Nesprávne a správne triangulácie

Bohdan Jóža

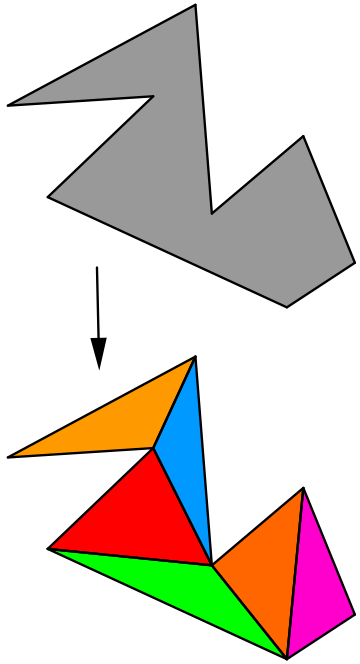
doc. RNDr. Andrej Ferko, PhD.

Katedra Informatiky
Fakulta Matematiky, Fyziky a Informatiky
Univerzita Komenského

Triangulácia

Neformálna definícia problému

Rozdeľ mnohouholník na trojuholníky.



História

1911 Lennes, prvý algoritmus

1978 Garey, prvý algoritmus rýchlejší ako $O(n^2)$

1988 Tarjan a van Wyk, $O(n \log \log n)$

1990 Chazelle, lineárny algoritmus

Riešenie (Lennes)

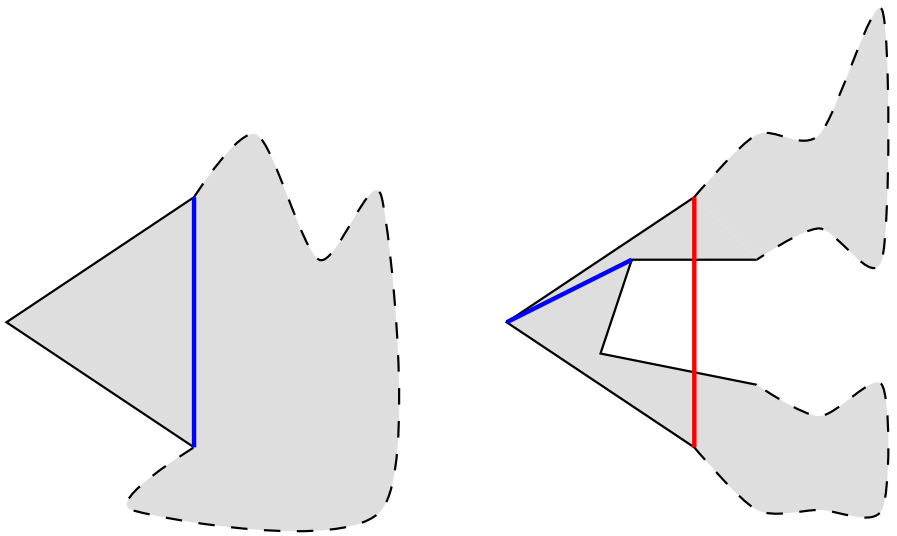
Každý mnohouholník s viac ako 3 vrcholmi obsahuje vnútornú diagonálu.

Toto umožňuje Divide & Conquer algoritmus:

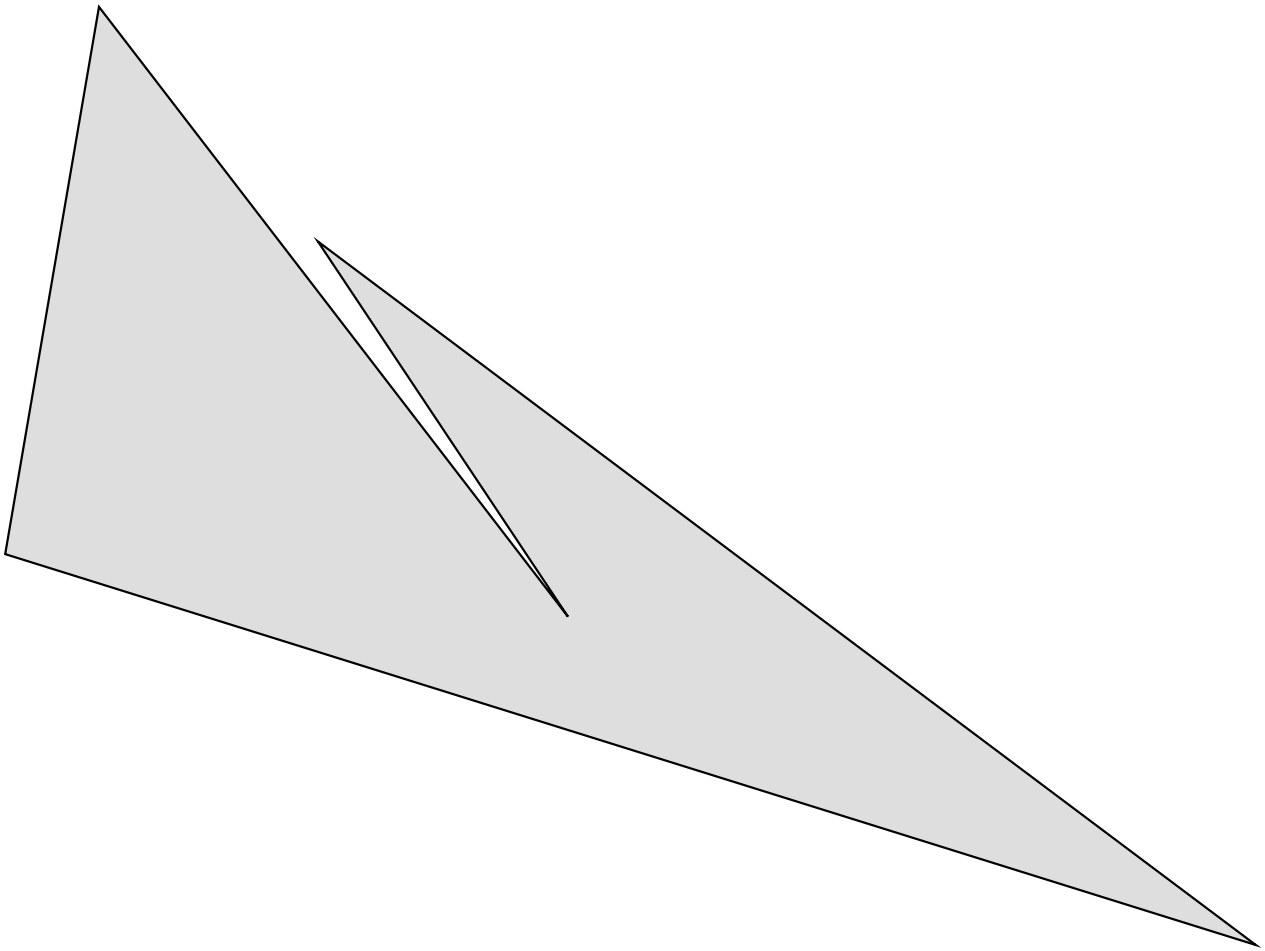
1. Nájdi nejakú vnútornú diagonálu
2. Rozdeľ ňou mnohouholník na dva nové
3. Rekurzívne ich trianguluj

