

Generator liczb losowych w pętli ...

Zacznijmy od prostego programiku:

```
float a=random(1.0); //Liczba losowa z zakres 0..1
                        //Oczywiście zakres można zmienić
println(a); //Wypisana na konsole
```

Który uruchomicie kilka razy...

Co widać?

Zamiast `random(1.0)`; może być też `random(2.0)`; albo `random(10.0)`;

Idea pozostaje ta sama. Początkiem zakresu będzie 0 a końcem podana liczba.

Można też podać początek i koniec zakresu, wtedy wywołanie funkcji będzie wyglądać tak:

```
float a=random(1.0,5.0); //Liczba losowa z zakres 1..5
```

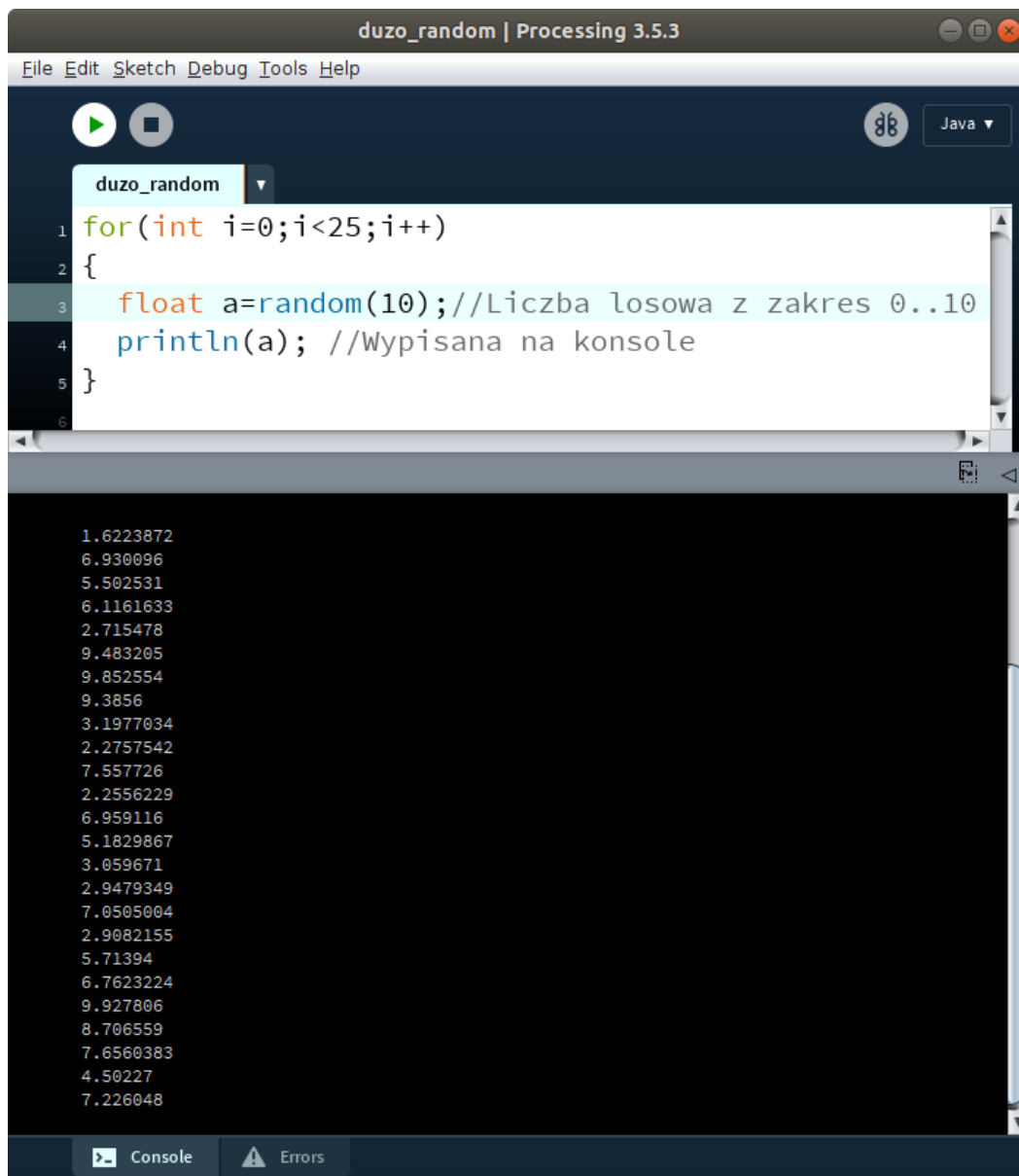
A co się stanie gdy podamy zakres -5.0 do -1.0?

