

Języki skryptowe

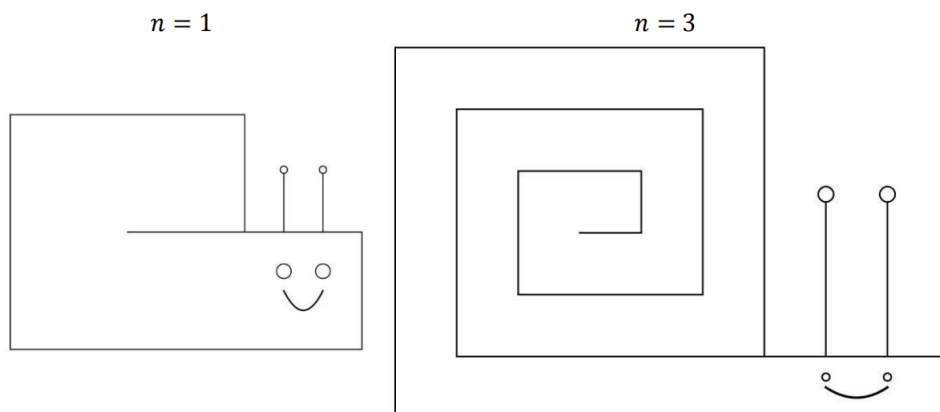
Tomasz Tomaszek, grupa 1/2

24 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:

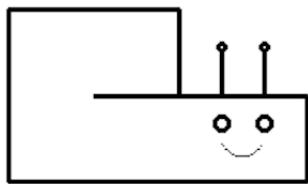
$$x = prev_x - \frac{s}{4}$$

$$y = prev_y + 35$$

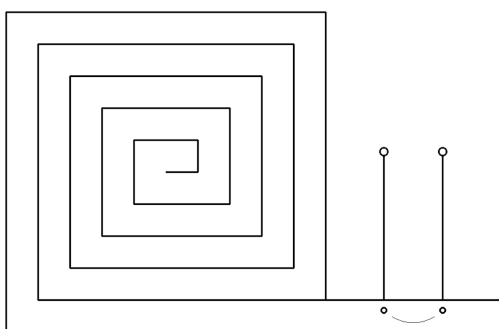
Współczynnik a został dobrany rozwiązując poniższy układ równań. Wartości 0.05 oraz 0.001 były odpowiednie dla $n = 1$ oraz $n = 10$. Dobranie odpowiedniego wzoru funkcji było zdecydowanie najtrudniejszą częścią projektu.

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

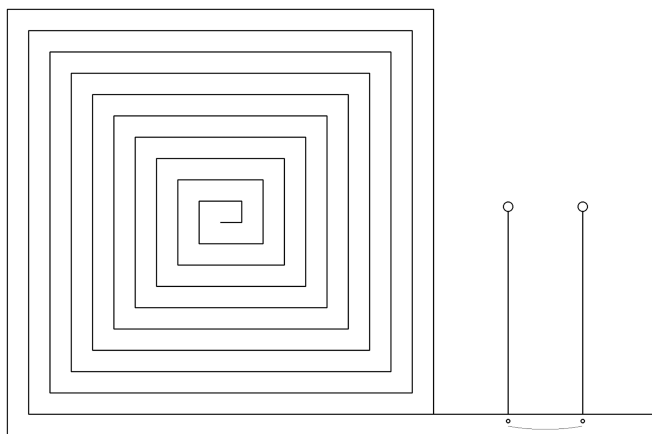
Program w zakresie od $-\frac{s}{12}$ do $\frac{s}{12}$ wyznacza punkty, które tworzą parabolę ax^2 . Współczynnik $a = 0.077222605 \cdot 0.647478803^n$. Dla każdej wielkości ślimaka współczynnik a będzie przybierał inną wartość.



Rysunek 2: Dla $n = 1$



Rysunek 3: Dla $n = 5$



Rysunek 4: Dla $n = 10$

Pełen kod aplikacji

0.1 snail.py

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

```

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

```

```

        width = 3)
88     next_y -= self.anthenas_radius
89     self.draw.ellipse((next_x - self.anthenas_radius, next_y - self.
        anthenas_radius, next_x + self.anthenas_radius, next_y + self
        .anthenas_radius), outline="black", width=3)
90
91     def drawSmile(self):
92         center_x, center_y = self.position_x - self.segment_length / 4,
            self.position_y + 35
93         for x in range(-int((self.segment_length / 12)), int(self.
            segment_length / 12)):
94             a = 0.077222605 * (0.647478803 ** self.n)
95             y = int(a * x ** 2)
96             if y < 10:
97                 self.draw.point((center_x + x, center_y - y), fill = "
                    black")
98
99     def drawSnail(self):
100         self.drawShell()
101         self.drawHead()
102         self.drawEyes()
103         self.drawSmile()
104         self.drawAnthenas()
105         timestamp = datetime.datetime.now().strftime("%Y%d%m%H%M%S")
106         image_path = f"output/image_{timestamp}.png"
107         self.image.save(image_path)
108         image_path = f"backup/image_{timestamp}.png"
109         self.image.save(image_path)
110         image_path = f"image_{timestamp}.png"
111         try:
112             with open("image_names.txt", "a") as file:
113                 file.write(image_path + '\n')
114         except FileNotFoundError:
115             print("error")
116         except ValueError:
117             print("error")
118         n = None
119
120     try:
121         with open("input.txt", "r") as file:
122             lines = file.readlines()
123             if lines:
124                 n = int(lines[-1])
125             else:
126                 print("Error: File is empty")
127                 n = None
128         except FileNotFoundError:
129             print("Error: FileNotFoundError")
130         except ValueError:
131             print("Error: ValueError")
132         n = None
133
134
135     if n is not None:
136         snaily = Snail(n)

```


0.2 snail.bat

```
1      @echo off
2 :menu
3 cls
4 echo #####
5 echo          MENU
6 echo #####
7 echo 1. Wygeneruj rysunek
8 echo 2. Generuj raport
9 echo 3. Zamknij program
10 echo #####
11
12 set /p choice="wybierz: "
13
14 if "%choice%"=="1" goto startup
15 if "%choice%"=="2" goto generateReport
16 if "%choice%"=="3" goto close
17
18 echo error
19 timeout /nobreak /t 1 >nul
20 goto menu
21
22 :startup
23 echo Generuje rysunek
24 py ".\snail.py"
25 pause
26 goto menu
27
28 :generateReport
29
30
31 :close
32 timeout /nobreak /t 1 >nul
33 exit /b
```