Nájdi vnútornú diagonálu (Sellares, Toussaint)

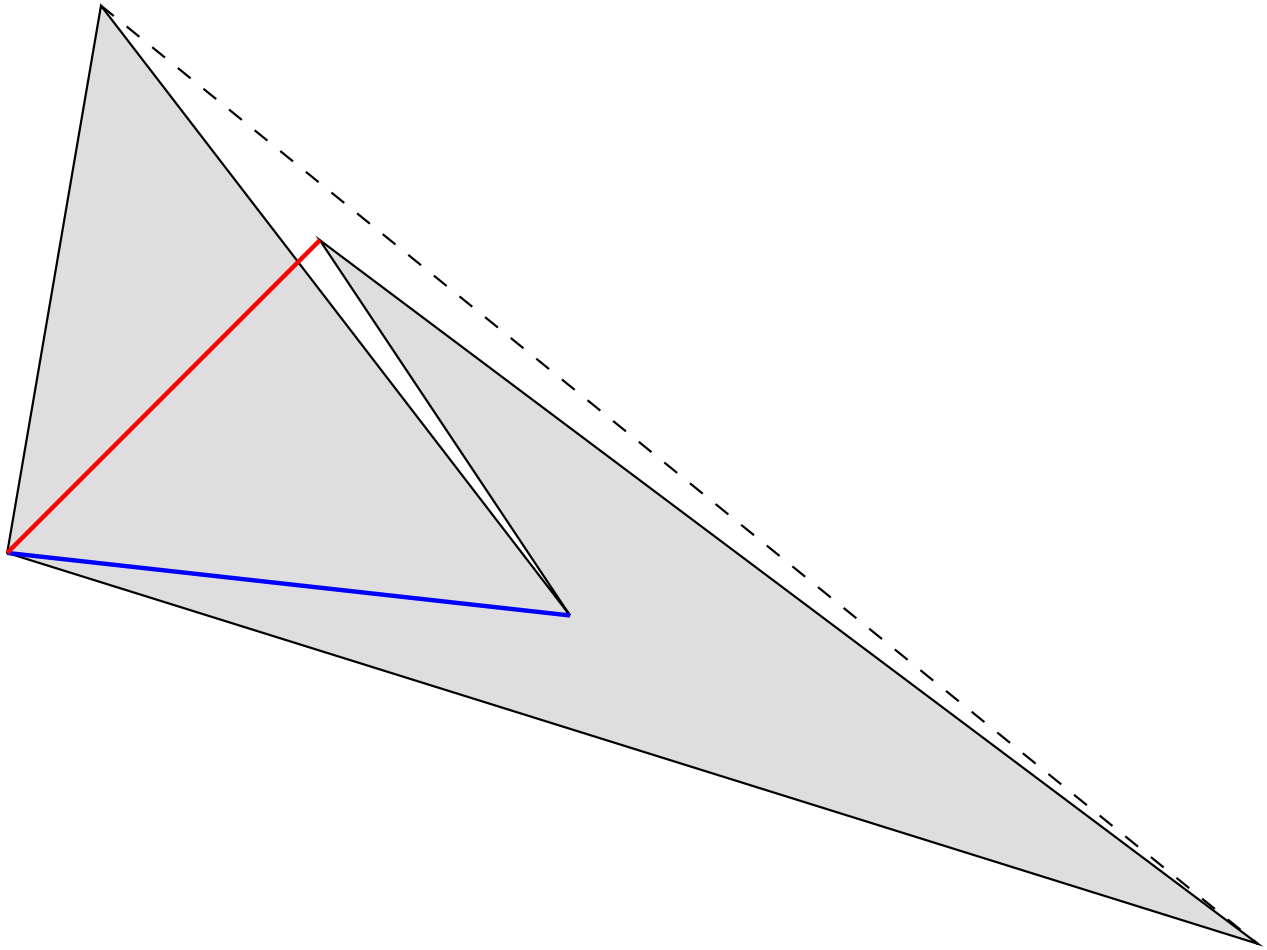
1. Nájdi konvexný vrchol
2. Ak jeho trojuholník neobsahuje ďalšie vrcholy, protiľahlá strana je diagonála
3. Inak jeden z vrcholov vnútri trojuholníka tvorí s vybratým vrcholom diagonálu



