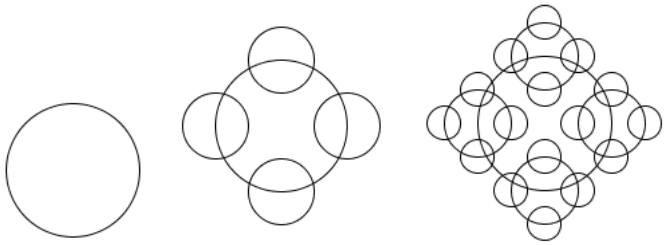
Laboratorium 5b – Processing c.d.

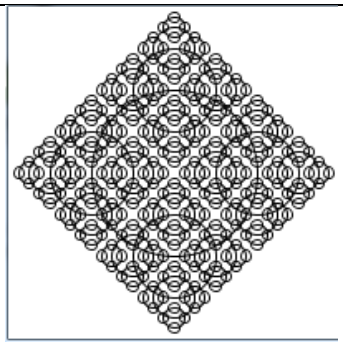
Wstęp

Laboratorium 5b poszerza zagadnienie generowania i przetwarzania obrazów z wykorzystaniem języka Processing 3, dedykowanego do obróbki grafiki.

Ćwiczenie 1 – Rekurencja

Zadanie: Narysuj prosty fraktal kołowy	

Zbuduj algorytm rysowania fraktala kołowego	Fraktal kołowy polega na rysowaniu koła a kolejnych poziomów, gdzie kolejny poziom polega na narysowaniu 4 kół o promieniu dwukrotnie mniejszym niż koło poprzedniego poziomu i środkach rozmieszczonych równomiernie po krawędzi koła poprzedniego poziomu
	 
	Algorytm rysujący jest dość prosty i pobiera jako parametry współrzędne i promień pierwszego koła oraz minimalną wielkość najmniejszego koła (niezbędną do zakończenia rekurencyjnego wywoływania funkcji).
	`void rysujKolo(float x, float y, float promien, int min) {`
	`  ellipse(x, y, promien, promien);`
	`  if(promien > min) {`
	`  rysujKolo(x + promien/2, y, promien/2, min);`
	`  rysujKolo(x - promien/2, y, promien/2, min);`
	`  rysujKolo(x, y + promien/2, promien/2, min);`
	`  rysujKolo(x, y - promien/2, promien/2, min);`
	`  }`
	`}`
	Przykładowy efekt działania funkcji

	
Zadanie samodzielne	
	Zmodyfikuj algorytm rysowania frakala, tak aby możliwe było wybieranie (parametr wywołania funkcji) wersji fraktala z 2, 4 lub 8 kołami w następnym poziomie.

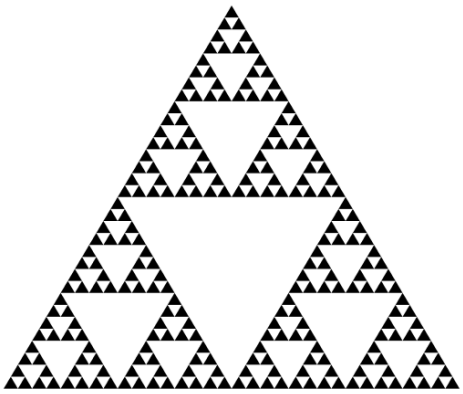
Ćwiczenie 2 – Rekurencja

Zadanie: Narysuj trójkąt Sierpińskiego	
Zbuduj funkcję rekurencyjną, rysującą trójkąt Sierpińskiego dowolnego poziomu	Trójkąt Sierpińskiego jest fraktalem rysowanym według zasady: Każdy trójkąt poprzedniego poziomu jest zastępowany przez grupę 3 trójkątów o wierzchołkach rozmieszczonych tak, że jeden z nich pokrywa się z wierzchołkiem trójkąta poprzedniego poziomu a pozostałe dwa umieszczane są w połowie boków trójkąta poprzedniego poziomu  Aby zbudować taki trójkąt potrzebne są więc współrzędne trójkąta 1-go poziomu oraz ilość poziomów. Algorytm rysujący będzie musiał wyliczać pozycje wierzchołków kolejnych trójkątów dzieląc na pół odcinki między wierzchołkami trójkąta poprzedniego poziomu. Zapis współrzędnych wierzchołków można uprościć stosując klasę Pvector pozwalającą na zapis współrzędnych wektora (2 lub 3 wymiarowych) `lewy = new PVector(x, y)` Funkcja realizująca rysowanie trójkątów Sierpińskiego ma 5 parametrów: Bieżący poziom rysowania trójkątów, maksymalny poziom rysowania trójkątów, 3 współrzędne bieżącego trójkąta Funkcja zawiera instrukcję warunkową, w przypadku gdy jesteśmy na ostatnim poziomie rysowany jest trójkąt. W przeciwnym przypadku obliczane są współrzędne środków odcinków bieżącego trójkąta i wywoływana jest rekurencyjnie funkcja narysowania 3 trójkątów

