УДК 519.856:004.42:791.9
DOI: 10.25206/2310-4597-2023-1-3-9

ГЕНЕРАТОР ПСЕВДОСЛУЧАЙНЫХ ЧИСЕЛ В ИГРОВЫХ МЕХАНИКАХ

PSEUDORANDOM NUMBER GENERATOR IN GAME MECHANICS

Е. Г. Андреева, В. А. Молчалин
Омский государственный технический университет, г. Омск, Россия

E. G. Andreeva, V. A. Molchalin
Omsk State Technical University, Omsk, Russia

Аннотация. На рынке игровой индустрии большой процент занимают жанры, для которых генератор псевдослучайных чисел (ГПСЧ) является интегральной частью игровых механик. Получение случайных чисел также называют генерацией случайных чисел, что является процессом, который с помощью устройства или программы генерирует последовательность чисел или символов, которая может быть предсказана разумным образом только на основании случайности. В настоящее время случайные числа используются в данных областях предметных областях. Целью настоящей работы является разработка программного алгоритма, программы для исследования двух ГСПЧ – на основе линейного конгруэнтного метода и метода «Вихрь Мерсенна».

***Ключевые слова*:** случайное число, генератор случайных чисел (ГСЧ), генератор псевдослучайных чисел (ГПСЧ), игра, игровая механика, начальное значение (сид) ГСПЧ, Вихрь Мерсенна, линейный конгруэнтный метод, период ГПСЧ.

Abstract. In the gaming industry, a large percentage is occupied by genres for which the pseudo-random number generator (PRNG) is an integral part of game mechanics. Obtaining random numbers is also called random number generation, which is a process that, using a device or program, generates a sequence of numbers or symbols that can be reasonably predicted based on randomness alone. Currently, random numbers are used in these areas of subject areas. The purpose of this work is to develop a software algorithm, a program for studying two PRNGs - based on the linear congruent method and the Mersenne Vortex method.

Keywords: random number, random number generator (RNG), pseudo-random number generator (RNG), game, game mechanics, initial value (sid) of RNG, Mersenne vortex, linear congruent method, RNG period.

I. Введение

В игровой индустрии ГПСЧ используются повсеместно. Одним из примеров набирающих популярность игровых механик являются лутбоксы, которые представляют собой виртуальный предмет в компьютерных играх, при использовании которого игрок получает случайные виртуальные предметы различной ценности и назначения [1, 5].

Использование ГПСЧ характерно и для так называемых гача-игр (Gacha game) – это жанр компьютерных игр на оснгове механики гачапона, которая похожа на механику лутбоксов. Возможно название и гатя-игра или гатя-механика. Термин пришел из Японии, и эти игры достаточно популярны в Китае и Корее. В Японии гачапон – это торговый автомат с игрушками внутри, в котором игрушку можно выбрать случайным образом. В России аналогом гачапона является барабан с шариками в лотерейных механиках. В Японии же была создана первая игра на основе гача-механики – Dragon Collection [6].

Случайности в играх проявляются и в таком жанре игр как ММОРПГ. ММОРПГ или массовая многопользовательская ролевая онлайн-игра (Massively multiplayer online role-playing game, MMORPG, ММОРПГ) – компьютерная игра, в которой жанр ролевых игр совмещается с жанром массовых онлайн-игр.

Одной из ключевых механик ММОРПГ является выпадение предметов, согласно определенному шансу, установленному в генераторе псевдослучайных чисел игры. Предметы в ММОРПГ являются особенно важной составляющей игрового процесса, что, из-за случайного выпадения этих предметов с растущей редкостью при повышении уровня игры, создало ценность этих предметов для игроков [5].

Разрабатываемый генератор псевдослучайных чисел может применяться в работе многих игровых механик, основанных на этой игровой технологии.

