Implementation of Cellular Automata and Fisher Yates Methods on Map Generators in Building Games Using Unity 3D Game Engine 2021

Leni Marlina, Obi Yasil Lubis

Abstract

The increasing demand for dynamic and replayable game experiences has led to the widespread adoption of procedural content generation, particularly in creating non-repetitive and immersive game environments. This research presents the implementation of Cellular Automata and the Fisher-Yates Shuffle algorithm to develop an efficient and adaptive map generator system within a game built using the Unity 3D Game Engine. Cellular Automata is utilized to generate natural-looking map structures, such as cave-like or maze-like formations, through a grid-based simulation that evolves over multiple iterations based on the states of neighboring cells. This method enables the generation of unique and organic map layouts for each game session. To address potential issues such as disconnected areas and unfair object placement, the Fisher-Yates Shuffle algorithm is applied to randomly distribute key gameplay elements—including player spawn points, enemies, and items—across valid and accessible map regions in an unbiased manner. The system was tested under various parameter configurations to evaluate map quality, connectivity, and computational performance. The results demonstrate that the combination of Cellular Automata and Fisher-Yates Shuffle can effectively generate varied, functional, and aesthetically pleasing maps with high performance and consistency. Furthermore, Unity 3D proved to be a suitable platform for integrating the procedural generation system with other game components such as AI navigation and resource management. This study concludes that integrating Cellular Automata with Fisher-Yates Shuffle offers a robust solution for creating adaptive procedural map generators that enhance gameplay diversity, reduce manual level design time, and improve overall player engagement.

Keywords: procedural generation, map generator, Cellular Automata, Fisher-Yates Shuffle, Unity 3D, game development

Leni Marlina[1]
[1]Computational Science and Digital Intelligence
e-mail: lenimarlina@dosen.pancabudi.ac.id[1]

Obi Yasil Lubis[2]
[1]Computational Science and Digital Intelligence
e-mail: obilubis@gmail.com[2]

Introduction.

As the gaming industry grows rapidly, the amount of content needed in games is constantly increasing. An increase in the amount of content is needed to keep players interested in the game, hence the increasing design work is needed to meet these requirements. Creating in-game content, such as enemies, maps, levels, and items, certainly takes a lot of time and money. A human usually works very slowly compared to a computer. If such content can be developed algorithmically, the company that develops the game can save a lot of time and money in the work of these tasks. PCG or Procedural Content Generation is a way to create content in the game, which can be in the form of characters, stories, level missions, enemies and others automatically using certain methods.

Therefore, a game map generator is built dynamically using Procedural Content Generation (PCG) to automatically build different forms of mapping based on the number code selected. The automatic form of mapping is arranged in different shapes according to the chosen number code, of course, creating this content can help make it easier for game makers to work on each scene in the game they are building.

Games (Games)

According to Santoso, Erick, Gregorius Satia Budhi, and Rolly Intan (2017:9), a game is a system in which players engage with artificial conflicts. Here the player interacts with the system and the conflicts in the game are either engineering or artificial. In the game there are rules that aim to limit player behavior and determine the game. Games aim to entertain, usually games are widely liked by children to adults. Games are actually important for brain development, to improve concentration and train to solve problems precisely and quickly because in games there are various conflicts or problems that require players to solve them quickly and precisely.

Maze Games

According to Putri, Astrid Novita (2016:2). A maze game is a game of finding a way out, from several paths in an area of the game. The game template is in the shape of a square or rectangle whose size can be adjusted according to the user's wishes. Inside there are a series of paths in the form of branching labyrinths, but not every branch of the labyrinth is a way out because some are blocked by barrier walls.

