

Języki skryptowe

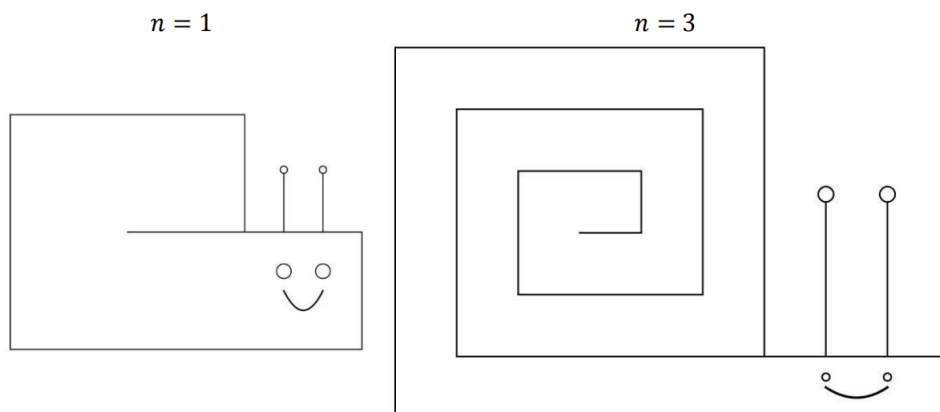
Tomasz Tomaszek, grupa 1/2

2 stycznia 2024

Część I

Opis programu

Należało napisać program, który dla zadanej liczby naturalnej n rysować będzie ślimaka. Liczba pętli muszli ślimaka wyznaczana jest przez wartość n , długość głowy ślimaka (część na prawo od muszli) jest równa połowie długości muszli (mierząc tę muszlę na górze). Czułki ślimaka mają wysokość równą połowie muszli licząc od górnej części głowy ślimaka. Oczy na czułkach są okręgami o promieniu proporcjonalnym do n (proporcjonalność tę można było dobrać według własnego uznania). Oczy na głowie są okręgami, których środki wyznaczone są przez położenie czulek (współrzędna n) oraz $\frac{1}{3}$ wysokości głowy i mają promień stały (niezależny od n). Uśmiech ślimaka jest fragmentem paraboli, której wierzchołek leży po środku oczu (współrzędna n) i w $\frac{1}{3}$ wysokości głowy licząc od dołu tej głowy (współrzędna n), a współczynnik n tej paraboli dobrany jest tak, że uśmiech kończy się w połowie głowy (współrzędna n) natomiast ślimak uśmiecha się „od oka do oka” (współrzędna n). Na rysunkach niezachowane są proporcje, ogólnie, dla $n = 1$ muszla ślimaka (mierząc ją na górze) ma długość 2 (rysunek lewy), a dla $n = 3$ muszla ślimaka (mierząc ją na górze) ma długość 6 (rysunek prawy).



Rysunek 1: Przykładowe ślimaki

Część II

Opis działania

Program wykorzystuje bibliotekę PIL, która umożliwia tworzenie obrazu. W skład struktury programu wchodzi klasa Snail oraz metody rysujące poszczególne części ślimaka. Program po podaniu przez użytkownika liczby całkowitej rysuje w oknie o rozdzielczości 1920x1080 ślimaka o wymaganych parametrach. Tło jest białe a kolor rysunku czarny. Metody rysujące tworzą ślimaka z linii, okręgów oraz parabol (parabolą jest w zasadzie zbiór punktów należących do wykresu funkcji kwadratowej). Jako długość początkowego segmentu obrałem 50px. Poniżej przedstawiam algorytmy wykorzystane przy rysowaniu ślimaka.

drawShell()

Metoda rysująca muszelkę ślimaka. Współrzędne początkowe obliczane są ze wzorów:

$$x = 50 + n \cdot 50, y = 50 + n \cdot 50$$

Współrzędne "środk" muszli zależą od liczby n .

Data: Współrzędne początkowe i końcowe: $prev_x, prev_y, next_x, next_y$

Liczba całkowita n

Długość segmentu s

for n **do**

$next_x + = s$

 rysuj linię

$prev_x = next_x$

$next_y - = s$

 rysuj linię

$s + = 50px$

$next_x - = s$

$prev_y = next_y$

 rysuj linię

$prev_x = next_x$

$next_y + = s$

 rysuj linię

$prev_y = next_y$ $s + = 50px$

end

$next_x = s - 50px$

rysuj linię

drawHead()

Metoda rysująca głowę ślimaka. Współrzędne początkowe są współrzędnymi końcowymi z metody drawShell().