Minimálny protipríklad



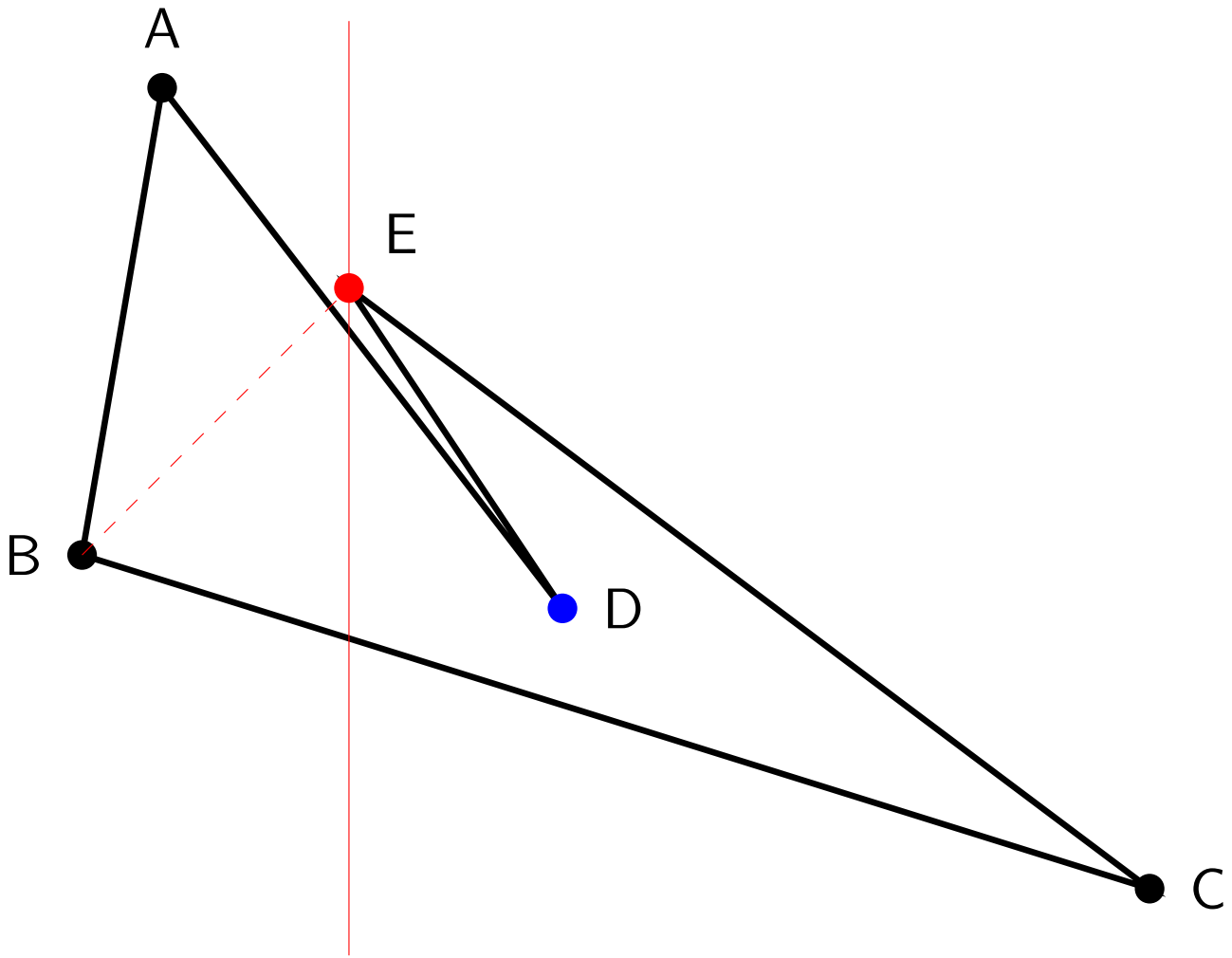
Minimálny protipríklad



Nesprávne algoritmy

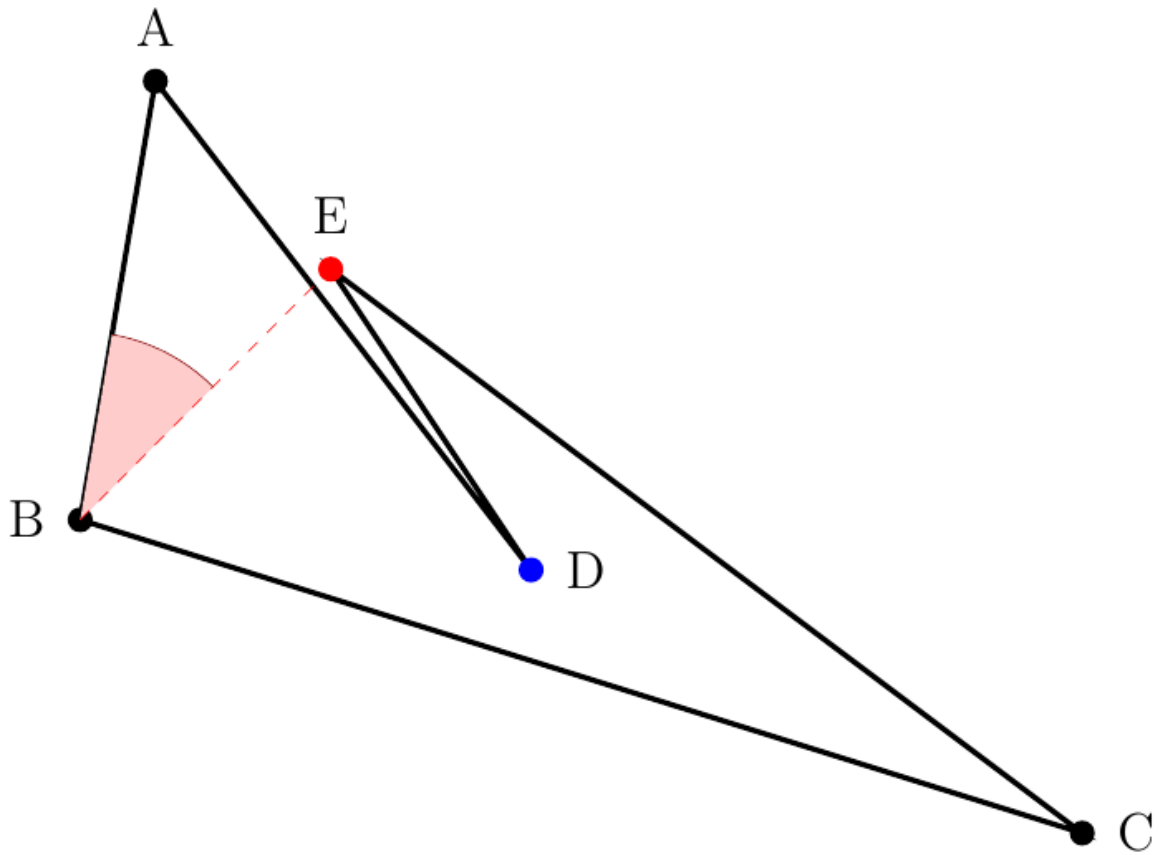
Translácia

Zametanie zvislou čiarou (ten najľavejší)