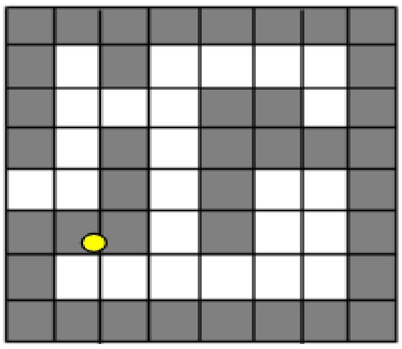


Figure 1. Example of a maze game view

Components of the Map Generator

Procedural Content Generation (PCG)

Linden, et al., (in Setyamurti, Wardhono, Afirianto, 2018:2417) explain that PCG is a way of creating content by following a certain method that allows content to be generated automatically. By using the appropriate generation method, various advantages can be obtained, including:

1. It can reduce the workload of artists and designers.
2. Increase the diversity of content.

Context Selection is a step that functions to select the part of the level to develop. ORE does not care about the method used in the selection of context as long as it produces one anchor. The easiest way to choose *a context* is to randomly select from all the *anchors* in a level to be raised

Research Methodology

Application Description

The application developed in this study is focused on the Map Generator making part so that the game creation for the application of this research is not built in a complex manner.

The Map Generator that will be built will function to make the work of a Game Developer, especially an Indie Game Developer, easier for them to build game scenes in less time than to create scenes completely manually.

The Map Generator developed can only process obstacles that have the same side angle where the shape that can be made by the Map Generator developed in this study is only a map with a maze shape.

Story Board

In the game to be built, a teenager is assigned as a treasure hunter in a maze for research purposes and food in the refugee camp. Where the world they inhabit has been polluted by a zombie outbreak, so later when searching for materials in the maze, these teenagers must avoid contact with zombies as much as possible so that they can survive in searching for the assigned materials.

A teenager (Player) walks and avoids contact with zombies in each level in the maze, later a teenager (Player) has to walk to find materials according to the mission given in each level, after the materials have been found the point to proceed to the next level will be opened.

When a teenager (Player) has reached the point of moving on to the next level, there will be a quiz with five questions where the questions contain questions about programming that are identical to the lessons of vocational students with the Software Engineering (RPL) education program.

In the game, each enemy character will be given a spotlight on the light, which later a teenager (Player) must avoid contact with the light. If the teenager (Player) touches the spotlight of the light, the light will turn red and within a few seconds of being in the point of light, the main character will be aware of his existence by the enemy and the mission will fail.

Then there will be several traps or objects that can kill the player, therefore a teenager (player) must avoid some of these dangerous objects.

Results

Hardware Implementation

The design of the application in this study requires several supporting hardware used in system development, namely computers with the following specifications:

1. AMD A8-7600 Radeon R7 processor, 10 compute cores 4C+6G 3.10Ghz.
2. Random Access Memory (RAM) 8 GB.
3. Dual Monitor 1600 X 900 pixels.
4. Keyboard and Mouse

5. Harddisk SSD Kingstone 120 GB.

Software Implementation

The software specifications used in system development are as follows:

Table 1. Software Specifications

Functional	Technology	Picture	Information
Software Game Engine Development	Multiplatform *tools* and *services* function to build *games* with a wide variety of different platforms such as 2D, 3D, VR, AR.	unity	Unity3d is the best *game engine* developed by unity *technologies* and is cross-platform. This *game engine* has the most complete features with a large community around the world.
Programming language	C# version 7.3 with support for .NET 4.6 *equivalent* as the framework.		C# is an object-oriented programming language that uses .NET as its framework where it is able to produce products in many forms, especially *game products*.
Script Editor	Visual Studio Code V.1.44.1		*A Script editor* capable of downloading the .NET framework as a reference package in C# programming languages.
OS	Windows 10 Professional	Windows 10 Professional	Windows 10 is used for application development because this operating system already supports the running of Unity3d applications. And this operating system is very stable for use in system development
Implementation Tools	Android		Android is used for implementation due to the general use of android in the daily environment.

Game Module Process Implementation

The implementation of this application process is carried out using Unity3D Game Engine and C *Sharp* (C#) as the programming language.

1. **Implementation of Cellular Automata & Fisher Yates Submodules on Map Generators.**

Implementation of map generation with Cellular Automata & Fisher Yates. This submodule provides the output of the map-forming numeric array. The output is then converted into string form into input for the Isometric Map Generator Submodule. The class used to implement the Cellular Automata Map Generator Submodule is the class in the MapGenerator.cs script. This class represents all object games that will be processed by the Map Generator built using the Cellular Automata & Fisher Yates method, in this class including mapping tile ground, obstacle, navmesh, navmask, map size, outline percent, tile size, seed, foreground/background color, etc..