A od 5 do 1 ?

No tak :-)

Ale warto było sprawdzić :-D

Funkcja `random()` służy właśnie do tego aby uzyskać w każdym wywołaniu inną liczbę losową. No tak właściwie to prawie losową, albo mówiąc w języku informatyki "pseudolosową". W rzeczywistości taki ciąg liczb generowanych przez funkcję powtarza się po odpowiednio dużej liczbie losowań. Ale jak na razie nie musimy się tym przejmować. Nie będziemy jej "wołać" znowu aż tak wiele razy:



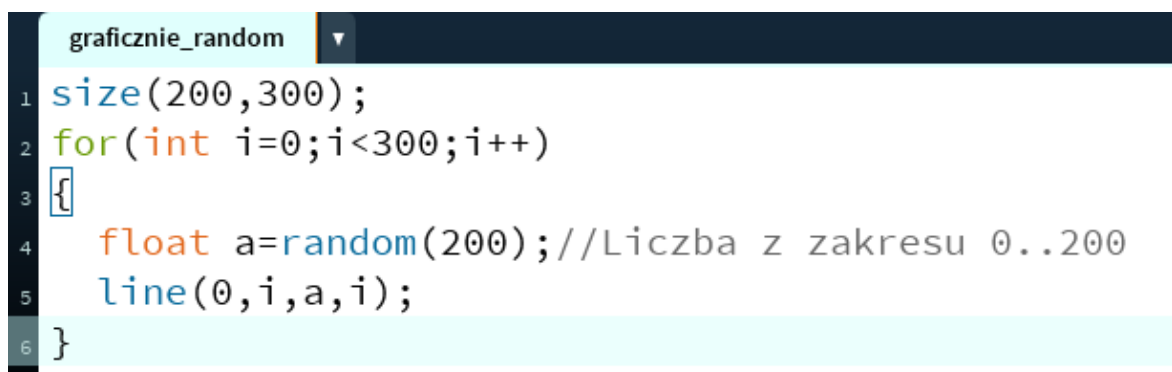
```
duzo_random | Processing 3.5.3
File Edit Sketch Debug Tools Help

duzo_random
1 for(int i=0;i<25;i++)
2 {
3   float a=random(10);//Liczba losowa z zakres 0..10
4   println(a); //Wypisana na konsole
5 }
6

1.6223872
6.930096
5.502531
6.1161633
2.715478
9.483205
9.852554
9.3856
3.1977034
2.2757542
7.557726
2.2556229
6.959116
5.1829867
3.059671
2.9479349
7.0505004
2.9082155
5.71394
6.7623224
9.927806
8.706559
7.6560383
4.50227
7.226048

Console Errors
```

No to teraz przekształćmy to na grafikę. Zaprezentujemy liczby losowe jako długość linii. Żeby było lepiej widać co jest co, użyjemy asymetrycznego okna 200,300.



```
graficznie_random
1 size(200,300);
2 for(int i=0;i<300;i++)
3 {
4   float a=random(200);//Liczba z zakresu 0..200
5   line(0,i,a,i);
6 }
```