Случайные числа важны и для научной работы. До создания аппаратных генераторов для получения случайных чисел бросались монеты, играли кости, вынимались шарики с цифрами из ящика. Но из-за механического характера этих методов генерация большого количества достаточно случайных чисел, что важно в статистических исследований, требовала много труда и времени. Электронные генератор случайных чисел на основе электронных компонентов появились в середине прошлого века, например, предложенный А.М. Тьюрингом.

В настоящее время случайные числа используются в предметных областях: социологические и исследования, математическое моделирование при обработке данных и процессов, криптография и информационная безопасность, принятие решений и оптимизация в экспертных системах, теория игр, и наконец, развлечения и игры [1].

Генераторы случайных чисел подразделяются на аппаратные генераторы случайных чисел, которые генерируют случайные числа в зависимости от текущего значения какого-либо атрибута физической среды, который практически сложно смоделировать, и генераторы псевдослучайных чисел, которые генерируют числа, которые выглядят случайными, но на самом деле являются детерминированными и могут быть воспроизведены, если известна математическая модель, на основании которой работает генератор псевдослучайных значений [8].

Числа, генерируемые с помощью этих ГСЧ, всегда являются псевдослучайными (или квазислучайными), то есть каждое последующее сгенерированное число зависит от предыдущего [1]:

$$r_{i+1} = f(r_{i-k+1}, r_{i-k+2}, \ldots, r_i).$$

Последовательности, составленные из таких чисел, образуют петли, то есть обязательно существует цикл, повторяющийся бесконечное число раз. Повторяющиеся циклы называются *периодами.* Такие ГСЧ не требуют больших объмов памяти, обладают достаточным быстродействие, однако основной недостаток – цикличность последовательности чисел генератора.

II. Постановка задачи

В большинстве своем, используются генераторы псевдослучайных чисел, которые называются алгоритмическими ГСЧ. Генератор псевдослучайных чисел (ГПСЧ) – это алгоритм, порождающий последовательность чисел, элементы которой почти независимы друг от друга и подчиняются заданному распределению, которое чаще всего является равномерным. Формирование последовательности случайных чисел ГПСЧ, начинается с определения начального значения (initial value, seed).

ГПСЧ используют для:

– разнообразия природы видео-игры,

– усложнения поведения мобов (нестатичный объект игры, затрудняющий движение игрока) и персонажей,

– генерации уровней,

– усложнения геймплея.

III. Теория

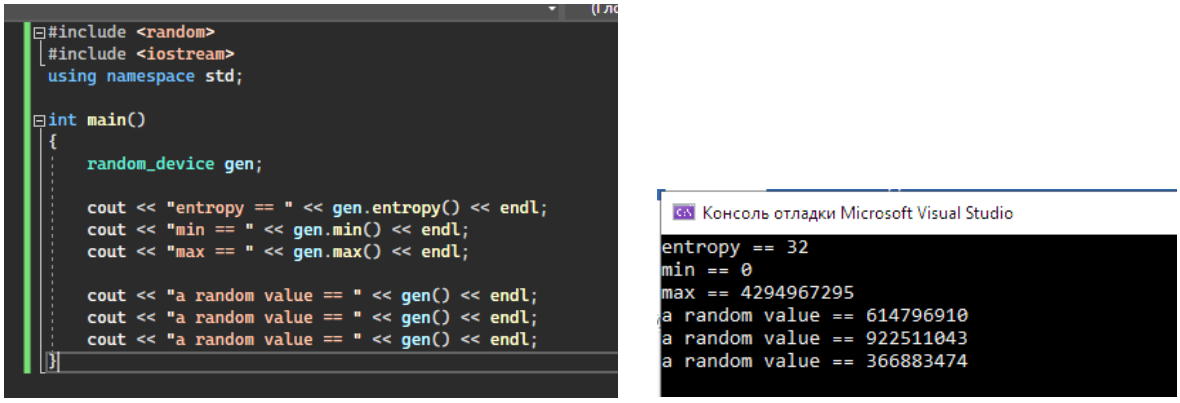
1. Способы получения начального значения для ГСПЧ

Все генераторы псевдослучайных чисел являются детерминированными, поэтому для того чтобы выдача чисел не повторялась, их состояние надо инициализировать какой-то уникальной последовательностью чисел. Программная ренализация ГПСЧ разрабатывалась на языке C++ в интегрированной среде Visual Studio. Для формирования базы случайных чисел в C++ можно использовать текущее время системы, например, команду srand(time(NULL)).

