Gra w chaos

P. S.

19 września 2020

1 Wstep

Fraktal pochodzi od łacińskiego słowa fractus, które oznacza złamany. Fraktal to obiekt samopodobny co oznacza, że jego cześci sa podobne do całości. Nie ma ścisłej definicji fraktali. Trudno go opisać za pomoca tradycyjnej geometrii. Przykładem fraktala jest trójkat Sierpińskiego. Jest to jeden z najprostszych fraktali. Aby otrzymać ten fraktal należy w trójkacie równobocznym połaczyć środki boków, nastepnie usunać środkowy z powstałych trójkatów i cały proces powtarzać dla mniejszych trójkatów. Algorytm nazywany gra w chaos pozwala generować fraktale. Stworzenie trójkata Sierpińskiego tym algorytmem składa sie z kilku kroków iterowanych określona liczbe razy. Na poczatku określamy wierzchołki trójkata, po czym wybieramy kolejny punkt, nastepnie stawiamy punkt w połowie odległości pomiedzy wybranym punktem, a jednym z losowo wybranym wierzchołkiem, po czym wybranym punktem staje sie punkt ostatnio narysowany i znowu stawiamy punkt w połowie odległości pomiedzy nowym punktem, a jednym z losowo wybranym wierzchołkiem. Cały proces powtarzamy określona liczbe razy.

2 Cel pracy

Celem pracy była implementacja algorytmu "Gra w chaos" w celu generowania fraktali na przykładzie trójkata Sierpińskiego.

3 Implementacja

Algorytm został zaimplementowany w C++ oraz w Pythonie. W projekcie została użyta biblioteka Qt5/PyQt5.

```
QPainter painter(this);
painter.setPen(QPen(Qt::blue, 1, Qt::DashDotLine, Qt::RoundCap));
const QPoint points[3] = {{0,400}, {200,0}, {400,400}};
QPoint point(200, 250);
for (int i = 0; i<100000; ++i) {
    std::random_device rd{};
    std::mt19937 gen{rd()};
    std::random_device rd2{};
    std::mt19937 gen2{rd2()};
    std::uniform_int_distribution<> distrib(0, 2);
    int x = distrib(gen);

    painter.drawPoint(point);
    point.setX((point.x()+points[x].x())/2);
    point.setY((point.y()+points[x].y())/2);

}
```

