

ГеоДемонстратор

-Програмске парадигме-

Аутори (тим КриЛа):

Кристина Пантелић, 91/2016,

Лазар Васовић, 99/2016.

Математички факултет, 2019

Замисао

- ◆ Геометријске трансформације → афине, 2Д
- ◆ Приказ → једноставан, непосредан, интерактиван
- ◆ Разумевање → лако, детаљан испис на стандардни излаз, упутства и тумачења у менију, прикладни искачући прозори
- ◆ Употреба → Геометрија, Рачунарска графика, не претерано компликовано уопштавање у случај више димензија (3Д) или пројекције

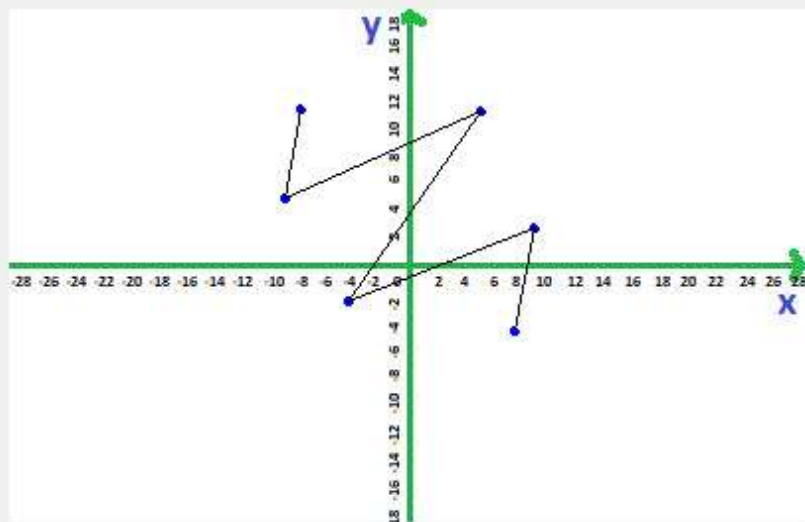
Реализација

- ◆ Графички кориснички интерфејс (ГКИ) → интерфејс Tsl/Tk
- ◆ Платно → координатни систем
- ◆ Задавање фигуре → догађаји миша
- ◆ Одабир параметара → поља за унос
- ◆ Трансформација → клик на дугме

GeoDemonstrator

Opcije Pomoć (H) Info (G)

Zakoračite u svet geometrijskih transformacija



Unosite tačke klikovima po platnu

Izaberite transformaciju

smicanje

Transformiši

x: 0.3

y: 0.1

θ:

Centar transformacije:

☐ centar platna

☐ centar mase

☒ uneta tačka

☐ Invertuj promenu

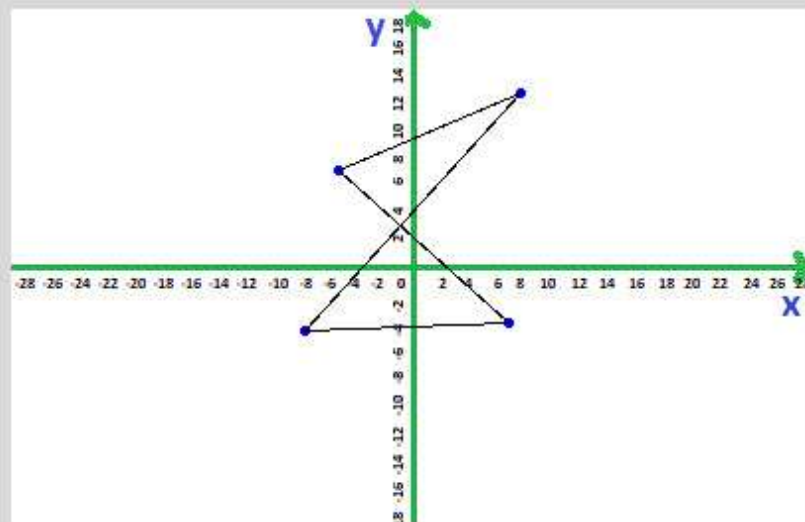
t1: 1

t2: 1

GeoDemonstrator

Opcije Pomoć (H) Info (G)

