

PocketMap

Anita Jamroży

Szymon Szymiczek



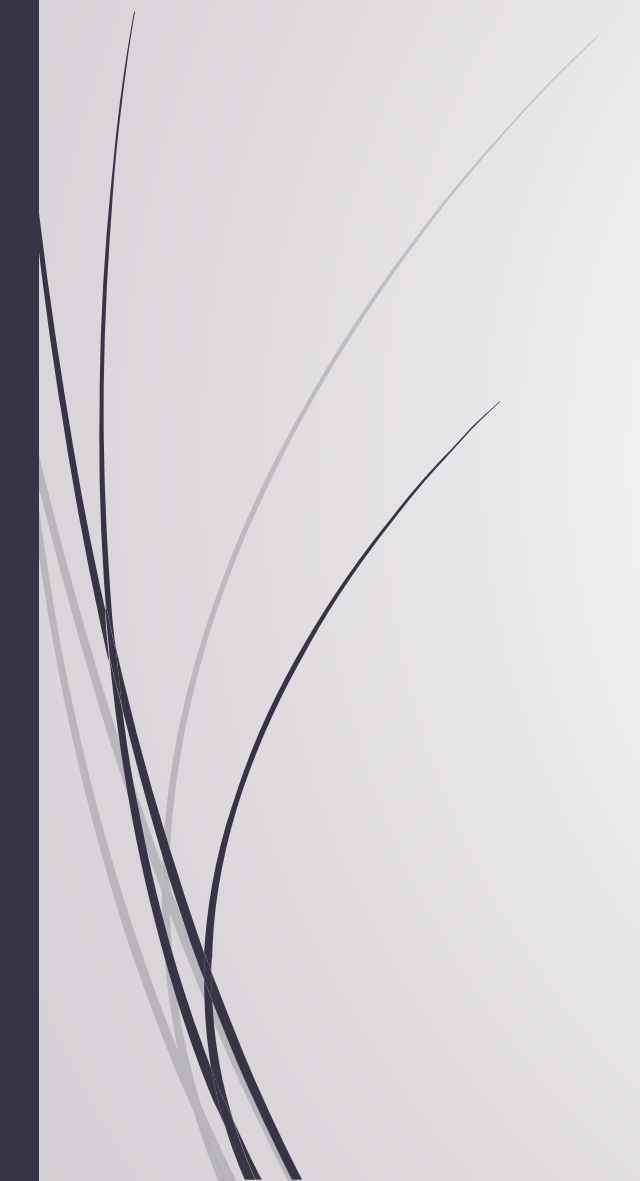
Plan prezentacji



1. Temat
2. Implementacja
3. Generator liczb pseudolosowych
4. Testy generatorów liczb pseudolosowych
5. Zadania



Temat

- Świat tak duży, że nie da się go przechować
 - Odpowiedź na potrzeby rpgowca
- 



Generowanie Proceduralne

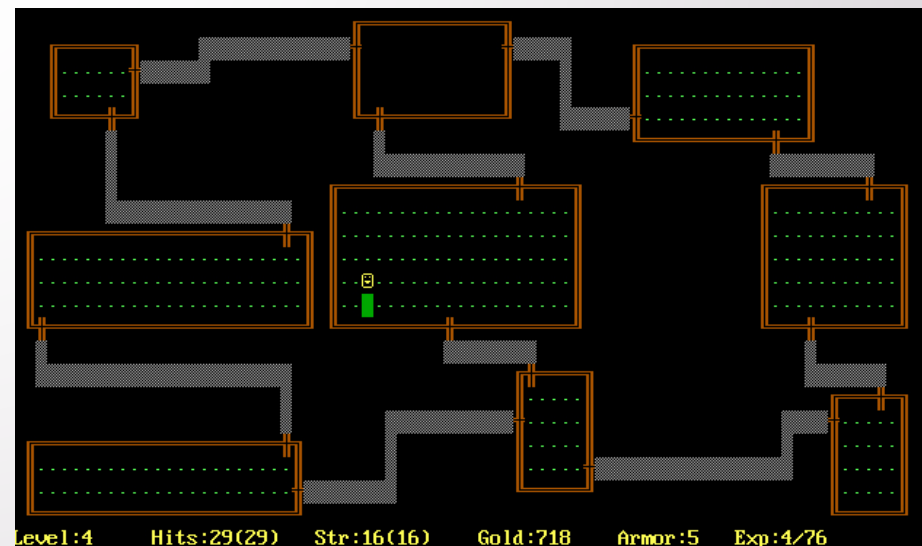
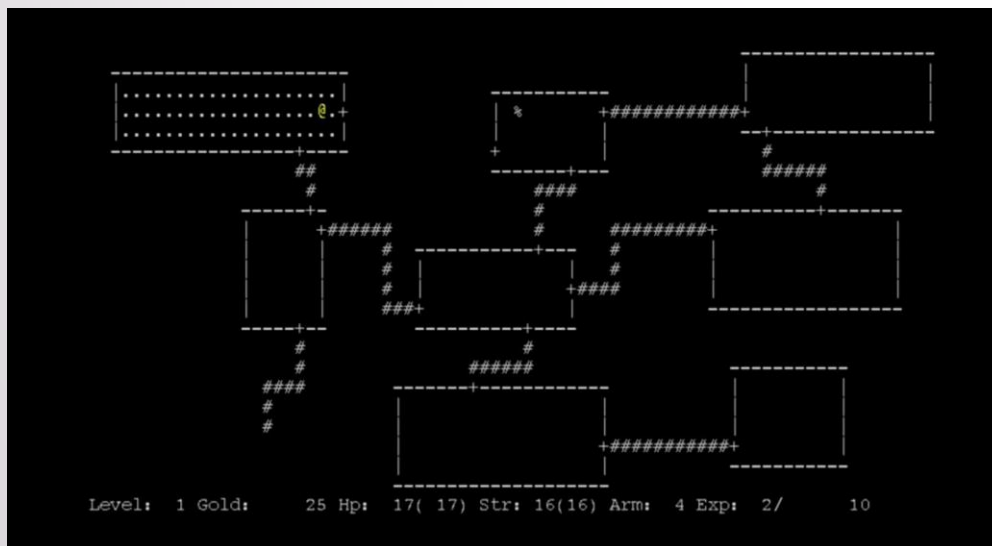