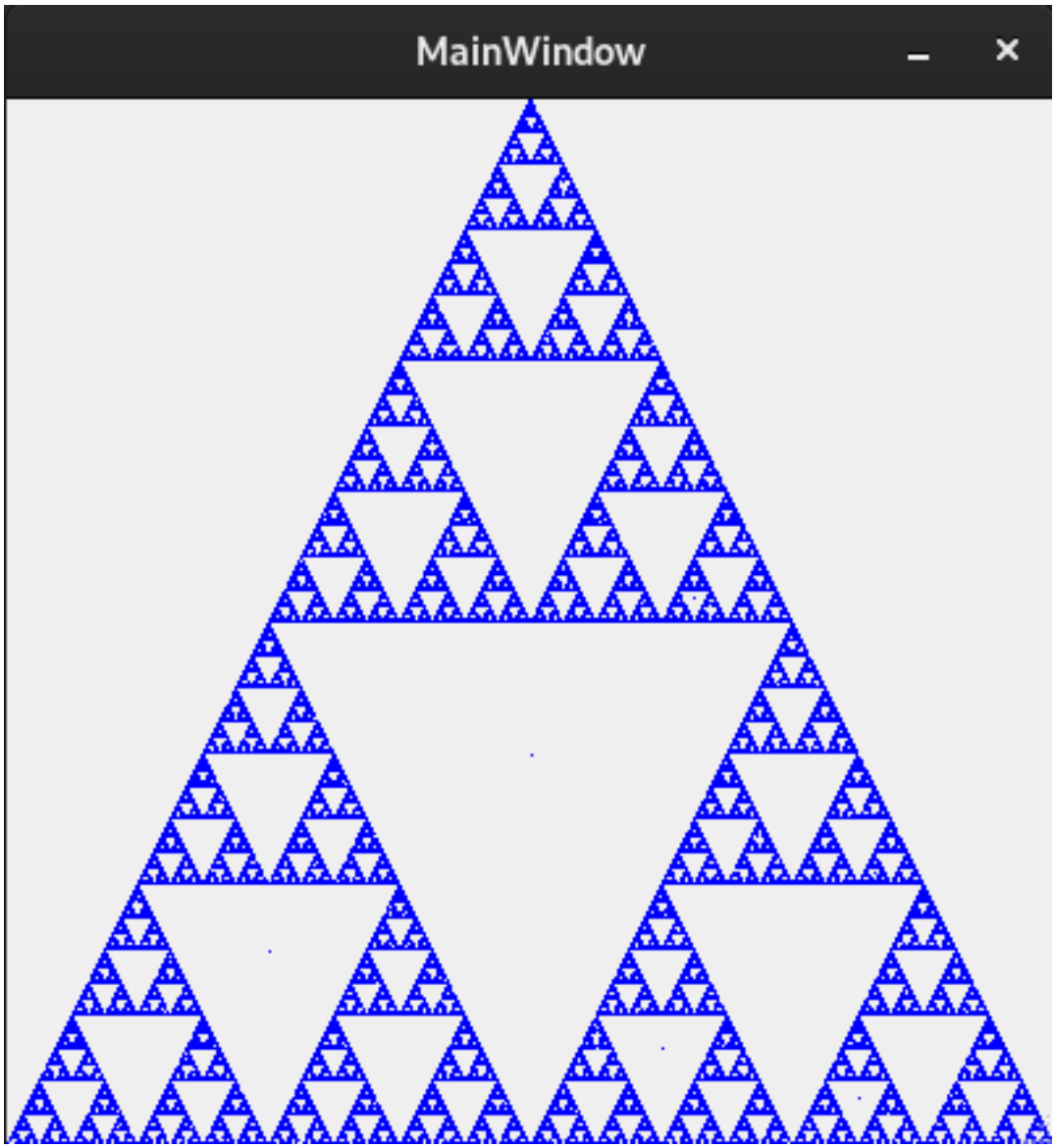
Rys 1. Implementacja w C++

```
def draw(self):
    from random import randint
    painter = QtGui.QPainter(self.label.pixmap())
    pen = QtGui.QPen()
    pen.setWidth(3)
    pen.setColor(QtGui.QColor('red'))
    painter.setPen(pen)
    p = QPoint(0,400)
    points = [QPoint(0,400), QPoint(200,0), QPoint(400,400)]
    for n in range(10000):
        painter.drawPoint(p)
        random = randint(0, 2)
        p.setX((p.x()+points[random].x())/2)
        p.setY((p.y()+points[random].y())/2)
    painter.end()
```