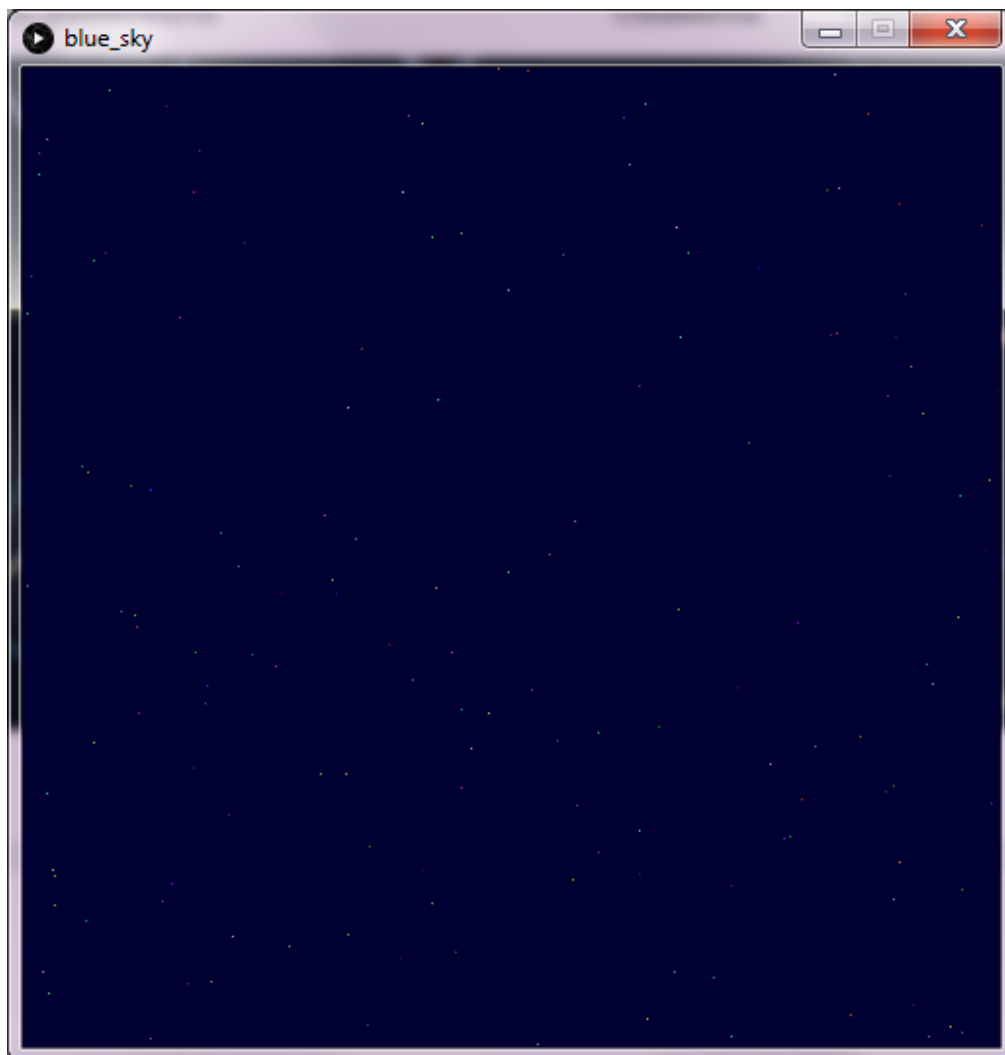
A teraz używając liczb losowych ustawmy sobie kolor każdej z linii:

```
stroke(random(256),random(256),random(256));
```

I zmieńmy orientację linii z poziomej na pionową:



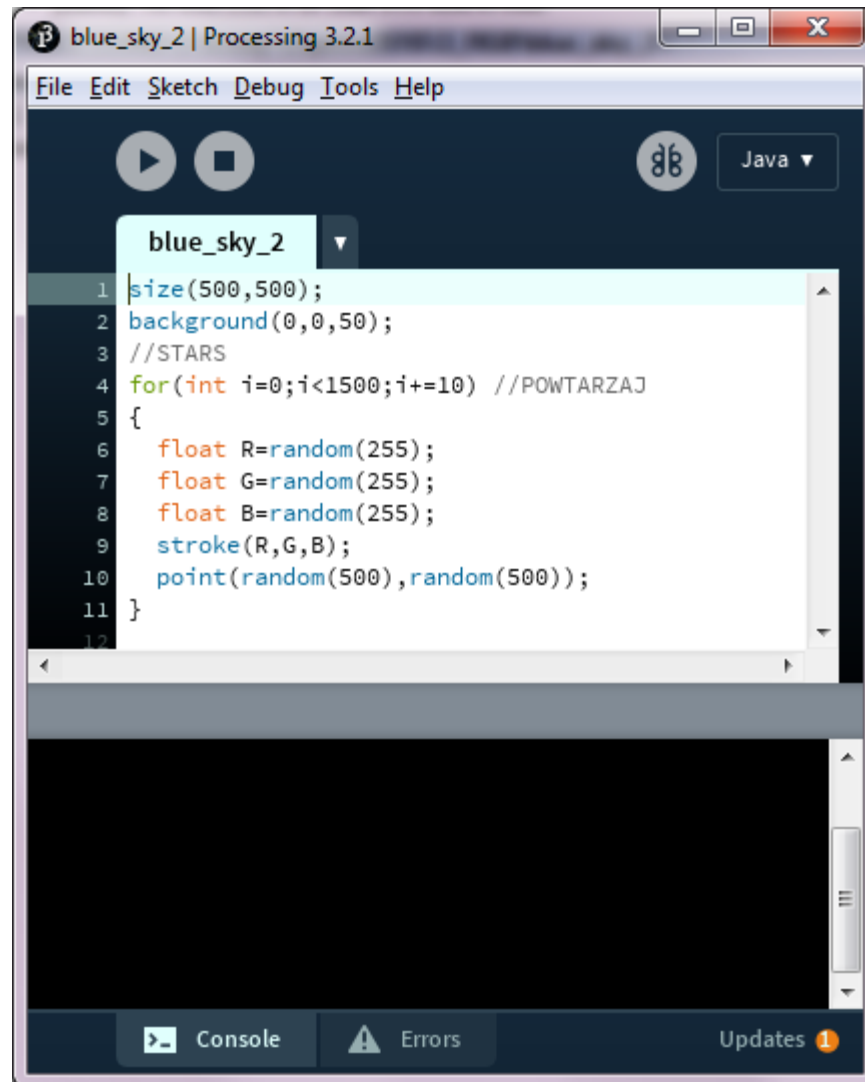
A jakby użyć punktów czyli `point(x,y)` i innego koloru tła...



Tak wygląda rozgwieżdżone niebo w Processingu. Program jest bardzo prosty, może sami go napisać? Podpowiadam tylko słowa kluczowe:

`#for #stroke #point #random` i jeszcze oczywiście `#size #background`

Oto kod programiku rysującego to rozgwieźdżone niebo:

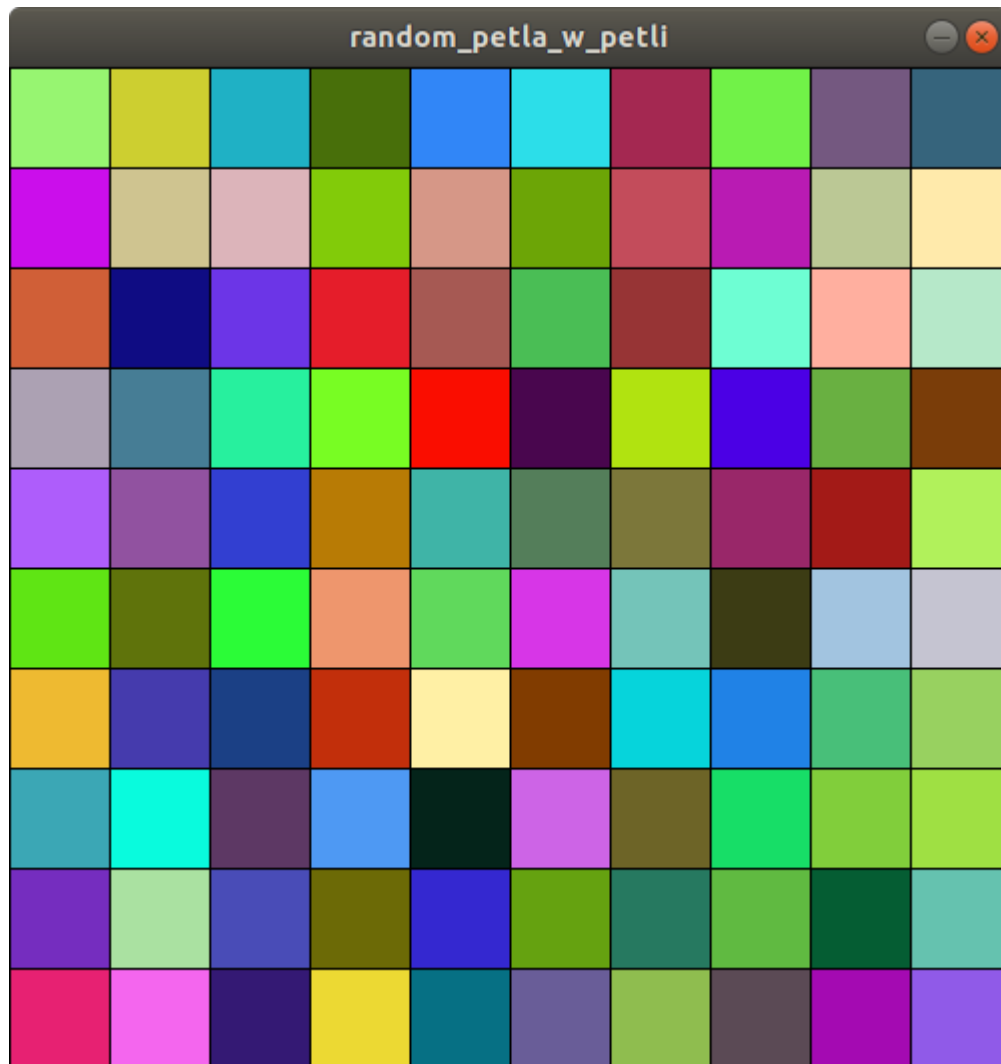


To o ile zwiększamy w pętli i jest prostym sposobem na regulację liczby gwiazd.

[Pętla, funkcja random, oraz kolorowe punkty czyli "rozwieźdżone niebo"](#)

Pętla w pętli

Gdybyśmy chcieli namalować sobie taki obrazek, to wykonanie go pojedynczą pętlą byłoby niezwykle trudne:

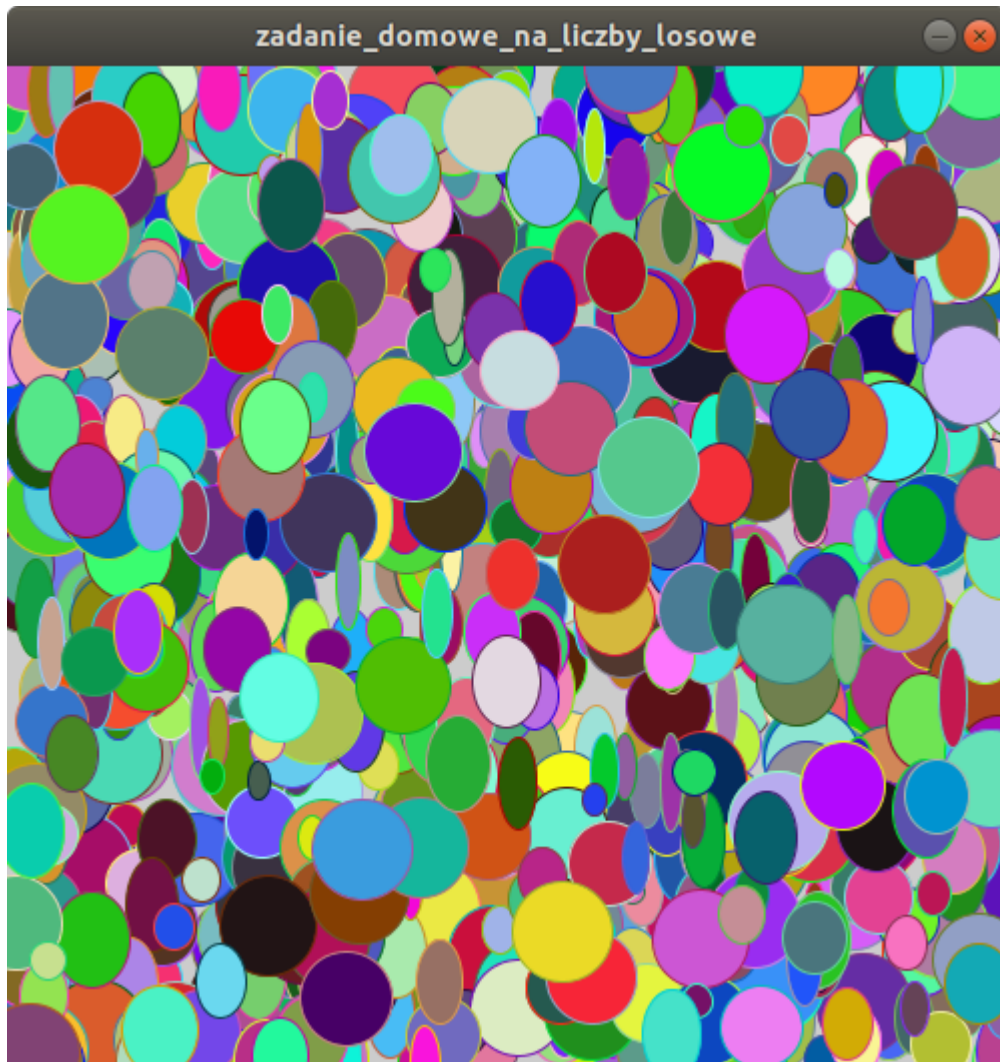


Natomiast używając pętli która ma we wnętrzu drugą pętlę dajemy sobie z tym zadaniem radę w 9 liniach prostego (?) kodu, który rysuje nam 100 różnokolorowych kwadratów. Oczywiście rozmiar kwadratu trzeba dopasować do rozmiaru okna, żeby wszystkie mogły się zmieścić.

```
random_petla_w_petli
1 size(500,500);
2 for(int i=0;i<10;i++)
3 {
4   for(int j=0;j<10;j++)
5   {
6     fill(random(256),random(256),random(256));
7     rect(i*50,j*50,50,50);
8   }
9 }
```

Spróbujcie trochę z tym programem poeksperymentować. Np. podmieńcie mnożniki przy *i* i *j*, albo kwadraty zamieńcie na elipsy. Do zagadnienia nie raz wrócimy.

Zadanie domowe na liczby losowe:



Zauważcie że elipsy mają zróżnicowane zarówno wypełnienie jak i obrzeża, żadna nie jest bardzo mała, a także nie są zbyt duże w stosunku do okna.

I jest jeszcze jedno ograniczenie w tym zadaniu które powinniście odgadnąć uważnie patrząc na ten zbiór elips.