- ▶ Element nazywany jest wygenerowanym proceduralnie jeżeli został stworzony automatycznie przez algorytmy komputerowe. Jako prosty przykład można przyjąć funkcję rysującą prostokąt w lewym górnym rogu ekranu.
- ▶ Prostokąt niegenerowany proceduralnie będzie najczęściej grafiką wczytywaną z pliku

Generowanie proceduralne

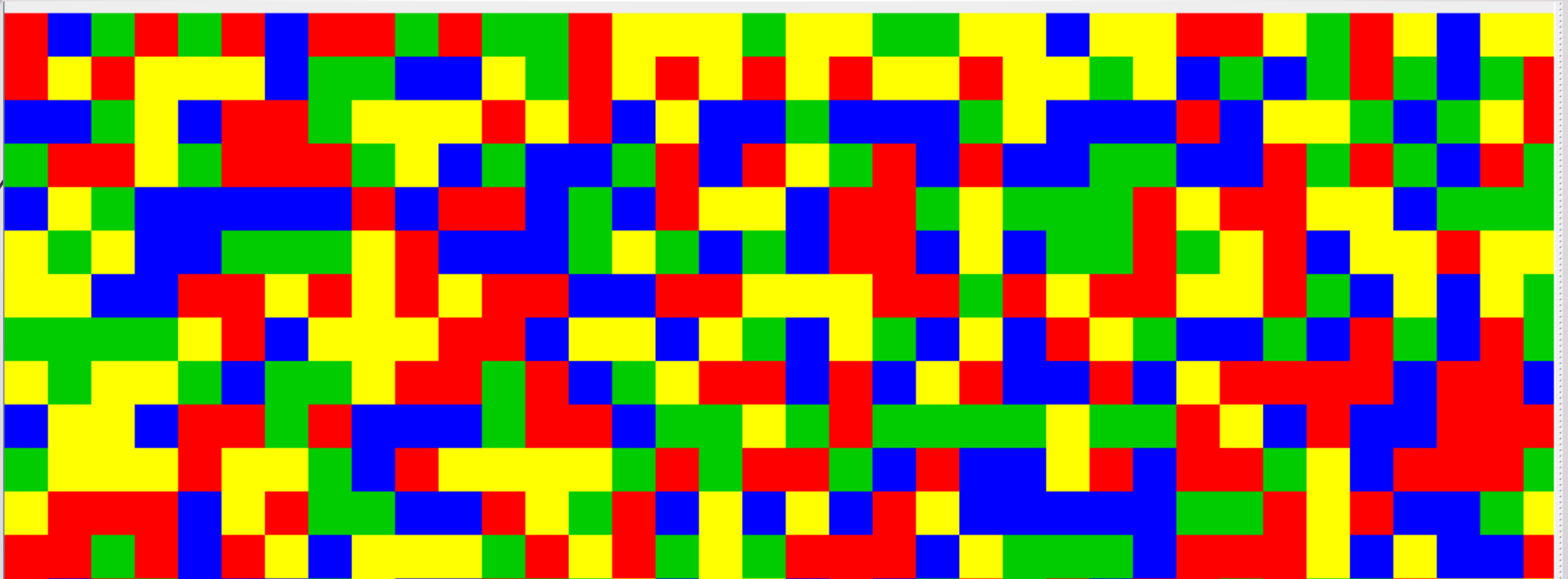
Sposoby użycia generowania proceduralnego:

- Generujemy mapę raz, by przechować na dysku i odczytywać w razie potrzeby
- Minecraft, Terraria
- Generujemy mapę wraz z eksploracją świata, odkrytą przestrzeń przechowujemy - No Man's Sky i jego 18 446 744 073 709 551 616 planet
- Generujemy mapę jednorazowo na potrzeby rozgrywki - Gry RougeLike; The Binding Of Isaac, Spelunky



Implementacja

- Nasze użycie generowania proceduralnego -
Generowanie mapy całkowicie "w biegu", bez przechowywania





Generatory Liczb Pseudolosowych

- Określane przez jakość generowanych liczb oraz szybkość
- Główna wada:

Generatory stworzone są tak aby przy podaniu ziarna generować ciąg liczb o właściwościach ciągu losowego. Jednak często gdy podajemy ciąg ziaren które nie są losowe to ciąg pierwszych wygenerowanych liczb z tych ziaren też nie będzie losowy.

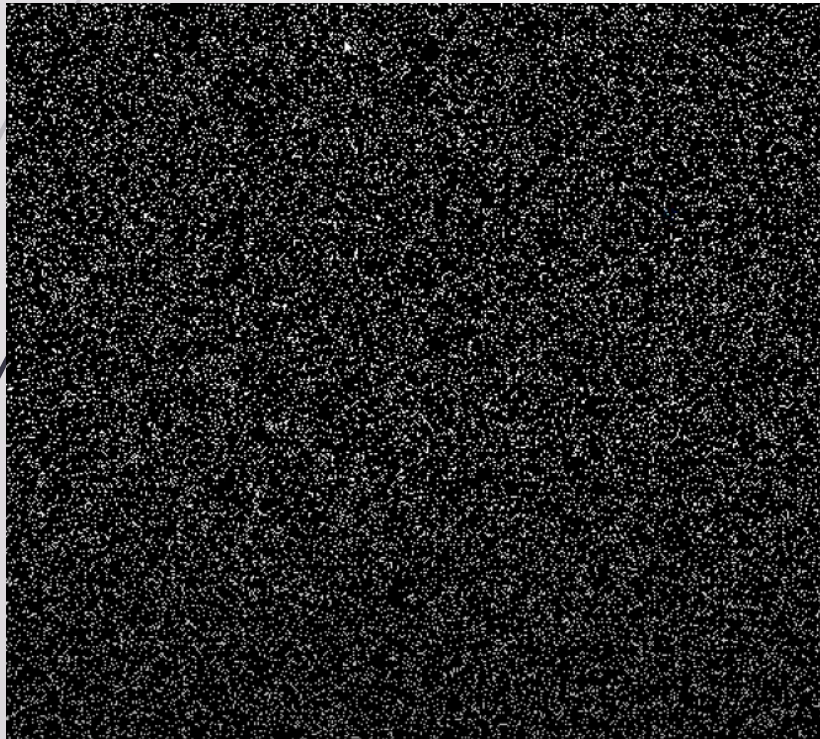
Generatory Liczb Pseudolosowych

function(21) → 102	function (1) → 2
57	function (2) → 7
2137	function (3) → 17
24	function (4) → 32
1001	function (5) → 52
420	function (6) → 77
8	function (7) → 107