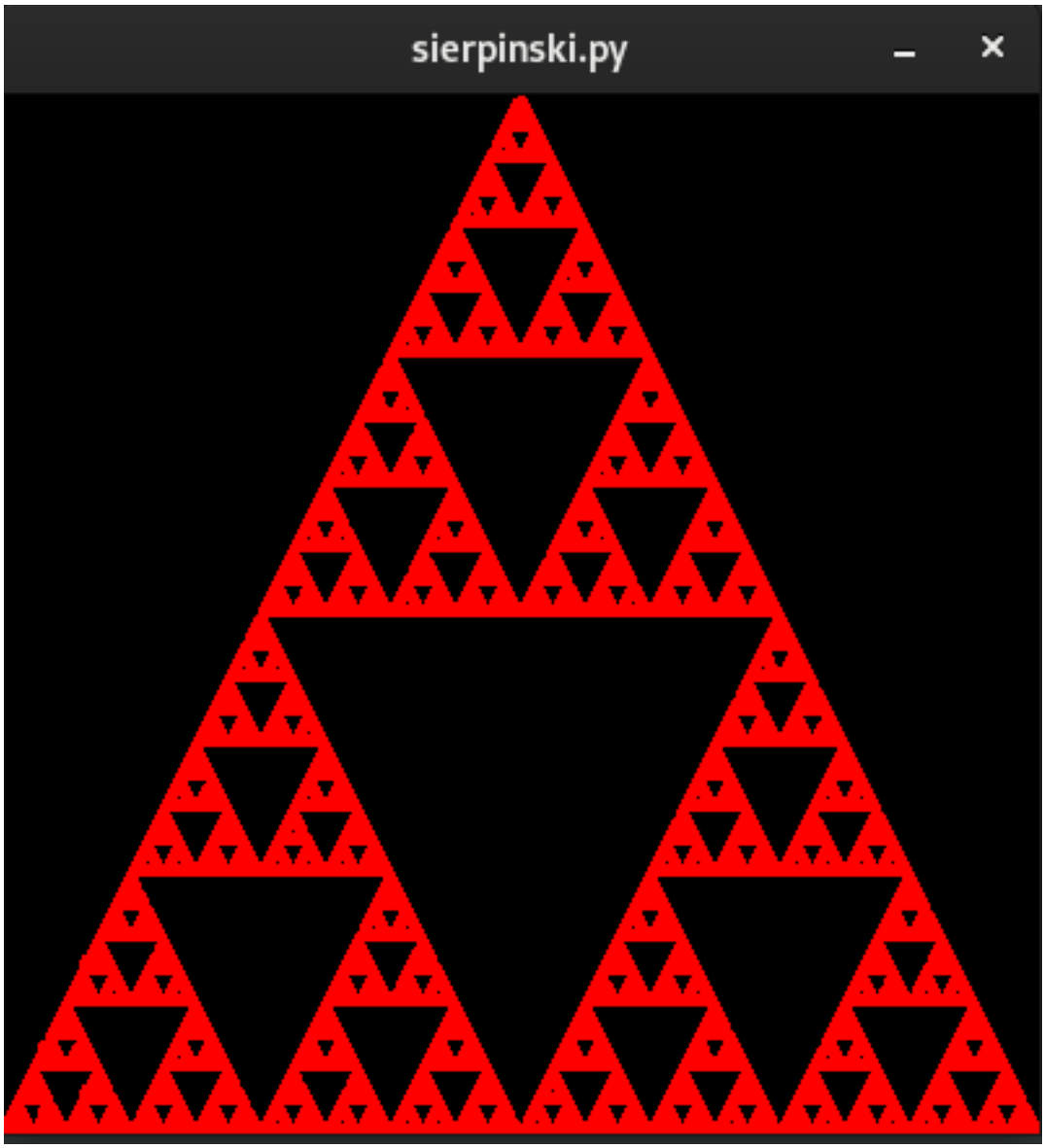
Rys 2. Implenetacja w Python

4 Podsumowanie

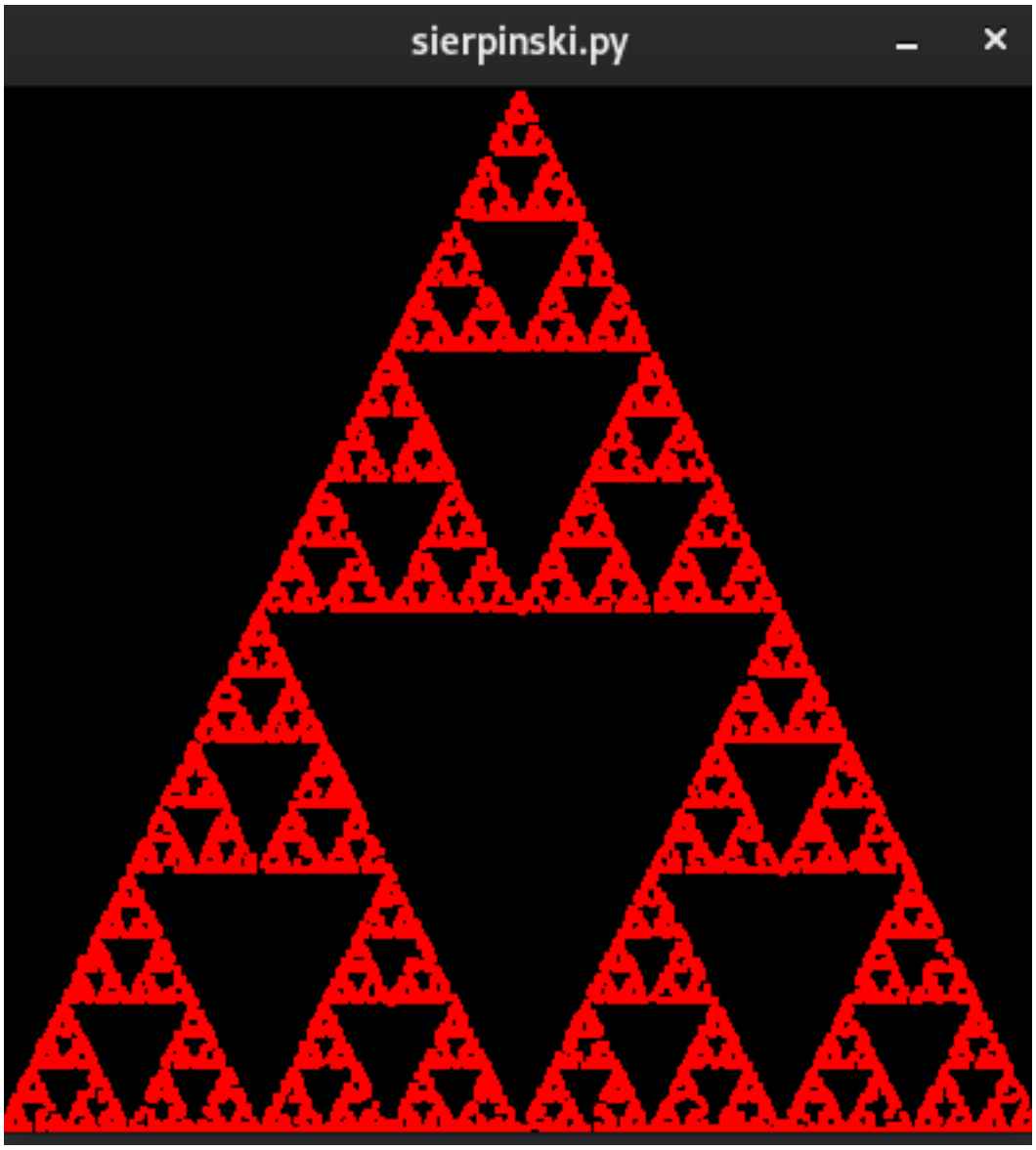
Udało sie zaimplementować algorytm, aby wynik był zadawalajacy, algorytm potrzebował około 100000 iteracji.