Data: Współrzędne początkowe i końcowe: $prev_x, prev_y, next_x, next_y$

Długość segmentu s

rysuj linię

$$next_y - = 50$$

rysuj linię

$$prev_y = next_y$$

$$next_x - = \frac{s}{2}$$

rysuj linię

drawEyes()

Metoda rysująca oczy ślimaka. Współrzędne środka są obliczane ze wzorów:

$$center_x = prev_x - \frac{s}{6}, center_y = prev_y + \frac{50}{3}$$

Data: Współrzędne środka okręgu: $center_x, center_y$

Długość segmentu s

rysuj okrąg

$$center_x - = \frac{s}{6}$$

rysuj okrąg

drawAnthenas()

Metoda rysująca czułki ślimaka. Współrzędne początkowe są obliczane ze wzorów:

$$x = prev_x - \frac{s}{6}, y = prev_y$$

Promień czulek jest obliczany ze wzoru:

$$(2 + n)px$$

Data: Współrzędne początkowe, końcowe i promień czulek:

$prev_x, prev_y, next_x, next_y, r$

Długość segmentu s

$$next_y = prev_y - \frac{s-100}{2}$$

rysuj linię

$$next_y - = r$$

rysuj okrąg

$$prev_x - = \frac{s}{6}$$

$$next_y + = r$$

$$next_x = prev_x$$

rysuj linię

$$next_y - = r$$

rysuj okrąg

drawSmile()

Metoda rysująca uśmiech ślimaka. Parabola jest w rzeczywistości zbiorem punktów należących do funkcji $y = ax^2$. Współrzędne wierzchołka paraboli obliczane są ze wzoru:

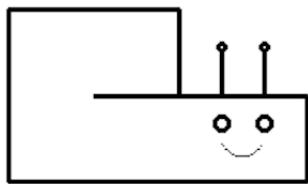
$$x = prev_x - \frac{s}{4}$$

$$y = prev_y + 35$$

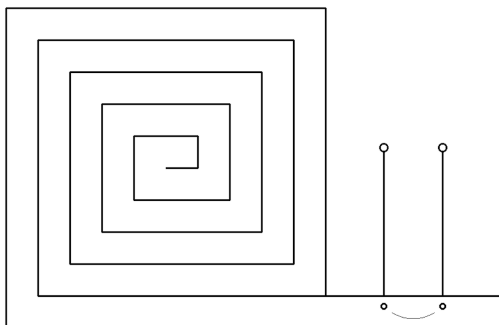
Współczynnik a został dobrany rozwiązując poniższy układ równań. Wartości 0.05 oraz 0.001 były odpowiednie dla $n = 1$ oraz $n = 10$. Dobranie odpowiedniego wzoru funkcji było zdecydowanie najtrudniejszą częścią projektu.

$$\begin{cases} 0.05 = ab \\ 0.001 = ab^{10} \end{cases}$$

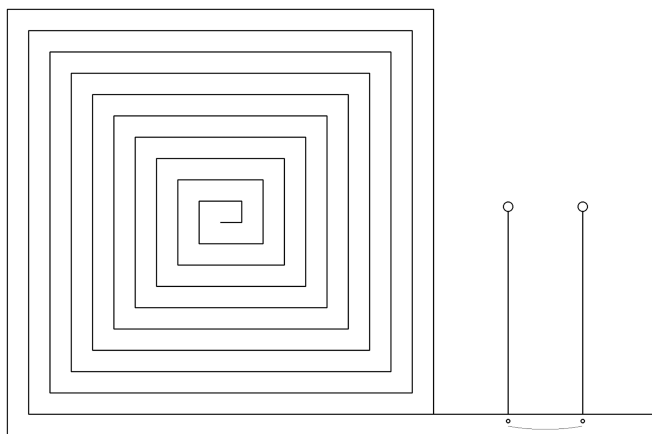
Program w zakresie od $-\frac{s}{12}$ do $\frac{s}{12}$ wyznacza punkty, które tworzą parabolę ax^2 . Współczynnik $a = 0.077222605 \cdot 0.647478803^n$. Dla każdej wielkości ślimaka współczynnik a będzie przybierał inną wartość.