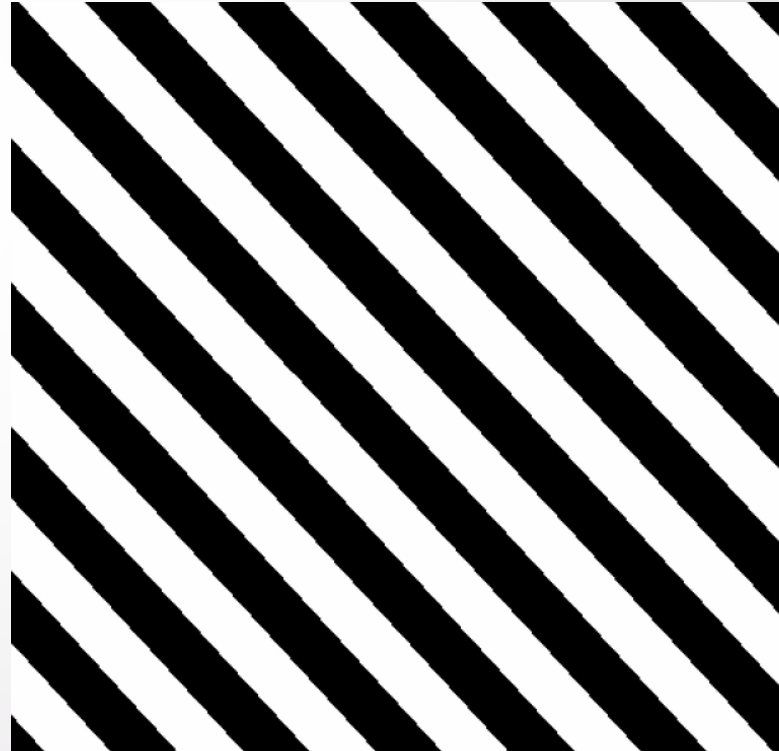
Różnica w wygenerowanych ciągach liczb dla różnych podejść

Generatory Liczb Pseudolosowych

Podejście 1:



Podejście 2:



Implementacja

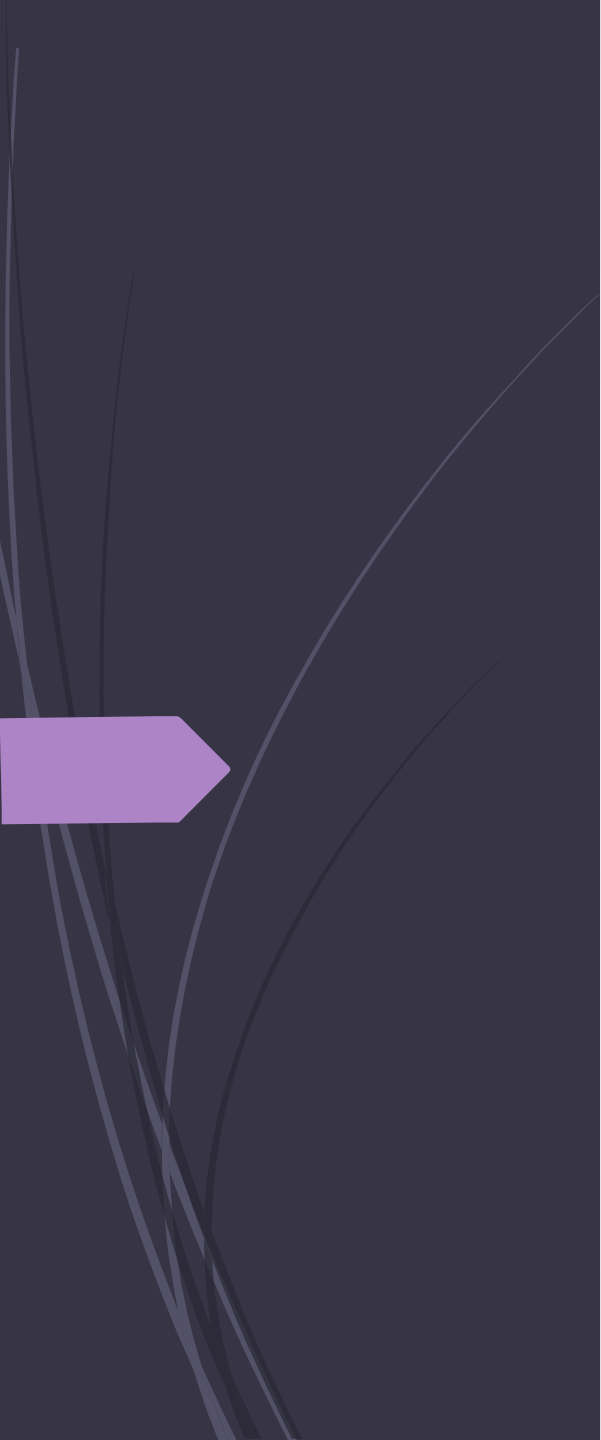
➤ Siatka mapy:

	9	0 9	1 9	2 9	3 9	4 9	5 9	6 9	7 9	8 9	9 9
	8	0 8	1 8	2 8	3 8	4 8	5 8	6 8	7 8	8 8	9 8
	7	0 7	1 7	2 7	3 7	4 7	5 7	6 7	7 7	8 7	9 7
	6	0 6	1 6	2 6	3 6	4 6	5 6	6 6	7 6	8 6	9 6
	5	0 5	1 5	2 5	3 5	4 5	5 5	6 5	7 5	8 5	9 5
	4	0 4	1 4	2 4	3 4	4 4	5 4	6 4	7 4	8 4	9 4
	3	0 3	1 3	2 3	3 3	4 3	5 3	6 3	7 3	8 3	9 3
	2	0 2	1 2	2 2	3 2	4 2	5 2	6 2	7 2	8 2	9 2
	1	0 1	1 1	2 1	3 1	4 1	5 1	6 1	7 1	8 1	9 1
	0	0 0	1 0	2 0	3 0	4 0	5 0	6 0	7 0	8 0	9 0
		0	1	2	3	4	5	6	7	8	9

Implementacja

➤ Siatka mapy:

+2													
9	8 11	9 11	10 11	11 11	12 11	13 11	14 11	15 11	16 11	17 11	18 11		
8	8 10	9 10	10 10	11 10	12 10	13 10	14 10	15 10	16 10	17 10	18 10		
7	8 9	9 9	10 9	11 9	12 9	13 9	14 9	15 9	16 9	17 9	18 9		
6	8 8	9 8	10 8	11 8	12 8	13 8	14 8	15 8	16 8	17 8	18 8		
5	8 7	9 7	10 7	11 7	12 7	13 7	14 7	15 7	16 7	17 7	18 7		
4	8 6	9 6	10 6	11 6	12 6	13 6	14 6	15 6	16 6	17 6	18 6		
3	8 5	9 5	10 5	11 5	12 5	13 5	14 5	15 5	16 5	17 5	18 5		
2	8 4	9 4	10 4	11 4	12 4	13 4	14 4	15 4	16 4	17 4	18 4		
1	8 3	9 3	10 3	11 3	12 3	13 3	14 3	15 3	16 3	17 3	18 3		
0	8 2	9 2	10 2	11 2	12 2	13 2	14 2	15 2	16 2	17 2	18 2		
	0	1	2	3	4	5	6	7	8	9	+8		



Testy Generatorów Liczb Pseudolosowych (pRNG tests)



Podział testów

- ▶ **Testy empiryczne** – NIE potrzebujemy wiedzieć na jakiej zasadzie działa RNG, a jedynie potrzebujemy sekwencji, którą wygenerował;
- ▶ **Testy teoretyczne** – potrzebujemy dogłębnej wiedzy, jak działa generator, ale w części przypadków nawet nie musi generować sekwencji. Największym ich minusem jest fakt, że jest ich stosunkowo niewiele.