Zakoračite u svet geometrijskih transformacija



Transformišite figuru po želji

Izaberite transformaciju

rotacija

Transformiši

x:

y:

θ: -10

Centar transformacije:

☐ centar platna

☒ centar mase

☐ uneta tačka

☒ Invertuj promenu

t1: 0.32

t2: 2.89

Оквир апликације

- ◆ Програмски језик → Пајтон (Python3)
- ◆ Оперативни систем → разни, прилично портабилан језик пошто се интерпретира
- ◆ Рилиз верзија → постоји .exe за Виндоус 10
- ◆ Додатна погодност → потпуна подршка за цео Јуникод, па и српску ћирилицу и латиницу

Оквир апликације

- ◆ Библиотеке необрађиване у оквиру факултетских курсева → tkinter, threading, operator, time, inspect, types
- ◆ Одраније познате библиотеке → sys, os, math, functools, random, copy
- ◆ Парадигме и концепти → покушај обухватања свега што Пајтон подржава, а да има смисла
- ◆ Декларативни концепти → нису се уклопили

Императивни, процедурални, објектно-оријентисани концепти

- ◆ Променљиве, функције, низање наредби, опис процедуре...
- ◆ Класе, наслеђивање, полиморфизам, преоптерећивање оператора, МИКСИНИ...

```
# Nosilac programa je klasa GeoDemonstrator, koja
# nasleđuje grafičku klasu Tk iz modula tkinter, kao
# i mixin klase radi razdvajanja funkcionalnosti
class GeoDemonstrator(Tk, GeoMixinTrans, GeoMixinHelp):
    # Konstruktor aplikacije
    def __init__(self):
        # Log poruka o pokretanju aplikacije
        print('Dobro došli u aplikaciju GeoDemonstrator!')

        # Pozivanje konstruktora roditeljske klase
        super().__init__()
```

...

Скрипт концепти

```
# Mali interpretator koji se oslanja
# na pozivanje Pajtonovog interpretera;
# u pitanju je poznata RE(P)L petlja
# iz koje se izlazi EOF-om (CTRL+D)
while True:
    try:
        exec(input())
    except EOFError:
        print('Kraj ulaza.')
        break
    except:
        print('Pokušajte ponovo.')
```

Обрада динамичких наредби (наредби представљених ниском, о чијој се синтаксичкој и семантичкој исправности ништа не зна), `exec`, `eval`, `try-except`

Скрипт концепти

```
# Funkcija za dohvatanje vrednosti promenljive; pokušava
# se evaluacija vrednosti ili ispaljuje izuzetak
def uzmi_prom(self, prom):
    try:
        if prom == 'x':
            return float(eval(self.x_koord.get()))
        elif prom == 'y':
            return float(eval(self.y_koord.get()))
        elif prom == 'u':
            return float(eval(self.ugao.get()))
        elif prom == 't1':
            return float(eval(self.t1_koord.get()))
        else:
            return float(eval(self.t2_koord.get()))
    except:
        showerror('Greška', 'Loši parametri transformacije!')
        return float('nan')
```

Функционални концепти

- ◆ Фје вишег реда (map, filter, partial), ламбде, декоратори, апстракција листи (генератори)

```
# Може бити linearna matrica dimenzija 2x2
if len(arg) is 2 and all(map(partial(eq, 2), map(len, arg))):
    return ((float(arg[0][0]), float(arg[0][1]), 0),
            (float(arg[1][0]), float(arg[1][1]), 0),
            (0, 0, 1))

# Inače mora biti objekat dimenzija 3x3
if len(arg) is not 3 or any(map(partial(ne, 3), map(len, arg))):
    raise TypeError
```