Rysunek 2: Dla $n = 1$



Rysunek 3: Dla $n = 5$



Rysunek 4: Dla $n = 10$

Pełen kod aplikacji

```
1 # Biblioteka PIL umożliwia tworzenie rysunków
2
3 from PIL import Image, ImageDraw
4
5 # Klasa snail
6
7 class Snail:
8
9     # Konstruktor obiektu
10
11     def __init__(self, n):
12         self.n = n # Atrybut n
13         self.segment_length = 50 # Jednostka długości n
14         self.eye_radius = 5
15         self.antennas_radius = 2 + self.n
16         self.image = Image.new("RGB", (1920, 1080), "white") #
            # Tworzenie rysunku
17         self.draw = ImageDraw.Draw(self.image) # Metoda rysująca
18         self.position_x = (50 + self.n * 50) # Aktualna pozycja "
            # rysowania"
19         self.position_y = (50 + self.n * 50)
20
21     # Metoda rysująca muszle
22
23     def drawShell(self):
24         prev_x = self.position_x # Zmienne pozycji początkowej linii
25         prev_y = self.position_y
26         next_x = prev_x # Zmienne pozycji końcowej linii
27         next_y = prev_y
28         for _ in range(self.n): # Pętla rysująca muszle
29             next_x += self.segment_length # Do końcowej współrzędnej x
                # dodajemy długość segmentu
30             self.draw.line([prev_x, prev_y, next_x, prev_y], fill = "
                black", width = 3) # Rysowanie linii
31             prev_x = next_x # Aktualizacja współrzędnej x
32             next_y -= self.segment_length # Aktualizacja współrzędnej y
33             self.draw.line([prev_x, prev_y, next_x, next_y], fill = "
                black", width = 3) # Rysowanie linii
34             self.segment_length += 50
35             next_x -= self.segment_length
36             prev_y = next_y
37             self.draw.line([prev_x, prev_y, next_x, next_y], fill = "
                black", width = 3)
38             prev_x = next_x
39             next_y += self.segment_length
40             self.draw.line([prev_x, prev_y, next_x, next_y], fill = "
                black", width = 3)
41             prev_y = next_y
42             self.segment_length += 50
43         next_x += self.segment_length - 50
44         self.draw.line([prev_x, prev_y, next_x, next_y], fill = "black",
            width = 3)
```

```

45     prev_x = next_x
46     self.position_x = prev_x
47     self.position_y = prev_y
48
49     # Metoda rysujaca glowe
50
51     def drawHead(self):
52         prev_x = self.position_x
53         prev_y = self.position_y
54         next_x = prev_x + self.segment_length / 2
55         next_y = prev_y
56         self.draw.line([prev_x, prev_y, next_x, next_y], fill = "black",
57                         width = 3)
58         prev_x = next_x
59         next_y -= 50
60         self.draw.line([prev_x, prev_y, next_x, next_y], fill = "black",
61                         width = 3)
62         prev_y = next_y
63         next_x -= self.segment_length / 2
64         self.draw.line([prev_x, prev_y, next_x, next_y], fill = "black",
65                         width = 3)
66         self.position_x = prev_x
67         self.position_y = prev_y
68
69     # Metoda rysujaca oczy
70
71     def drawEyes(self):
72         center_x = self.position_x - self.segment_length / 6
73         center_y = self.position_y + 50 / 3
74         self.draw.ellipse((center_x - self.eye_radius, center_y - self.
75                             eye_radius, center_x + self.eye_radius, center_y + self.
76                             eye_radius), outline="black", width=3)
77         center_x -= self.segment_length / 6
78         self.draw.ellipse((center_x - self.eye_radius, center_y - self.
79                             eye_radius, center_x + self.eye_radius, center_y + self.
80                             eye_radius), outline="black", width=3)
81
82     def drawAntennas(self):
83         prev_x = self.position_x - self.segment_length / 6
84         prev_y = self.position_y
85         next_x = prev_x
86         next_y = prev_y - ((self.segment_length - 100) / 2)
87         self.draw.line([prev_x, prev_y, next_x, next_y], fill = "black",
88                         width = 3)
89         next_y -= self.antennas_radius
90         self.draw.ellipse((next_x - self.antennas_radius, next_y - self.
91                             antennas_radius, next_x + self.antennas_radius, next_y + self.
92                             antennas_radius), outline="black", width=3)
93         prev_x -= self.segment_length / 6
94         next_y += self.antennas_radius
95         next_x = prev_x
96         self.draw.line([prev_x, prev_y, next_x, next_y], fill = "black",
97                         width = 3)
98         next_y -= self.antennas_radius
99         self.draw.ellipse((next_x - self.antennas_radius, next_y - self.

```

```

        antenas_radius, next_x + self.antenas_radius, next_y + self
        .antenas_radius), outline="black", width=3)
89
90     def drawSmile(self):
91         center_x, center_y = self.position_x - self.segment_length / 4,
            self.position_y + 35
92         for x in range(-int((self.segment_length / 12)), int(self.
            segment_length / 12)):
93             a = 0.077222605 * (0.647478803 ** self.n) # 1 - 0.05 10 -
                0.001 2 - 0.02 3 - 0.01 4 - 0.006 5 - 0.004 6 - 0.003
94             y = int(a * x ** 2)
95             if y < 10:
96                 self.draw.point((center_x + x, center_y - y), fill = "
                    black")
97
98     def drawSnail(self):
99         self.drawShell()
100        self.drawHead()
101        self.drawEyes()
102        self.drawSmile()
103        self.drawAntenas()
104        self.image.show()
105
106 n = int(input("Podaj liczbe n: "))
107 snail = Snail(n)
108 snail.drawSnail()

```