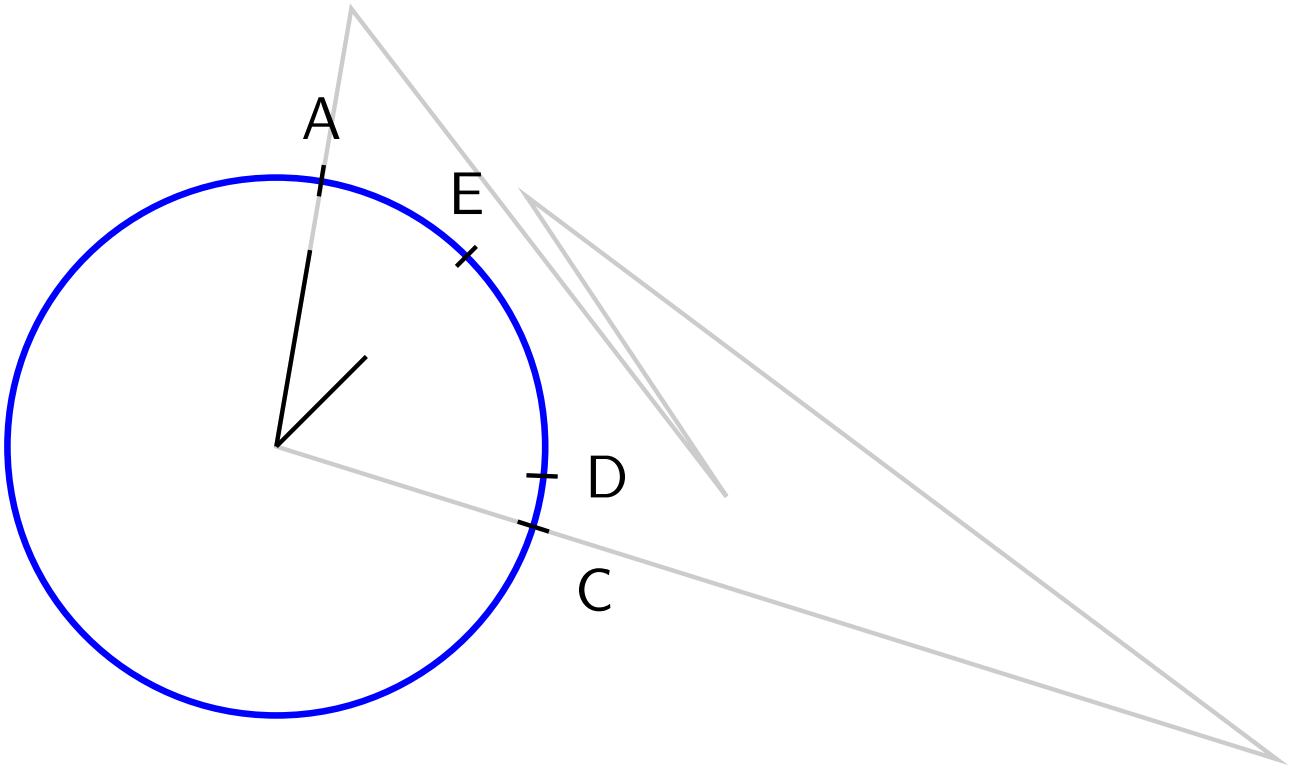


Rotovanie

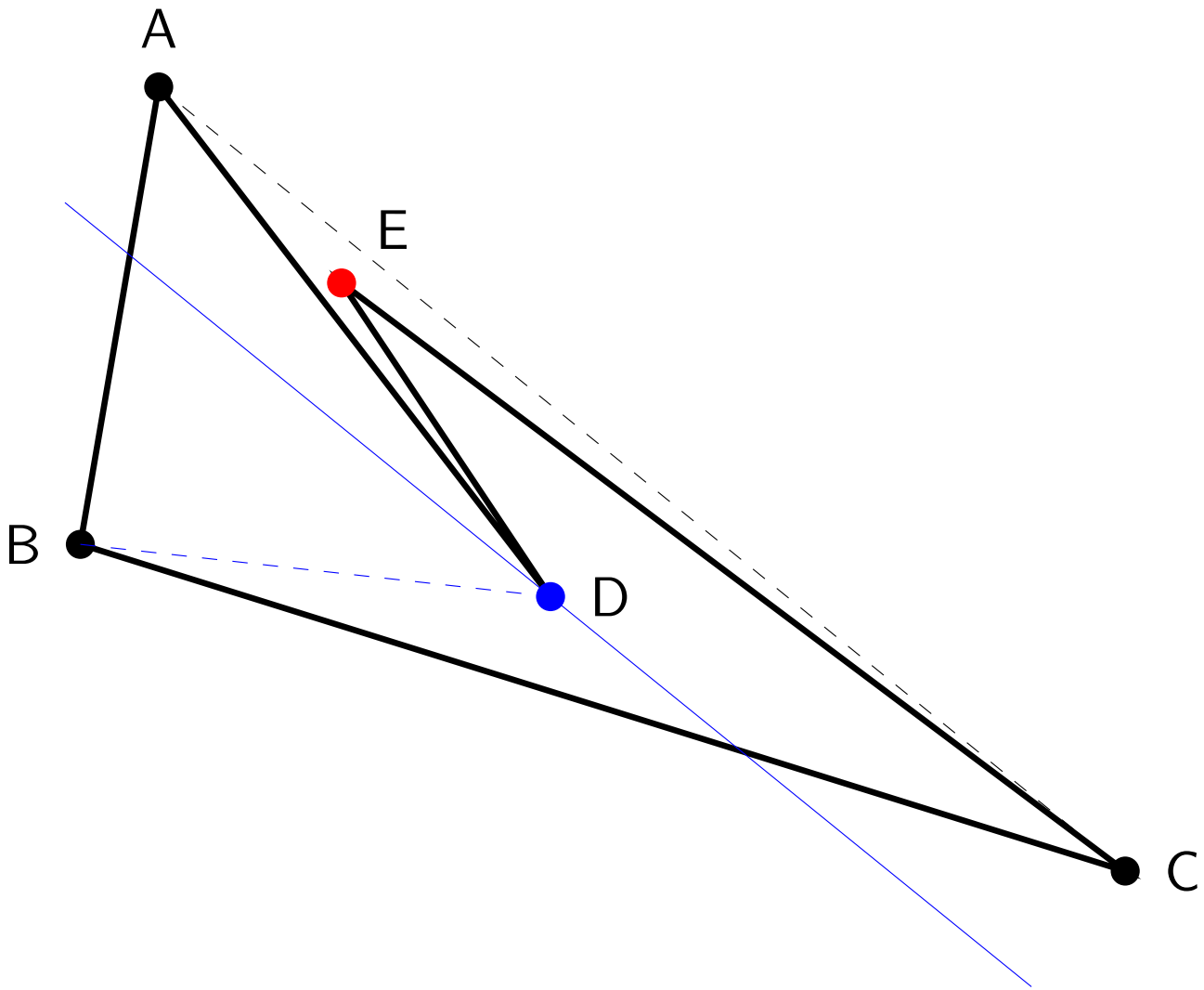
Zametanie rotujúcou čiarou okolo B



