technical Report WS-16-22.

[4] Georgios N. Yannakakis, and Julian Togelius, 2011, *Experience-Driven Procedural Content Generation*, IEEE TRANSACTIONS ON AFFECTIVE COMPUTING, VOL. 2, NO. 3, JULY-SEPTEMBER 2011.

[5] Gilbert Nwankwo, Sabah Mohammed and Jinan Fiaidhi, 2017, *Procedural Content Generation for Dynamic Level Design and Difficulty in a 2D Game Using UNITY*, Canada, *International Journal of Multimedia and Ubiquitous Engineering Vol.12, No.9 (2017), pp.41-52.*

[6] C. Sommariva, E. Nardon, P. Beyer, M. Hoelzl, G. T. A. Huijsmans, D. van Vugt and JET Contributors, 2016, *Test particles dynamics in the JOREK 3D non-linear MHD code and application to electron transport in a disruption simulation*,France, CEA.

[7] Michael Stauffer, Thomas Hanne, Rolf Dornberger, 2016, *Uniform and Non-Uniform Pseudorandom number generators in a Genetic Algorithm Applied to an Order Picking Problem*, Switzerland, 2016 IEEE Congress on Evolutionary Computation (CEC).

[8] Melissa E. O'Neill, 2014, *PCG: A Family of Simple Fast Space-Efficient Statistikally Good Algorithms for Random Number Generation*, USA, ACM journals of mathematical software (submitted).

[9] Julian Togelius, Emil Kastbjerg, David Schedl and Georgios N, 2011 Yannakakis, *What is Procedural Content Generation? Mario on the borderline*, Denmark, DOI: 10.1145/2000919.2000922.