Dokumentace k úlohám z předmětu Úvod do programování

Autor:
Vít Suchomel, 3. ročník, SGGI

1. Zadání

Vizualizace n-cípé hvězdy. S využitím želví grafiky vykreslete n-cípou hvězdu. Vstupní parametry představují počet cípů n, poloměr *r1* opsané kružnice *k1* a poloměr *r2* kružnice *k2* procházející místem napojení sousedních cípů.

2. Popis a rozbor problému

Úloha spadá do oblasti počítačové grafiky, konkrétně analytické geometrie v rovině. Cílem je transformovat geometrickou definici útvaru, danou počtem cípů a dvěma poloměry, na sérii vykreslovacích příkazů. Matematicky lze n-cípou hvězdu definovat jako uzavřený lomený polygon, který má *2n* vrcholů. Tyto vrcholy se pravidelně střídají mezi dvěma soustřednými kružnicemi, přičemž sudé vrcholy (hroty hvězdy) leží na vnější kružnici a liché vrcholy (body napojení) na kružnici vnitřní.

Pro určení přesné polohy vrcholů je nejvhodnější využít polární souřadný systém, kde je každý bod definován vzdáleností od středu a úhlem. Celkový úhel 360° je nutné rozdělit na *2n* pravidelných segmentů. Tyto polární souřadnice je následně třeba převést na kartézské souřadnice *[x, y]*, se kterými pracuje grafická knihovna.

3. Existující algoritmy

Při řešení úloh v želví grafice (Turtle graphics) se v zásadě uplatňují dva odlišné přístupy k vykreslování geometrických útvarů.

Prvním je relativní vykreslování, které je pro želví grafiku nativní. Želva je ovládána příkazy pro pohyb vpřed o určitou délku a otočení o určitý úhel. Tento způsob je intuitivní pro pravidelné mnohoúhelníky, avšak pro hvězdu definovanou dvěma poloměry je matematicky nevýhodný. Před samotným kreslením by bylo nutné složitě dopočítat délku strany a vnitřní úhel lomené čáry pomocí kosinové věty, což zbytečně zvyšuje výpočetní náročnost a riziko zaokrouhlovacích chyb.

Druhým přístupem je absolutní vykreslování neboli analytický přístup. V tomto případě se pozice každého vrcholu vypočítá předem pomocí goniometrických funkcí na základě úhlu a vzdálenosti od středu. Želva je poté instruována k přesunu na konkrétní souřadnice v ploše. Tento přístup byl zvolen pro řešení této úlohy, protože vstupní parametry (poloměry) přímo odpovídají proměnným v rovnici pro polární souřadnice, což činí řešení přímým a přesným.

4. Popis zvoleného algoritmu

Zvolený algoritmus je založen na iterativním výpočtu souřadnic bodů po obvodu kružnice. Proces začíná validací vstupních dat. Algoritmus ověřuje nejen geometrickou smysluplnost (vnější poloměr musí být větší než vnitřní), ale také kontroluje, zda zadané hodnoty nepřekračují stanovené provozní limity. Tím je zajištěno, že se hvězda vejde na obrazovku a počet cípů nezpůsobí neúměrnou časovou zátěž vykreslování. Následně program vstupuje do cyklu, který se opakuje *2n + 1* krát, přičemž přídavná iterace slouží k uzavření obrazce spojením posledního bodu s prvním.

V každém kroku cyklu se nejprve určí aktuální poloměr. Algoritmus využívá operaci modulo pro zjištění parity indexu bodu. Pro sudé indexy je použit vnější poloměr (hrot hvězdy), zatímco pro liché indexy poloměr vnitřní (úžlabí). Následně se vypočítá úhel bodu a provede se transformace na kartézské souřadnice pomocí funkcí *sinus* a *cosinus*. V poslední fázi kroku se grafické pero přesune na vypočtené souřadnice. Algoritmus rozlišuje první bod, kdy se pero pouze přesouvá bez kreslení, a ostatní body, kdy dochází k vykreslení úsečky.

5. Struktura programu

Program je implementován v jazyce Python a využívá objektově orientované programování (OOP). Pro zajištění požadované funkcionality využívá dvě standardní knihovny: modul *turtle* pro ovládání grafického kurzoru (želvy) a modul *math*, který poskytuje nezbytné goniometrické funkce a konstantu π pro výpočty souřadnic.

Jádrem aplikace je třída *Hvezda*, která zapouzdřuje veškerá data i chování související s vykreslovaným útvarem. Při inicializaci instance (metoda *__init__*) jsou do vnitřních atributů objektu uloženy parametry zadané uživatelem: počet cípů (*self.n*), poloměr opsané kružnice (*self.r_vnejsi*) a poloměr vnitřního napojení (*self.r_vnitrni*). Tím je zajištěno, že každá instance hvězdy si nese svou vlastní definici a je nezávislá na ostatních.