kolejnego poziomu.

```
void TrojkatS(int poziom, int maxPoziom, PVector lewy,
PVector gora, PVector prawy)
{

  poziom++;

  if(poziom >= maxPoziom) {
    triangle(lewy.x, lewy.y, gora.x, gora.y, prawy.x,
prawy.y);
  }
  else {
    PVector a = PVector.add(lewy,
PVector.div(PVector.sub(gora, lewy), 2));
    PVector b = PVector.add(prawy,
PVector.div(PVector.sub(gora, prawy), 2));
    PVector c = PVector.add(lewy,
PVector.div(PVector.sub(prawy, lewy), 2));

    TrojkatS(poziom, maxPoziom, a, gora, b);
    TrojkatS(poziom, maxPoziom, lewy, a, c);
    TrojkatS(poziom, maxPoziom, c, b, prawy);
  }

}
```

Pozostaje zdefiniowanie rozmiarów kna, kolorów linii, wypełnienia i tła oraz początkowe wywołanie funkcji.
Efekt końcowy:



Zadanie samodzielne

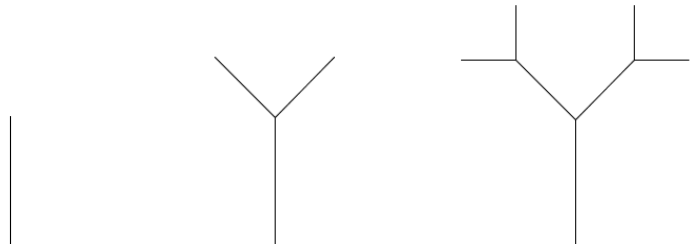
Zmodyfikuj algorytm rysowania trójkąta Sierpińskiego tak, aby rysowane były również trójkąty z pośrednich poziomów. Trójkąty każdego z poziomów powinny być wypełniane innym kolorem.

Ćwiczenie 3 – Rekurencja

Zadanie: Narysuj drzewo binarne

Zbuduj algorytm rysowania drzewa binarnego

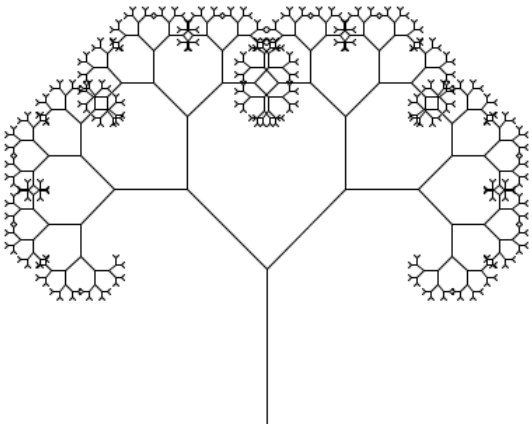
Drzewo binarne polega na ryswaniu linii (gałęzi) z każdej gałęzi poprzedniego poziomu, począwszy od pierwszej linii-korzenia. W drzewie binarnym z gałęzi „wyrastają” zawsze dwie gałęzie kolejnego poziomu