2. **Generate Map in Class MapGenerator.cs**

This class is the first part of MapGenerator.cs coding, in this discussion the coding in the Map class Generator.cs will be divided into several parts to make it easier to understand. The coding in this section is used to generate the map generator. Here's the generate map coding.

Line 1-60 in Class MapGenerator.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class MapGenerator : MonoBehaviour
{
public maps;
public int mapIndex;
public Transform tilePrefab;
public transform[] obstaclePrefab;
public Transform navmeshFloor;
public Transform navmeshMaskPrefab;
public Vector2 maxMapSize;
[Range(0,1)]
public float outlinePercent;
public float tileSize;
List<Coord> allTileCoords;
Tail<Coord> shuffledTileCoords;
Tail<Coord> shuffledOpenTileCoords;
Transform[,] tileMap;
Map currentMap;
void OnNewWave(int waveNumber){
mapIndex = waveNumber -1;
GenerateMap();
}
public void GenerateMap(){
currentMap = maps[mapIndex];
tileMap = new Transform[currentMap.mapSize.x,currentMap.mapSize.y];
System.Random prng = new System.Random (currentMap.seed);
Generating Coords
allTileCoords = new List<Coord>();
for (int x = 0; x < currentMap.mapSize.x; x++){
for (int y = 0; y < currentMap.mapSize.y; y++){
allTileCoords.Add(new Coord(x,y));
}
}
shuffledTileCoords = new Queue<Coord> (Utility.ShuffleArray
(allTileCoords.ToArray(), currentMap.seed));

Create map holder object
stringholderName = "Generate Map";
if(transform. Find(holderName)){
DestroyImmediate(transform. Find(holderName).gameObject);
}

