Object-Oriented Programming

Exercise series 7

Introduction

In this series, you will expand your solutions for series 5. Indeed, you will design and implement additional classes in order to be able to apply transformations on `BrokenLine` objects. In this context, a transformation consists in taking each segment of a `BrokenLine` object and transforming it into a new subsequence of segments following a given set of rules.

In particular, you will implement the 3 following transformations :

— *Koch curve* : the initial segment is split in three segments of the same length. Then, an equilateral triangle is built on the middle segment. Finally, the middle segment is removed.

— *Square variant* : the initial segment is transformed in the same way as for Koch curve, but a square is built on the middle segment instead of an equilateral triangle before removing this middle segment.

— *Minkowski curve* : the initial segment is split in four segments of the same length. Then, a square is built on the second segment and a second square is built under the third segment. Finally, the second and third segments are removed, giving the broken line the appearance of a pulse.

All three transformations can be repeated several times on a same `BrokenLine` object in order to approximate fractals. These transformations are shown in Fig. 1.

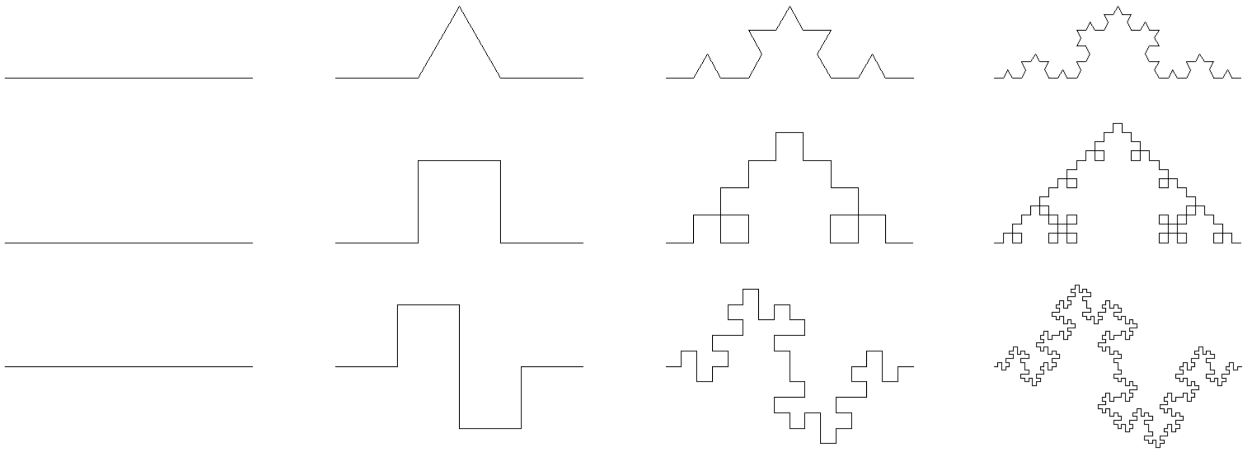

FIGURE 1 – Transformations of a segment. The initial `BrokenLine` is transformed several times. From top to bottom : *Koch curve*, *square variant* and *Minkowski curve.*

Exercise 1

Before doing any implementation, take a closer look at the transformations shown on p. 1. Can you define a set of variables with which can you model all these transformations ? If yes, what are those variables ?

Tip : what about the orientation of the segments with respect to the horizontal axis ?

Exercise 2

Create a `Transformation` class which complies with your answer to exercise 1. This new class should provide a `transform()` method which receives a segment as a parameter and returns a reference to the last (sub-)segment of the transformed segment.

Then, adapt the classes you used to solve exercises from series 5 such that you can apply any transformation on them.

Remark : is the `abstract` keyword relevant here ? Why (not) ?

Exercise 3

Extend your `Transformation` class in order to implement each of the transformations described on p. 1. Then, create a test program to test your solution. For instance, you can instantiate 3 `BrokenLine` objects and transform them several times, each with a different transformation, in order to obtain a display similar to the rightmost figures on p. 1.