Metafora – hodiny

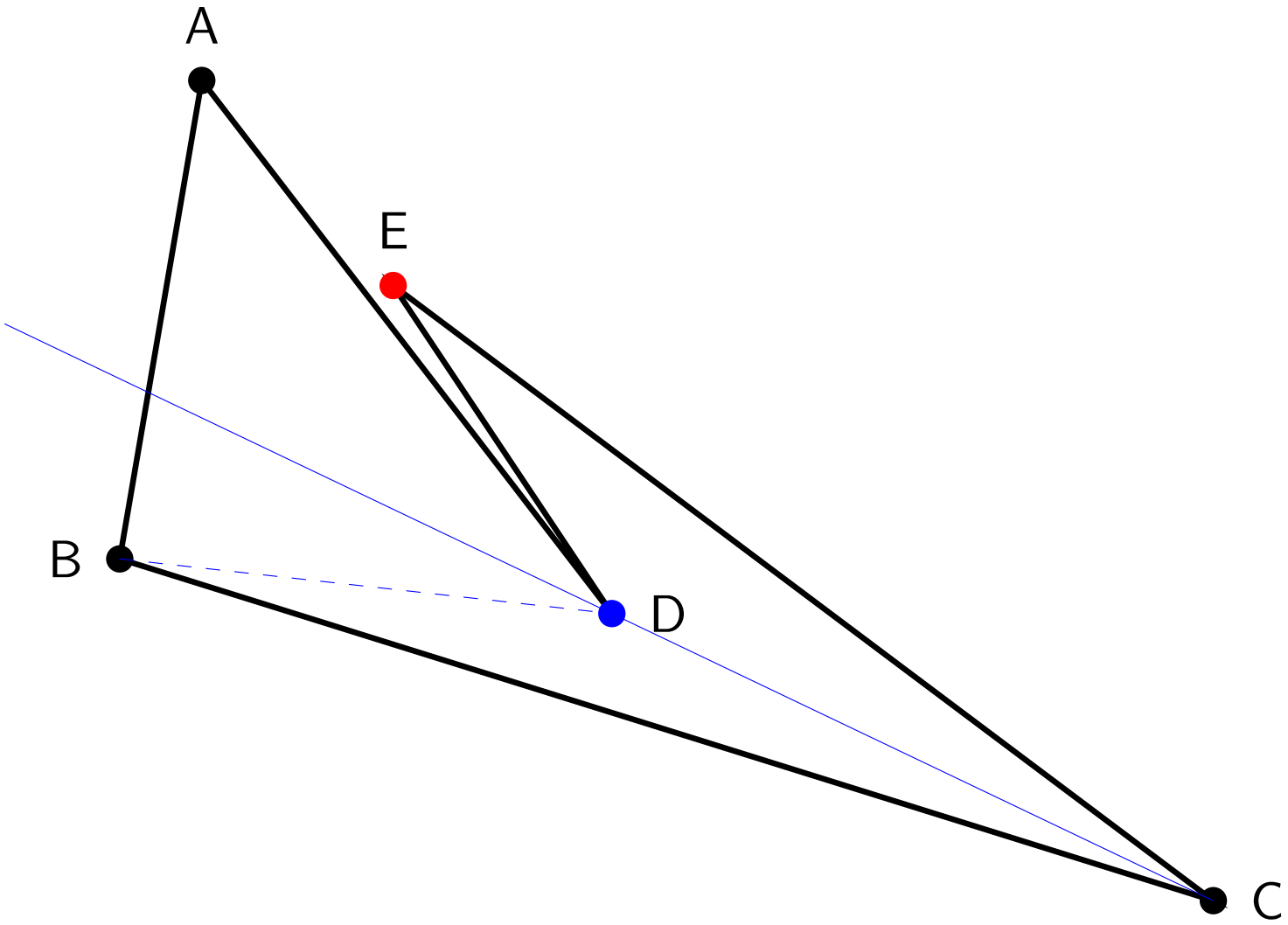


Správna translácia



Správna rotácia

Stred presunúť do C alebo A.