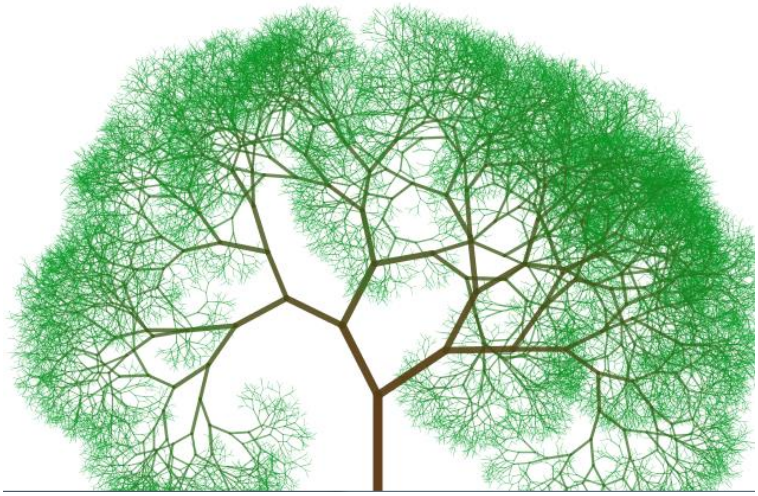
Kąt wychylenia kolejnych poziomów gałęzi (w powyższym przypadku 45 stopni) może być w zakresie 0-90 i niekoniecznie taki sam dla obu „konarów”.

Algorytm rysujący tym razem zamiast obliczania położenia kolejnych gałęzi wymaga zapisywania na stosie kolejnych przekształceń kanwy (obrót).

```
float kat = radians(45);
void setup() {
  size(640, 360);

}

void draw() {
  background(255);
  stroke(0);
  translate(width/2,height);
    //początkowy pien drzewa
    line(0,0,0,-120);
    translate(0,-120);
    konar(120);
}

void konar(float h) {
  // Zmniejszenie długości o 1/3
   h *= 0.66;
  if (h > 10) {
    //konar 1
    pushMatrix();
    rotate(kat);
    line(0, 0, 0, -h);
    translate(0, -h);
    konar(h);
    popMatrix();

    //konar 2
    pushMatrix();
    rotate(-kat);
    line(0, 0, 0, -h);
    translate(0, -h);
    konar(h);
    popMatrix();
  }
}
```



Zadanie samodzielne	
	Zmodyfikuj algorytm rysowania drzewa binarnego tak, aby gałęzie były rysowane z losowym odchyleniem od zdefiniowanej wartości ich nachylenia. Odchylenie powinno zawierać się w zakresie +-10% zdefiniowanej wartości. Kolejne poziomy powinny być rysowane ze zmianą koloru. Gałęzie w kolejnych poziomach powinny być rysowane cieńszą linią niż poziom wcześniejszy.



Wprowadź losową modyfikację długości gałęzi

Ćwiczenie 4 – Animacja

Zadanie: Animuj drzewo binarne	
Zbuduj algorytm animowania „rozkładania się" drzewa binarnego	Na poprzednim laboratorium sygnalizowaliśmy, że Processing może pracować w trybie aktywnym. Obecnie wykorzystamy tą możliwość do wykonania animacji „rozkładania się" drzewa binarnego. Przypominając: komendy zawarte w obszarze draw() { } wykonywane są

cyklicznie. Częstotliwość przerysowywania obrazu można regulować parametrem frameRate(n) umieszczonym w tym obszarze. Processing będzie się starał utrzymać zdefiniowaną ilość klatek na sekundę w miarę złożoności rysunku i możliwości sprzętowych.

Modyfikacja algorytmu z ćwiczenia 3 będzie polegała więc na wprowadzeniu zmiennej odpowiadającej za kąt rozwarcia gałęzi i zwiększanie jej co klatkę. Dodatkowo wprowadzamy ograniczenie rozwarcia gałęzi do 90 stopni

```
void draw() {
if (b <= 90) b++;
kat = radians(b);
…
…
}
```

Zadanie samodzielne

Zmodyfikuj algorytm tak aby drzewo rozkładało i składało się w pętli.

Rozpoczynając od rozłożonego drzewa (np. 45 stopni) zbuduj algorytm animacji „kiwania się na wietrze” czyli odchylania wszystkich gałęzi w tą samą stronę i powrotu. Spróbuj uwzględnić efekt przyspieszania przy rozpoczynaniu ruchu i zwalniania przy osiąganiu stanu końcowego.