Transform mapHolder = new GameObject (holderName).transform;
mapHolder.parent = transform;
```

Figure 2. Map Generator line 1-60 script image

Testing Scenarios

The test scenarios of the Map Generator Module and Dynamic Levels in the stealth game are simple.

1. Testing Submodule Cellular Automata Map Generator.

   This section describes the testing process for the Cellular Automata & Fisher Yates Submodule on the Map Generator. The formation of the map will be carried out with ten different obstacle percent values . There are eight test variables in the formation of this map, including: Map Size, Obstacle Percent, Seed, Obstacle High Distance, Foreground & Background Color, Total Obstacle, Outline Percent, Tile Size.

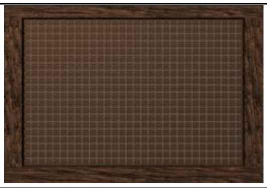
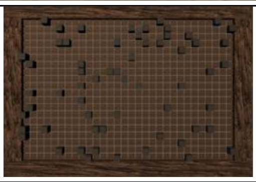
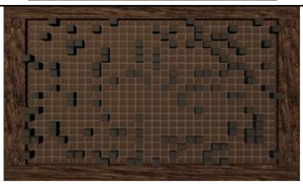
   In this test, a map of Cellular Automata & Fisher Yates will be made with a size of 30 x 20. In this test we will set it with the value seed=1 . The values in each verifiable to form the game map can be seen in the following table:

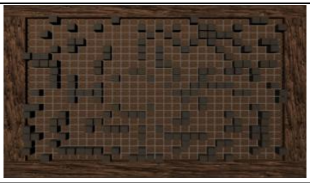
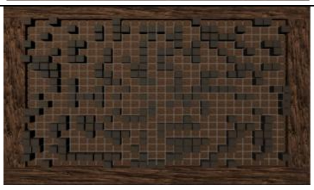
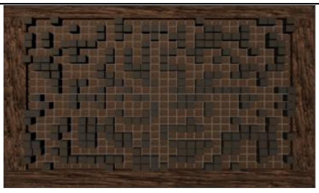
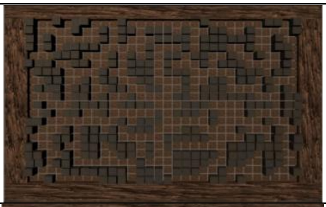
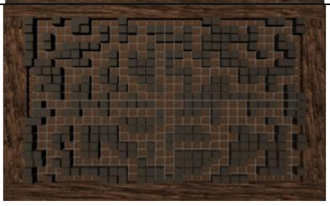
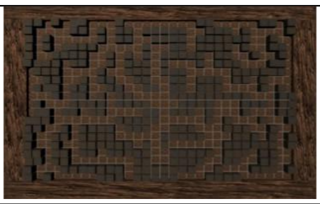
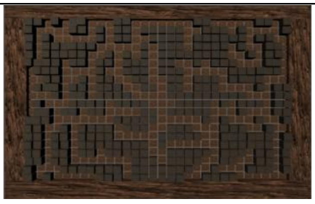
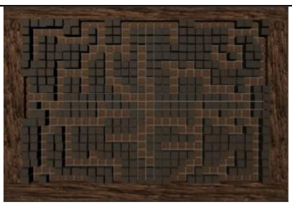
Table 2. Testing of Cellular Automata and Fisher Yates Submodules on Map Generator.

Testing	
Test Variables	**Value**
Map Size	X:30 x Y:20
Obstacle Percent	0% - 100%
Seed	1
Obstacle High Distance	2-3
Foreground & Background Colour	8E8E8E & B7B7B7 (Hexadecimal)
Total Obstacle	2 Elements
Outline Percent	0.1
Tile Size	2

The process of forming maps in each iteration with different *obstacle percent* values using the map generator submodule using the *Cellular Automata* & *Fisher Yates* method is as follows:

Table 3. Testing iteration stage I

Phase	**Result**
Initialization (Obstacle Percent = 0%)	
Iteration 1 (Obstacle Percent = 10%)	
Iteration 2 (Obstacle Percent = 20%)	

3 iterations (Obstacle Percent = 30%)	
Iteration 4 (Obstacle Percent = 40%)	
Iteration 5 (Obstacle Percent = 50%)	
Iteration 6 (Obstacle Percent = 60%)	
Iteration 7 (Obstacle Percent = 70%)	
Iteration 8 (Obstacle Percent = 80%)	
Iteration 9 (Obstacle Percent = 90%)	
Iteration 10 (Obstacle Percent = 100%)	

From the map produced by the *Cellular Automata* and *Fisher Yates* methods above, *the forming strings produced in the case of Test I are as follows:*

Table 4. Test map forming string I

The resulting string as a mapmaker in the case of Test I	
2#30#20> 121212100012000000021000002101100001102011021122021222100001101100001012021111021211101200102212220000010221122220100120202222100102020122211220020210221102102101210210012120002202211000021011101110000012122100211022022002200120212011110000121011022100220210221000020220002021222200002212212210001102000111102112002112102111101201102222200002101111100010002122100000002122100000102000200121022021210000011212201112210212011022111211022000002211110121012011012121001221102110110212011000000022102111001200210220001211212011122210010001110110200000212021211100221200000 1	
String length	600 Characters

Conclusion

In the process of the analysis, design, implementation, and testing stages, the following conclusions were obtained:

1. The ratio of the area of the game map area in the process of forming a *map generator* with the *Cellular Automata* and *Fisher Yates* methods greatly affects the shape of the resulting map.