Vizualizácia

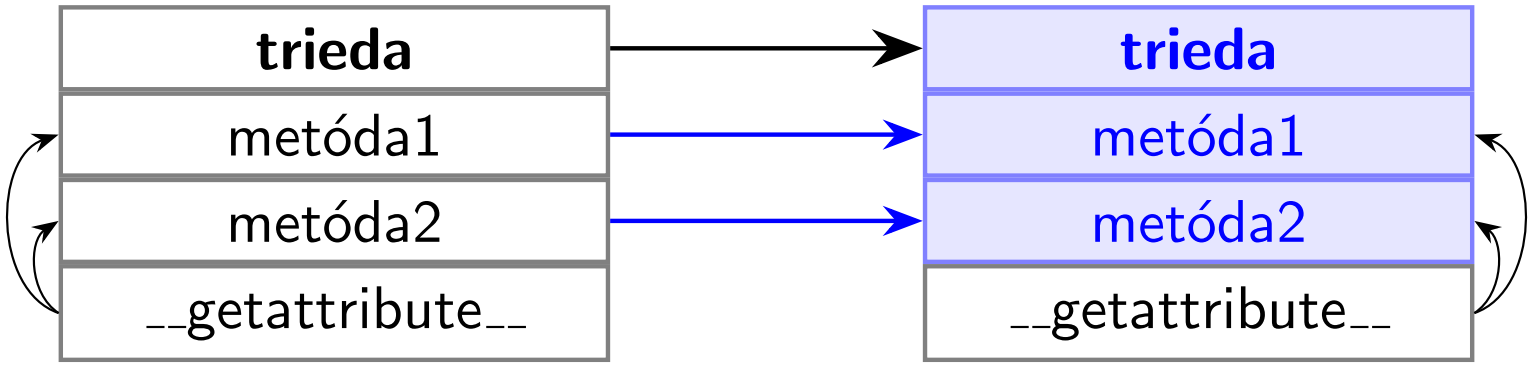
Wizer

Jednoduchá vizualizácia algoritmov na mnohouholníkoch.