Test Chi-kwadrat

$$\chi^2 = \sum_r \frac{(f_i - np_i)^2}{np_i}$$

gdzie: χ^2 - test chi-kwadrat;

f_i - liczba zaobserwowanych wartości z danego przedziału;

np_i - liczba jednostek (n), które powinny znaleźć się w danym przedziale (wartości oczekiwane przedziałów)



Hipoteza zerowa

„Zakładamy, że różnica między analizowanymi parametrami lub rozkładami wynosi zero.”

Przykład:

Rzucamy wiele razy monetą. Założmy 50. Zakładamy, że ilość wypadniętych orłów i reszek jest sobie równa i wynosi 25, ponieważ przy każdym rzucie prawdopodobieństwo wypadnięcia orła i reszki jest sobie równe i wynosi $\frac{1}{2}$.

Jak zinterpretować podane wyniki?

Jedną z zaproponowanych metod jest zaliczenie testu na podstawie tych zależności:

- ▶ Jeśli χ^2 mniejsze od 1% danych wejściowych albo większe od 99% tych samych danych, to nie można zakwalifikować tych danych jako losowe.
- ▶ Jeśli χ^2 znajduje się w 1-5% danych wejściowych albo 95-99% danych wejściowych, to jest losowość jest możliwa.
- ▶ Jeśli χ^2 znajduje się w 5-10% danych wejściowych (potrzebna jest tu bardziej szczegółowa tabela) albo w 90-95%, to losowość jest prawie możliwa.



Dlaczego napisanie samodzielnie testu jest takie trudne?

Potrzebujemy bardzo szerokiej wiedzy z zakresu statystyki, najczęściej niemożliwe jest, aby uzyskać ją w tak krótkim czasie.



Gotowe testy generatorów liczb losowych



Diehard

TestU01

Diehard

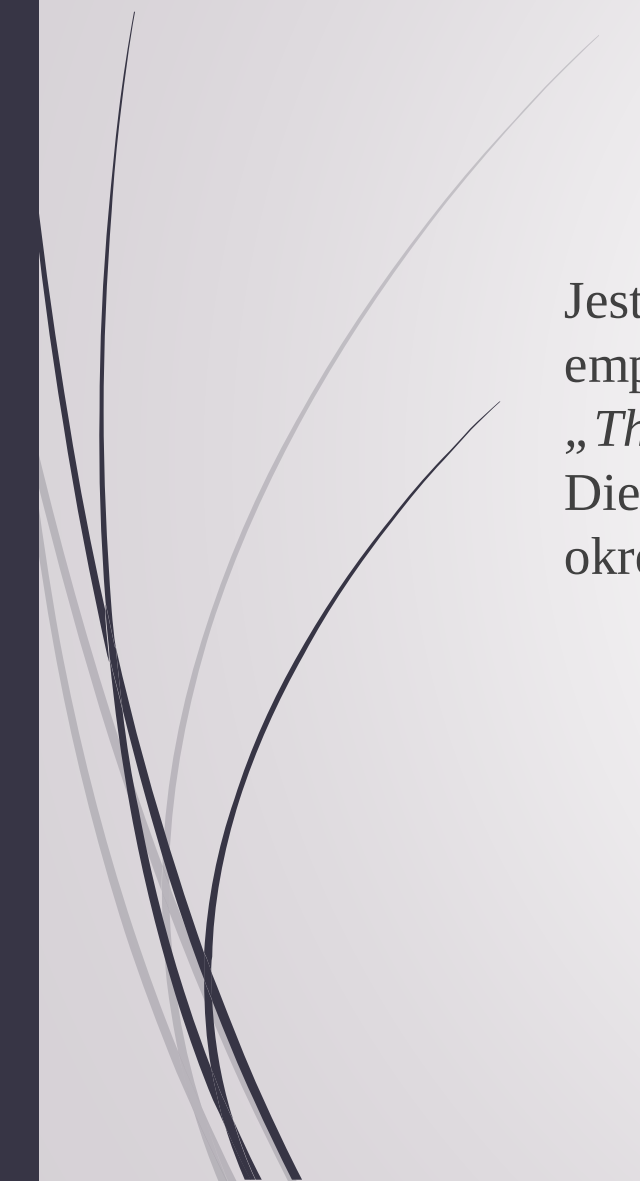
► Jest to zbiór 15 testów statystycznych. Ich wynikiem jest statystyczna wartość p (p-value), którą można interpretować jako dowód na brak losowości. Testowane są 32-bitowe liczby całkowite (integers). Minusem testu jest potrzeba dużej liczby próbek. Podczas zadania plik zawierający wszystkie te liczby zajmuje dużo miejsca oraz jego czas generowania jest dosyć długi.