Также, сид (seed – стартовое число при генерации псевдослучайных чисел) можно выставить своими руками, задав какую-нибудь числовую последовательность.

Другим вариантом является класс random_device в C++. Этот класс описывает источник случайных чисел и согласно стандарту ГОСТ Р ИСО 28640-2012 [2] может быть недетерминированным или криптостойким (но это необязательно). В реализации в VisualStudio полученные значения являются недетерминированными и криптостойкими, но производительность ниже, чем у генераторов, созданных на основе механизмов и адаптеров механизмов.

Результаты выполнения random_device равномерно распределены в замкнутом диапазоне $[0, 2^{32}]$. В целом, random_device используется для получения начальных значений для других генераторов, созданных с помощью механизмов или адаптеров механизмов. На рис. 1, *а* приведен пример программного использования класса random_device.



```
#include <random>
#include <iostream>
using namespace std;

int main()
{
    random_device gen;

    cout << "entropy == " << gen.entropy() << endl;
    cout << "min == " << gen.min() << endl;
    cout << "max == " << gen.max() << endl;

    cout << "a random value == " << gen() << endl;
    cout << "a random value == " << gen() << endl;
    cout << "a random value == " << gen() << endl;
}
```

```
Консоль отладки Microsoft Visual Studio
entropy == 32
min == 0
max == 4294967295
a random value == 614796910
a random value == 922511043
a random value == 366883474
```

а) б)

Рис. 1. Пример использования в коде класса random_device в C++ (*а*)
и результаты использования кода (*б*) для генерации сида

На рис. 1,б показан результат работы кода, где entropy – это оценка стохатичности источника в битах.

2. Разработка и анализ генераторов псевдослучайных чисел

2.1. Линейный конгруэнтный метод (ЛКМ)

Члены последовательности вычисляются рекуррентно по формуле [1]

$X_{k+1} = (aX_k + c) \bmod m, n \geq 1$ (1) где ключевые целые числа: модуль m > 0, множитель $0 \leq a \leq m$, приращение $0 \leq c \leq m$, X_0 – начальное значение или сид, a mod m – взятие остатка по модулю некоторого натурального числа m.

```
void LMC(
    int Xo, int m, int a, int c,
    vector<double>& randomNums,
    int noOfRandomNums)
{
    randomNums[0] = Xo;

    for (int i = 1; i < noOfRandomNums; i++) {
        randomNums[i] = ((Xo * a) + c) % m;
        Xo = randomNums[i];
    }
}
```

Рис. 2. Функция, реализующая линейную
конгруэнтную последовательность на языке C++

Линейная конгруэнтная последовательность, определенная числами m, a, c и X_0 периодична с периодом, не превышающим m. ЛКМ дает статистически приемлемое распределение псевдослучайных чисел, но не является криптостойким из-за своей предсказуемости [1]. Реализация линейного конгруэнтного метода происходит

на языке C++ (рис. 2), и для нее взяты $X_0 = 0$, где в коде X_0 сохраняется как вектор randomNums, m = 6075, a = 106, c = 1283.

2.2. Вихрь Мерсенна

Вихрь Мерсенна использует свойства простых чисел Мерсенна и обеспечивает быструю генерацию высококачественных по критерию случайности псевдослучайных чисел.

Этот генератор используется во многих играх, среди которых особо примечательна серия игр Pokemon, где в четвертом поколении игр серии используется этот генератор. Из-за своего недостатка в криптостойкости, повсеместном использовании генератора в механиках игры и большом количестве фанатов, это привело к тому что образовался целый пласт игроков, которые создали программы и алгоритмы для манипуляции псевдослучайными числами игры и предсказаниями результатов механик.