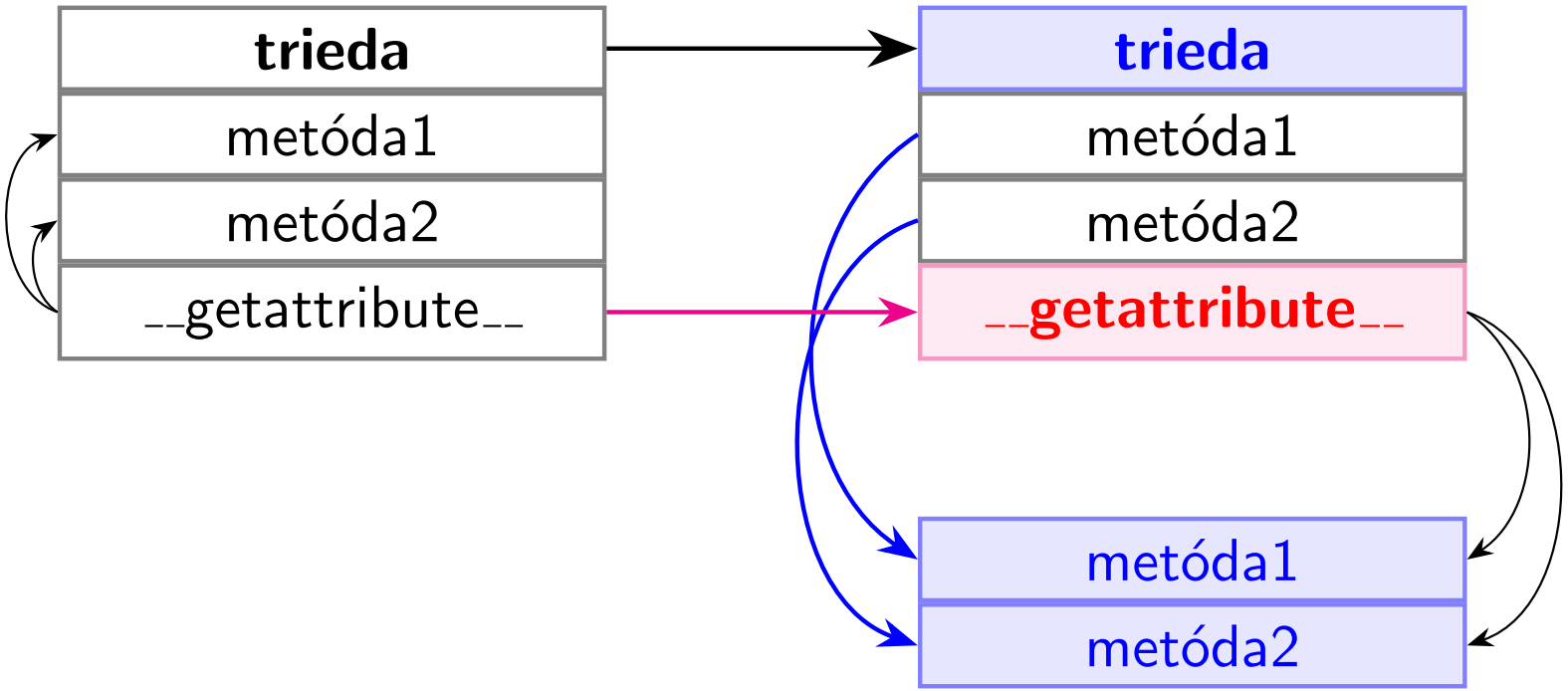
Ciele:

- jednoducho – bez drastických zmien v kóde algoritmu
- všeobecne – potenciálna možnosť vizualizovať iné druhy algoritmov