Hlavní výkonnou částí třídy je metoda *nakresli*. Ta nejprve provádí validaci dat, aby se předešlo logickým chybám ještě před spuštěním grafického okna. Pokud jsou parametry v pořádku, metoda spouští objekt *turtle.Turtle*, nastaví mu maximální rychlost vykreslování a tloušťku pera. Následně se spouští hlavní iterační cyklus, který probíhá v rozsahu *2n + 1* opakování. Přídavek jedné iterace je klíčový pro uzavření polygonu – zajišťuje, že se vykreslování vrátí přesně do výchozího bodu.

Uvnitř tohoto cyklu dochází k přepočtu polárních souřadnic na kartézské. Program v každém kroku volí poloměr na základě parity iterační proměnné (využití operátoru modulo %). Pro sudé kroky se používá vnější poloměr (hrot), pro liché vnitřní (úžlabí). Souřadnice *[x, y]* jsou následně vypočteny pomocí funkcí *math.cos* a *math.sin*. Metoda také inteligentně řídí stav pera – pro první bod je pero zvednuté (*penup*), aby nedošlo k

vykreslení čáry ze středu souřadnic, pro všechny následující body je pero položeno (*pendown*), čímž vzniká obrys hvězdy.

Řídící část programu, umístěná mimo třídu, zajišťuje interakci s uživatelem. Celý blok načítání vstupů je uzavřen v konstrukci *try-except*. Toto řešení robustně odchytává chyby při konverzi dat (např. pokud uživatel zadá text místo čísla) a zabraňuje pádu programu s výpisem *traceback*. V případě validního vstupu je vytvořena instance třídy *Hvezda* a je zavolána její vykreslovací metoda. Návratová hodnota metody (*True/False*) pak slouží k informování uživatele o výsledku operace.

6. Problematické situace a ošetření

Během běhu programu mohou nastat dvě kategorie chyb, které je nutné ošetřit. První kategorií jsou chyby vstupních dat, kdy uživatel zadá textový řetězec namísto čísla. Taková situace by standardně vedla k pádu programu s výjimkou *ValueError*. V kódu je proto vstupní sekce uzavřena do bloku *try-except*, který v případě neplatného vstupu vypíše srozumitelné varování a program bezpečně ukončí.

Druhou kategorií jsou logické či geometrické chyby. Uživatel může zadat číselné hodnoty, které však netvoří validní hvězdu. Jedná se například o situaci, kdy je počet cípů menší než tři, nebo kdy je poloměr vnitřní kružnice větší než poloměr kružnice opsané. Tyto stavy jsou detekovány podmínkami na začátku metody *nakresli*. Pokud metoda zjistí logický rozpor, vypíše specifickou chybovou hlášku a vrátí *False*, čímž zamezí vykreslení nesmyslného útvaru.

Třetí kategorií, je překročení maximálních povolených hodnot. Aby nedošlo k "zamrznutí" aplikace při výpočtu tisíců úseček nebo k vykreslování mimo viditelnou oblast okna, byly zavedeny pevné limity (např. maximální poloměr 400 a maximálně 50 cípů).

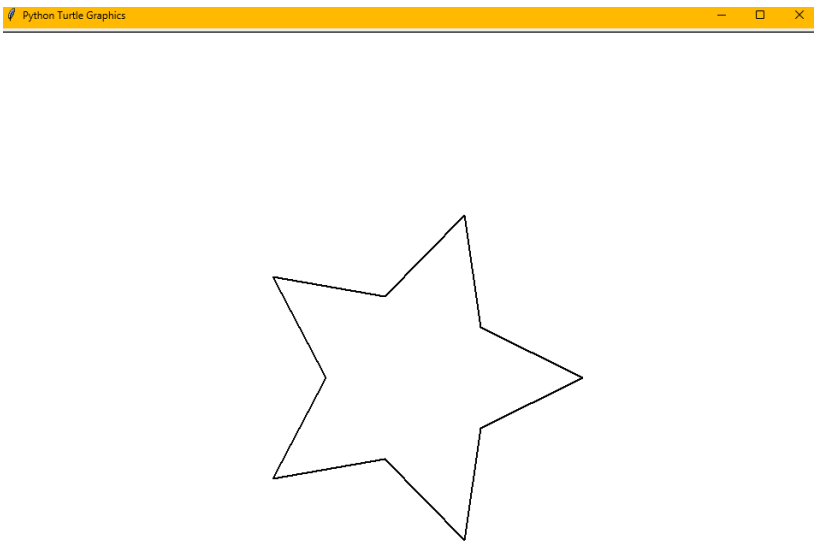
7. Vstupní a výstupní data

Interakce s programem probíhá kombinovanou formou. Vstupní data zadává uživatel prostřednictvím textové konzole. Program postupně vyžaduje tři hodnoty: počet cípů (celé číslo) a dva poloměry (reálná čísla). Ukázka vstupních dat je na obrázku 1.