Вихрь Мерсенна алгоритмически реализуется двумя основными частями: рекурсивной и закалки.

Необработанные последовательности, генерируемые рекурсией, обладают плохим равномерным распределением на больших размерностях. Чтобы это исправить, используется метод закалки, на выходе которого получается итоговая псевдослучайная последовательность.

```
unsigned long long genrand64_int64(void)
{
    int i;
    unsigned long long x;
    static unsigned long long mag01[2] = { 0ULL, MATRIX_A };

    if (mti >= NN)
    {
        if (mti == NN + 1)
            init_genrand64(5489ULL);

        for (i = 0; i < NN - MM; i++)
        {
            x = (mt[i] & UM) | (mt[i + 1] & LM);
            mt[i] = mt[i + MM] ^ (x >> 1) ^ mag01[(int)(x & 1ULL)];
        }
        for (; i < NN - 1; i++) {
            x = (mt[i] & UM) | (mt[i + 1] & LM);
            mt[i] = mt[i + (MM - NN)] ^ (x >> 1) ^ mag01[(int)(x & 1ULL)];
        }
        x = (mt[NN - 1] & UM) | (mt[0] & LM);
        mt[NN - 1] = mt[MM - 1] ^ (x >> 1) ^ mag01[(int)(x & 1ULL)];

        mti = 0;
    }

    x = mt[mti++];

    x ^= (x >> 29);
    x ^= (x << 17) & 0xD66B5EF5B4DA0000ULL;
    x ^= (x << 37) & 0xFDED6BE000000000ULL;
    x ^= (x >> 41);

    return x;
}
```

Рис. 3. Функция ГПСЧ «Вихрь Мерсенна»

В данной работе функция ГПСЧ была создана на языке C++ в 64-разрядной версии генератора. Сам генератор чисел вместе с алгоритмом закалки представлен на рис. 3.

IV. Результаты вычислительных экспериментов. Анализ генераторов псевдослучайных чисел

После разработки генераторов нужно удостоверится в их качестве, проанализировав их характеристики. Основные качественные требования к ГПСЧ – это большой период, эффективность, воспроизводимость, насколько близко к случайности выдаются числа. В данной работе анализируются генераторы через список методов [3]:

1. Оценка периода. После анализа периода и работы алгоритмов, будет дана оценка.

2. Скорость работы. Сравнивается скорость работы при генерации 100, а потом 1000 случайных чисел. Эта скорость работы будет считаться с помощью функции clock() и будет измеряться в миллисекундах [4];

3. Гистограмма распределения случайной величины. При построении гистограммы область значений случайной величины (a,b) разбивается на n сегментов, а затем подсчитывается количество попаданий величин в каждый сегмент. Используется MS Excel, где сгенерированные 1000 случайный чиселразмещаются на на интервале (0,1).

4. Анализ общепринятых недостатков.

5. Вывод по алгоритму.

Линейный конгруэнтный метод.

1. Период T ≤ m, что очень удобно, и при увеличении периода увеличивается время расчетов.

2. После выполнения кода, были получены результаты:

– скорость генерации 100 чисел для линейного конгруэнтного метода 227 мс;

– скорость генерации 1000 чисел для линейного конгруэнтного метода 2904 мс;

как видно при увеличении количества чисел для генерации, среднее время генерации увеличивается

3. Собрав 1000 чисел была создана гистаграмма (рис. 4):