► Przykładem testu w ramach Diehard jest ***Birthday test***:

Wybierz m urodzin w roku n dni (n musi być dość duże - większe niż 2^{18}). Niech J będzie liczbą wartości, które występują na liście więcej niż raz. J jest również znany z asymptotycznego rozkładu Poissona ze średnią ($m^2 / 4n$). Test Chi-kwadrat może porównać próbkę PRNG J ze znany rozkładem J .



TestU01



Jest to biblioteka napisana w języku C, która również implementuje zbiór empirycznych testów dla RNG. Testy są napisane na podstawie książki D. Knutha „*The Art of Computer Programming*”. Testy pokrywają się częściowo z tymi z Diehard. Razem z testami możliwa jest również szczegółowa dokumentacja określająca zasoby potrzebne do przeprowadzenia zestawu testów.



Zadanie 1

- Użyj w klasie MapPanel w metodzie getIcon metodę która wypełni mapę czterokolorową mozaiką.
- Zmieniając w TerrainManager metodę getMosaicIcon przetestuj wszystkie generatory liczb losowych.
- Jako odpowiedź podaj, który generator najlepiej sprawdza się w aplikacji PocketMap oraz co jest główną wadą uniemożliwiającą użycie pozostałych generatorów.



Zadanie 2

Celem ćwiczenia jest napisanie testu pokerowego. Program ma za zadanie na podstawie testu chi-kwadrat powiedzieć, czy wpisane słowo/zdanie zawiera przypadkowe litery.

Waszym zadaniem jest:

Uzupełnić funkcje *chiSquared* oraz *expectedValues* w klasie Zadanie2. Dla *chiSquared* jest to implementacja wzoru na chi kwadrat, a do *expectedValues* pętla, która zwraca `ArrayList<Double> expectedValues`.

Uwaga: W pakiecie resources znajduje się dokładny opis działania programu. Szukanym plikiem jest `zad2Guide.pdf`



Zadanie 3

- 1) Włączyć program dieharder i uruchomić wszystkie testy (-a) na pliku inputFile.txt po wykonaniu testu o nazwie *diehard_craps* wyłączyć program wpisując Ctrl-C. Jako odpowiedź podać: czy wszystkie testy zostały zdane? Jeśli nie, to które nie zostały.
- 2) Opisać, czym jest p-value – na co wskazuje, czy lepsza jest duża czy niewielka wartość, czy jednoznacznie określa, że liczby są losowe?
- 3) Opisać, czym są /dev/random i /dev/urandom oraz czym się różnią.



Bibliografia



- <http://www-users.math.umn.edu/~garrett/students/reu/pRNGs.pdf>
- <https://edu.pjwstk.edu.pl/wyklady/adn/scb/wyklad7/w7.xml>
- <https://www.youtube.com/watch?v=WXPBoFDqNVk>
- https://en.wikipedia.org/wiki/Diehard_tests
- <http://theurbanengine.com/blog//the-diehard-tests>
- <https://dl.acm.org/doi/10.1145/1268776.1268777>
- <https://www.youtube.com/watch?v=ZZY9YE7rZJw&t=2108s>
- https://en.wikipedia.org/wiki/Lehmer_random_number_generator
- <https://towardsdatascience.com/poker-test-pattern-based-random-number-detection-in-python-ec2dba4955bc>