Функционални концепти

множење две матрице

```
Geom(tuple(tuple(sum(self[i][k] * drr[k][j] for k in range(3))  
                  for j in range(3)) for i in range(3)))
```

матрица са тачком

```
tuple(sum(self[i][j] * drr[j] for j in range(3)) for i in range(3))
```

```
# Tačka predstavljena koordinatama; funkcionalno  
# dekorisana tako da podržava totalno uređenje  
@total_ordering  
class Tačka:
```

Компонентни концепти

- ◆ ГКИ као скуп графичких компоненти повезан са независно имплементираним радњама
- ◆ Одвојени модули као уговорени интерфејси

 [geom.py](#)

 [omot.py](#)

 [gki.py](#)

 [test.py](#)

 [main.py](#)

 [trans.py](#)

 [nit.py](#)

 [util.py](#)

Програмирање вођено догађајима, реактивни и конкурентни концепти

- ◆ Концепт главне петље, ослушкивање и обрада догађаја као интеракција ГКИ са корисником
- ◆ Промена стања програма као реакција на корисничку акцију, могућност представљања стања неке графичке компоненте као резултата логичке формуле
- ◆ Вишенично програмирање, паралелизовани подели па владај алгоритам за налажење конвексног омотача скупа унетих тачака

Генерички концепти

- ◆ Образац обраде независан од улазних типова
- ◆ Разни типови полиморфизма: магични методи, преоптерећивање оператора...

[illegible]

Генерички концепти

```
# Ispitivanje da li je argument tačka
def point(arg):
    if isinstance(arg, Tačka):
        return deepcopy(arg.mat)

# Sve vrednosti moraju biti numeričke
dr = tuple(map(float, arg))

# Dalje razmatranje po dužini argumenta
if len(dr) is 2:
    return (dr[0], dr[1], 1)
elif len(dr) is 3:
    return (dr[0]/dr[2], dr[1]/dr[2], 1)
else:
    raise TypeError
```

```
# Uobičajeno sabiranje dve tačke ili
# tačke sa skalarom (vektORIZACIJA)
def __add__(self, dr):
    if isinstance(dr, (int, float)):
        return Tačka(self[i]+dr for i in range(2))
    else:
        drr = Tačka.point(dr)
        return Tačka(self[i]+drr[i] for i in range(2))
```

Метапрограмирање

- ◆ Динамичко писање програма, писање унутар њега самог, прављење и употреба метакласе

```
# Pravljenje metaklase prosledivanjem odgovarajućih
# argumenata konstruktoru metaklase (to je klasa koja
# instancira klase, a ne objekte u užem smislu) type:
# ime klase, torka baznih klasa, rečnik klase
MetaNit = type('MetaNit', tuple([type]), {})

# Vezivanje definisanog metoda za metaklasu
MetaNit.__call__ = poziv

# Pravljenje klase Nit kao klase čija je
# metaklasa MetaNit, a natklasa Thread
Nit = MetaNit('Nit', tuple([Thread]), {})

# Postavljanje lambda metoda za stringovnu predstavu
Nit.__str__ = lambda self: 'Nit za {} nad {}'.format(self._target.__name__, self._args)

# Vezivanje metoda napravljene klase
# za one napisane za klasu StaraNit
Nit.__init__ = StaraNit.__init__
```


Рефлексија, ин(тро)спекција

- ◆ Читање и даља употреба (преписивање) метода и атрибута неке класе или објекта, могућност програма да сам себе анализира

```
# Ostalo se dodaje pomoću inspekcije
for metod in getmembers(StaraNit, isfunction):
    if not metod[0].startswith('_'):
        # Izdvajanje metoda
        met = getattr(StaraNit, metod[0])

        # Dodeljivanje napravljenog metoda
        # exec('Nit.{0} = fja'.format(metod[0]))

        # Pravljenje duboke kopije metoda
        setattr(Nit, metod[0], FunctionType(met.__code__, met.__globals__,
                                              met.__name__, met.__defaults__, met.__closure__))
```



Хвала на пажњи!