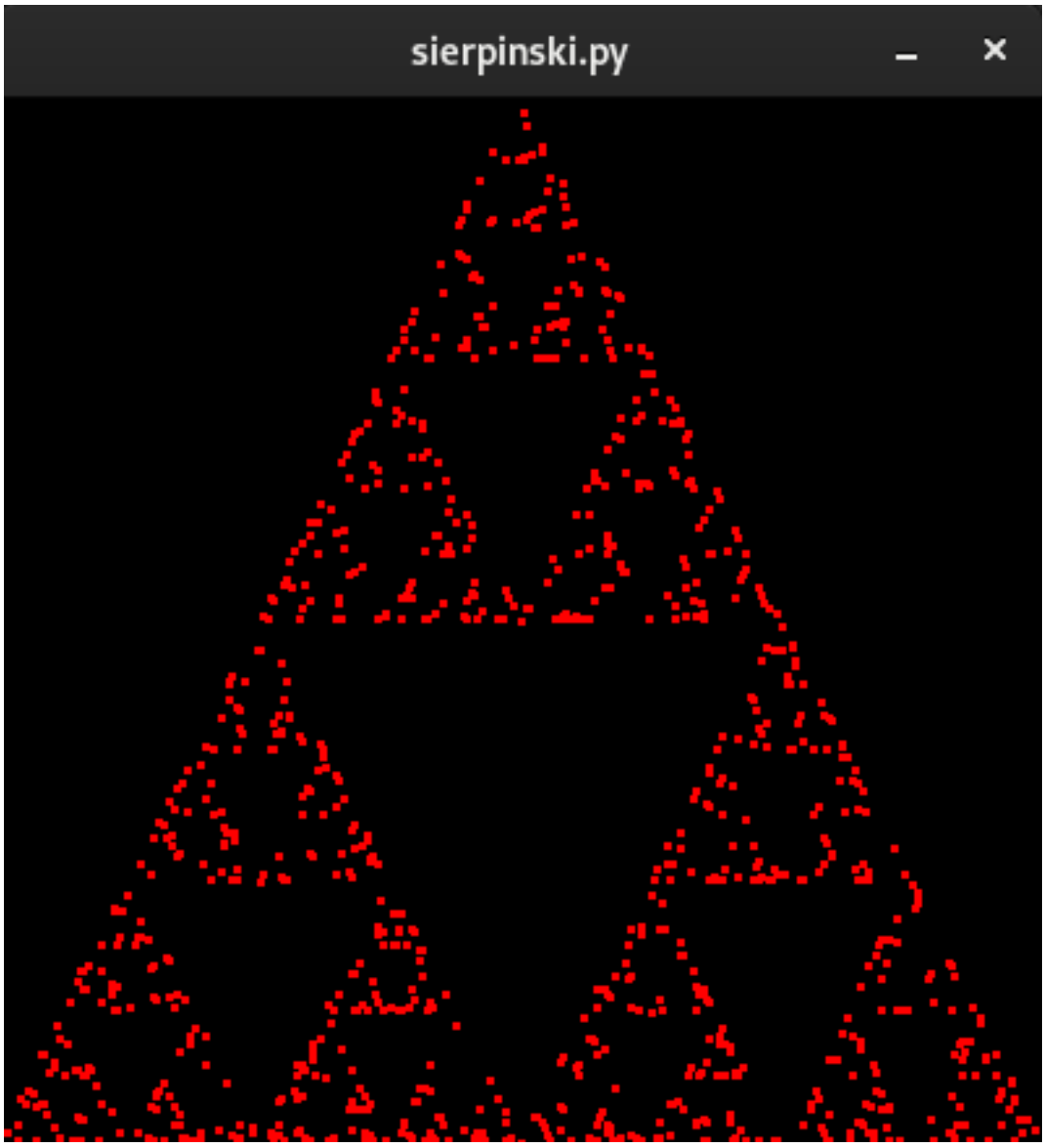
Rys. 3 Działanie programu napisanego w C++



Rys. 4 Działanie programu napisanego w Python



Rys. 5 Działanie programu napisanego w Python przy 10000



Rys. 6 Działanie programu napisanego w Python przy 1000

5 Bibliografia

[1] https://pl.wikipedia.org/wiki/Tr%C3%B3jk%C4%85t_Sierpi%C5%84skiego

[2] https://pl.wikipedia.org/wiki/Gra_w_chaos

[3] https://en.wikipedia.org/wiki/Chaos_game

[4] https://doc.qt.io/qt-5/reference-overview.html

[5] https://doc.qt.io/qtforpython/