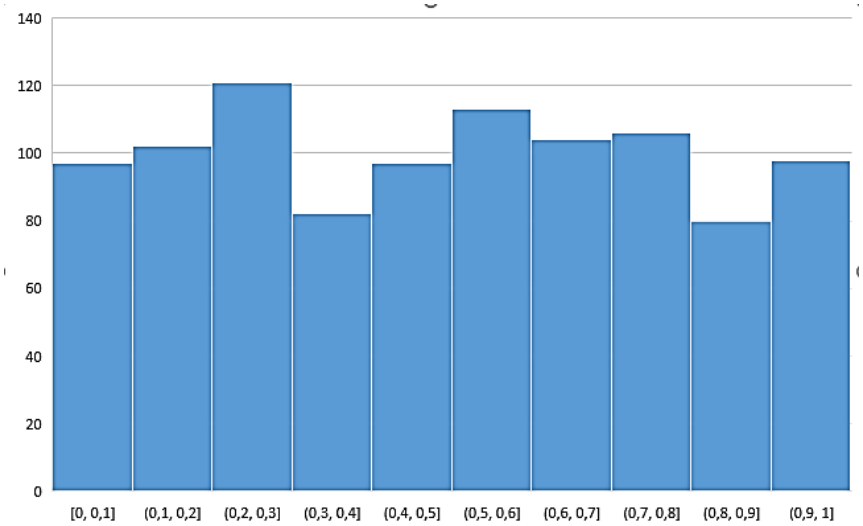


Рис. 4. Гистограмма выпадения случайных чисел
для линейного конгруэнтного метода

На гистограмме видно, что распределение чисел хоть и близко к равномерному, но все же не одинаково случайно.

4. Одним из основных недостатков ЛКМ является наличие операций умножения и деления, из-за чего достаточно простой алгоритм использует много ресурсов. Гистограмма так же показала, что не все биты одинаково случайны.

5. Алгоритм долгое время являлся очень популярным, но все же он является старым и в настоящее время заменен лучшими алгоритмами, которые имеют более высокую скорость или случайность.

Вихрь Мерсенна.

1. Период T = 2^{19937}–1 (для чего и подбирались парамеиры метода n, w и r так чтоб n·w–r было равно 19937), согласно числам Мерсенна. Размер w матрицы A предпочтительно выбирать равным размеру битового слова для ускорения работы на компьютерах, тем самым задавая диапазон [0, 2^{w}–1]. Период алгоритма является одним из самых больших среди множества алгоритмов

2. После выполнения кода, были получены результаты:

– скорость генерации 100 чисел для линейного конгруэнтного метода 345 мс;

– скорость генерации 1000 чисел для линейного конгруэнтного метода 2744 мс.

Как видно, хранение битов в памяти позволило увеличить среднюю скорость при выдаче 1000 чисел, т.е. данный

3. Собрав 1000 чисел была создана гистаграмма (рис. 5):

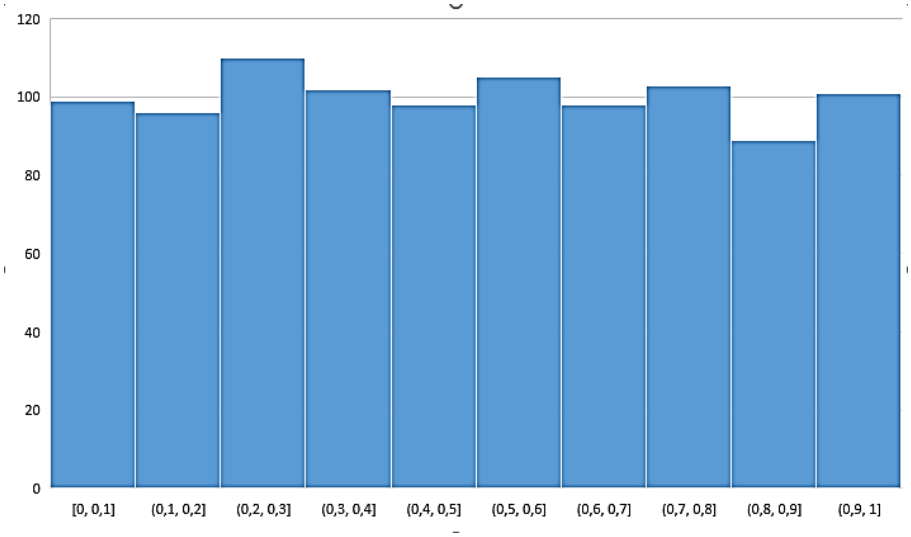


Рис. 4. Гистограмма выпадения случайных чисел для вихря Мерсенна

Распределение является наглядно равномерным, показывая что алгоритм закалки работает и вихрь Мерсенна выдает достаточно случайные числа на каждом интервале.