2. By applying *the Fisher yates* method to *the map generator* generated by the *Cellular Automata method*, the mapping results are more complex, so that there is no part of *the ground* on the map that cannot be reached by *the player* later.
3. *The obstacles* generated on the maze-shaped map can only be tiled with the same side size, otherwise the resulting map will not have a complex maze shape.
4. The length of map data generated by the *map generator* is arguably limited because it can only store up to 2,147,483,647 map data with different map shapes.
5. In building the game *this Map Generator* is very helpful in creating game maps with very easy and fast map creation compared to building the entire map manually.

References

[1] Santoso, Erick, Gregorius Satia Budhi, and Rolly Intan, 2015. DESIGN OF EDUCATIONAL GAME APPLICATIONS USING WATERFALL MODEL, Garut Teacher Training and Education College Volume.3 Number 1, March 2017.

[2] Putri, Astrid Novita, 2016. OPTIMIZATION OF ALGHORITMA BREADTH FIRST SEARCH ON GAME ENGINE 3D THIRD PERSON SHOOTER MAZE BASED ON ANDROID INTELLIGENT AGENT, UNIVERSITY OF SEMARANG Volume.14 Number 1, July 2016.

[3] Setyamurti, Anindita, Wibisono Sukmo, Tri Afirianto, 2018. Implementation of Procedural Generation to Build Tactical RPG Level using the Occupancy Regulated Extension Method, Universitas Brawijaya Volume.2 Number 8, August 2018.

[4] Septiono, Dony Setiawan and Mussadun, 2016. LAND USE CHANGE MODEL TO SUPPORT FOREST MANAGEMENT UNIT MANAGEMENT PLAN (Case Study of KPH Yogyakarta) Universitas Brawijaya Volume.2 Number 8, 2018.

[5] Sulistya, Indah Erika and Febri Maspiyanti, 2018. LEGEND GAME OF THE ORIGIN OF LAKE TOBA BASED ON DESKTOP, Pancasila University Volume.9 Number 2, July 2018.

[6] Darmawan, Aspian and Mei P Kurniawan, 2016. DESIGN AND MANUFACTURE OF SURVIVAL HORROR GAME "I FOUND YOU" USING UNITY 3D GAME ENGINE STMIK AMIKOM, Yogyakarta, June 2016.
[7] Ginting, Budi Serasi, Fajar Ramadhan, 2018. ARTIFICIAL INTELLIGENCE-BASED BECOME A KING GAME DESIGN, STMIK Kaputama Binjai Volume.2 Number 1, April 2018.
[8] Dedi Nugraha, Sri Winiarti, 2014. DEVELOPMENT OF LEARNING MEDIA TRACKING SYSTEM IN MULTIMEDIA-BASED ARTIFICIAL INTELLIGENCE COURSE, AHMAD DAHLAN UNIVERSITY Volume.2 Number 1, February 2014.
[9] Rodianto, Eva Sapitri Andani, 2019. ACADEMIC ADMINISTRATION INFORMATION SYSTEM IN WEB-BASED TUTORING (CASE STUDY OF DILA SAMAWA), SUMBAWA UNIVERSITY OF TECHNOLOGY, VOLUME.1 Number 1, August 2019.
[10] Ade Sutedi, Cepy Slamet, Dhami Johar Damari, 2015. DESIGN AND BUILD AN OPEN ACCESS JOURNAL USING THE OBJECT-ORIENTED METHOD WITH A UML-BASED WEB ENGINEERING APPROACH. Garut College of Technology, Volume.12 Number 1, Garut 2015.
[11] Rosa A.S., M.Shalahuddin, 2014. Software Engineering, Informatics, Bandung 2013.
[12] Suendri, 2018. Implementation of UML (Unified Modelling Language) Diagram in the Design of Lecturer Remuneration Information System with Oracle University Database of Univer, State Islamic University of North Sumatra, Volume.03 Number 01, November 2018.
[13] Fikria, Balqis Kamalia, 2014. IMPLEMENTATION OF THE FISHER-YATES SHUFFLE ALGORITHM AS A RANDOMIZER OF NON-PLAYABLE CHARACTER (NPC) POSITIONS IN THE MARINE TANTRA GAME. Maulana Malik Ibrahim State Islamic University. Malang, November 2014.
[14] Hanggoro, Kridalukmana, Martono, 2015. Creation of Unity-Based "Clean Jakarta" Game Application, Faculty of Engineering, Diponegoro University, Volume.3 Number 4, October 2015.
[15] Gurupendidikan.co.id. (2021). Flowchart symbols. Retrieved March 09, 2021, from https://www.gurupendidikan.co.id/simbol-flowchart/