Výstupem je primárně grafické okno knihovny *Turtle*, ve kterém se vykreslí výsledná hvězda formou černé obrysové čáry. Sekundárním výstupem jsou textové zprávy v konzoli, které informují o úspěšném dokončení vykreslování nebo o případných chybách. Příklad vizualizace je zobrazen na obrázku 2.

```
hon3.11.exe" "c:/Škola/_Úvod do programování/Zkouška/ukol_2_vizuali
Zadejte počet cípů (např. 5): 5
Zadejte poloměr kružnice opsané: 200
Zadejte poloměr kružnice procházející místem napojení cípů: 100
```

Obrázek 1 – Příklad vstupních dat



Obrázek 2 – Příklad výstupních dat

8. Závěr a možná vylepšení

Vytvořená aplikace splňuje zadání a spolehlivě vizualizuje n-cípé hvězdy na základě uživatelských parametrů. Využití analytické geometrie pro výpočet souřadnic se ukázalo jako robustní řešení, které je jednodušší než relativní navigace želvy. Objektový návrh kódu navíc usnadňuje případné budoucí úpravy.

Vhodným vylepšením by byla možnost volby barvy výplně hvězdy nebo automatické škálování velikosti útvaru tak, aby se vždy vešel do okna bez ohledu na zadané poloměry.

Literatura:

BAYER, T.: Programování 9 – Třídy a objekty. Přednáškové materiály. Přírodovědecká fakulta Univerzity Karlovy, Praha, https://web.natur.cuni.cz/~bayertom/images/courses/Prog1/programovani9.pdf (28. 1. 2026).

PILGRIM, M. (2012): Ponořme se do Pythonu 3. CZ.NIC, https://diveintopython3.py.cz/index.html (28. 1. 2026).

PYLADIES CZ (2017): Kurz Pythonu pro začátečníky – Brno jaro 2017. Naucse.python.cz, https://naucse.python.cz/2017/pyladies-brno-jaro-po/ (28. 1. 2026).

PYTHON SOFTWARE FOUNDATION (2026): turtle — Turtle graphics. Python 3.12 Documentation. URL: https://docs.python.org/3/library/turtle.html (28. 1. 2026).

WEISSTEIN, E. W. (2026): Star Polygon. MathWorld – A Wolfram Web Resource, https://mathworld.wolfram.com/StarPolygon.html (28. 1. 2026).

Zdrojový kód programu

```
#Vizualizace n-cípé hvězdy
#Vít Suchomel, 3. ročník, SGGI
#zimní semestr 2025/26
#Úvod do programování

import turtle
import math

max_cipu = 50
max_polomer = 400

class Hvezda:

  #Třída reprezentující n-cípou hvězdu.
  #Definuje parametry a umožňuje vykreslení pomocí želví grafiky.

  def __init__(self, pocet_cipu, polomer_opsane, polomer_napojeni):

    #Funkce inicializuje parametry hvězdy.

    self.n = pocet_cipu
    self.r_vnejsi = polomer_opsane
    self.r_vnitrni = polomer_napojeni

  def nakresli(self):

    #Funkce vykreslí hvězdu do grafického okna.
```

```
#Nejpreve zkontroluje vstupní data (limity a logickou správnost).
#Poté vypočítá souřadnice bodů pomocí goniometrických funkcí.
#Nakonec vykreslí hvězdu spojením bodů.
#Vrací True, pokud vše proběhle správně, jinak False.

if self.n < 3:
   print("Chyba: Nelze vykreslit hvězdu s méně než třemi cípy")
   return False

if self.r_vnejsi <= self.r_vnitrni:
   print("Chyba: Polměr opsané kružnice musí být větší než poloměr napojení cípů.")
   return False

if self.n > max_cipu:
    print(f"Chyba: Příliš mnoho cípů. Maximum je {max_cipu}.")
    return False

if self.r_vnejsi > max_polomer:
   print(f"Chyba: Poloměr je příiš velký. Maximuální poloměr je {max_polomer}.")
   return False

pero = turtle.Turtle()
pero.speed(0)
pero.pensize(2)

pocet_bodu = 2 * self.n

for i in range(pocet_bodu + 1):
```

```
            uhel = i * (2 * math.pi / pocet_bodu)

            if i % 2 == 0:
                aktualni_polomer = self.r_vnejsi
            else:
                aktualni_polomer = self.r_vnitrni

            x = aktualni_polomer * math.cos(uhel)
            y = aktualni_polomer * math.sin(uhel)

            if i == 0:
                pero.penup()
                pero.goto(x, y)
                pero.pendown()
            else:
                pero.goto(x, y)

        pero.hideturtle()
        return True

try:
    n = int(input("Zadejte počet cípů (např. 5): "))
    r_vnejsi = float(input("Zadejte poloměr kružnice opsané: "))
    r_vnitrni = float(input("Zadejte poloměr kružnice procházející místem napojení cípů: "))

    zadana_hvezda = Hvezda(n, r_vnejsi, r_vnitrni)

    if zadana_hvezda.nakresli() == True:
```

```
        print("Hvězda nakreslena")
        turtle.done()
    except ValueError:
        print("Chyba: Musíte zadat platná čísla.")
```