4. По сравнению с многими новыми алгоритмами вихрь Мерсенна требует относительно много времени на генерацию числа, из-за того свойства, что алгоритм сразу генерирует последовательность, а это не всегда удобно для программ. Также одним из недостатков многие специалисты называют то, что алгоритм излишен для использования в играх из-за огромного размера периода.

5. Вихрь Мерсенна до сих пор является одним из часто используемых алгоритмов, несмотря на свои недостатки, и использовался во множестве популярных играх. Большой период и качество случайности чисел делают его хорошим выбором, если скорость генерации одного числа не так важна.

V. Выводы и заключение

При разработке алгоритма ГПСЧ необходимо выполнить следующие условия [3, 7]:

– достаточно длинный период, гарантирующий отсутствие зацикливания последовательности в пределах решаемой задачи;

– эффективность, т.е. быстрота работы алгоритма и малые затраты памяти;

– желательно одинаковое функционирование на различном оборудовании и операционных системах, т. е. портируемость;

В работе были исследованы два известных алгоритма для генераторов псевдослучайных чисел: линейный конгруэнтный метод и вихрь Мерсенна. Исследования показали, что алгоритм Вихрь Мерсенна показал себя быстрее линейного конгруэнтного метода при генерации большого количества чисел, а также он дает более равномерное распределение, т.е. вихрь Мерсенна выдает достаточно случайные числа на каждом интервале.

Список литературы

1. Слеповичев И. И. Генераторы псевдослучайных чисел. Саратов: СГУ, 2017. 117 с. [Электронный ресурс]. URL: https://www.sgu.ru/sites/default/files/textdocsfiles/2018/07/09/slepovichev_i.i._generatory_psevdosluchaynyh_chisel_2017.pdf (дата обращения: 10.03.2022).

2. ГОСТ Р ИСО 28640-2012. Статистические методы. Генерация случайных чисел. М.: Статинформ, 2014. 40 с. [Электронный ресурс]. URL: https://docs.cntd.ru/document/1200096454 (дата обращения: 17.03.2022).

3. Григорьев А. Ю. Методы тестирования генераторов случайных и псевдослучайных последовательностей // Ученые записки УлГУ. Сер. Математика и информационные технологии. Электрон. журн. 2017, № 1. С. 22–28. [Электронный ресурс]. URL: https://ulsu.ru/media/uploads/anako09%40mail.ru/2017/10/14/%D0%93%D1%80%D0%B8%D0%B3%D0%BE%D1%80%D1%8C%D0%B5%D0%B2.pdf (дата обращения: 14.04.2022).

4. Как найти время работы программы на C++. [Электронный ресурс]. URL: http://cppstudio.com/post/468/ (дата обращения: 7.04.2022).

5. Что такое ГСЧ? Урок для геймеров. [Электронный ресурс]. URL: https://gadgetshelp.com/razvlecheniia/chto-takoe-gsch-urok-dlia-geimerov/ (дата обращения: 21.04.2022).

6. John Harris How classic games make smart use of random number generation [Электронный ресурс]. URL: https://www.gamedeveloper.com/programming/how-classic-games-make-smart-use-of-random-number-generation (дата обращения: 21.04.2022).

7. Recommendation for Random Number Generation Using Deterministic Random Bit Genera-tors. [Электронный ресурс]. URL: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf (дата обращения: 26.04.2022).

8. Кнут Д. Э. Искусство программирования. Получисленные методы. Т. 2. М.: Изд. дом «Вильямс», 2013. 832 с. [Электронный ресурс]. URL: https://www.williamspublishing.com/Books/sci_Knuth2.html (дата обращения: 11.02.2022).