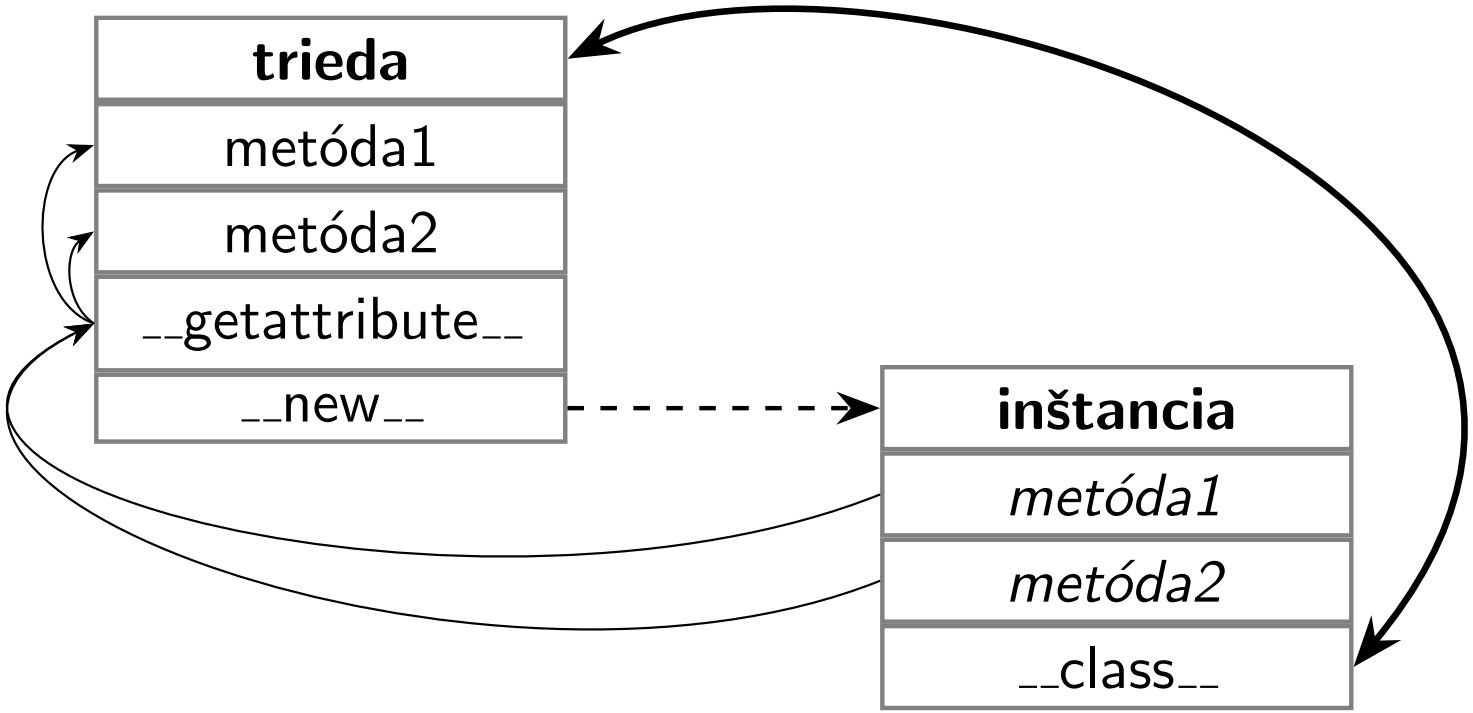
Dekorácia tried



Dekorácia tried

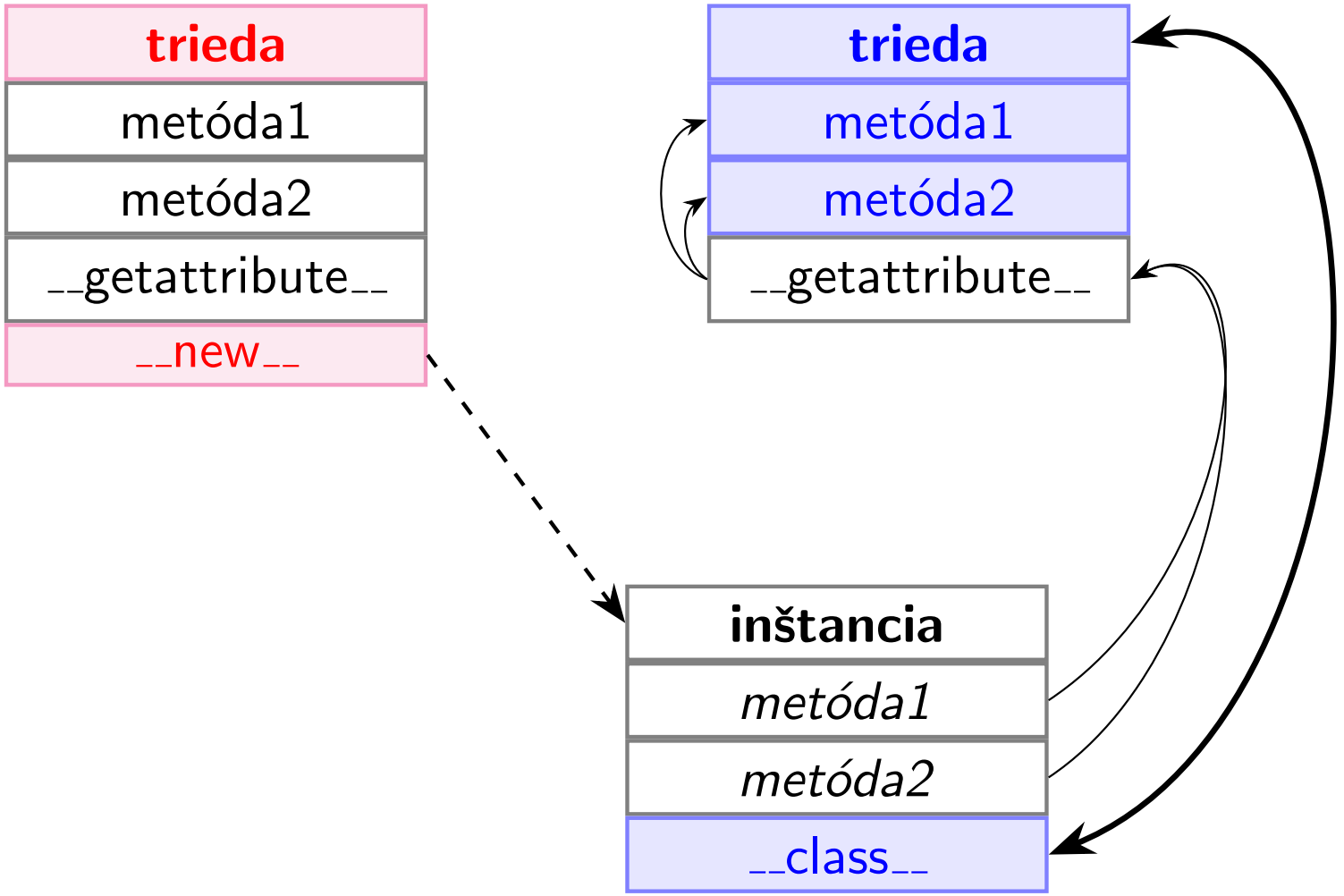


Logovanie


trieda
metóda1
metóda2
__getattribute__
__new__
inštancia
metóda1
metóda2
__class__

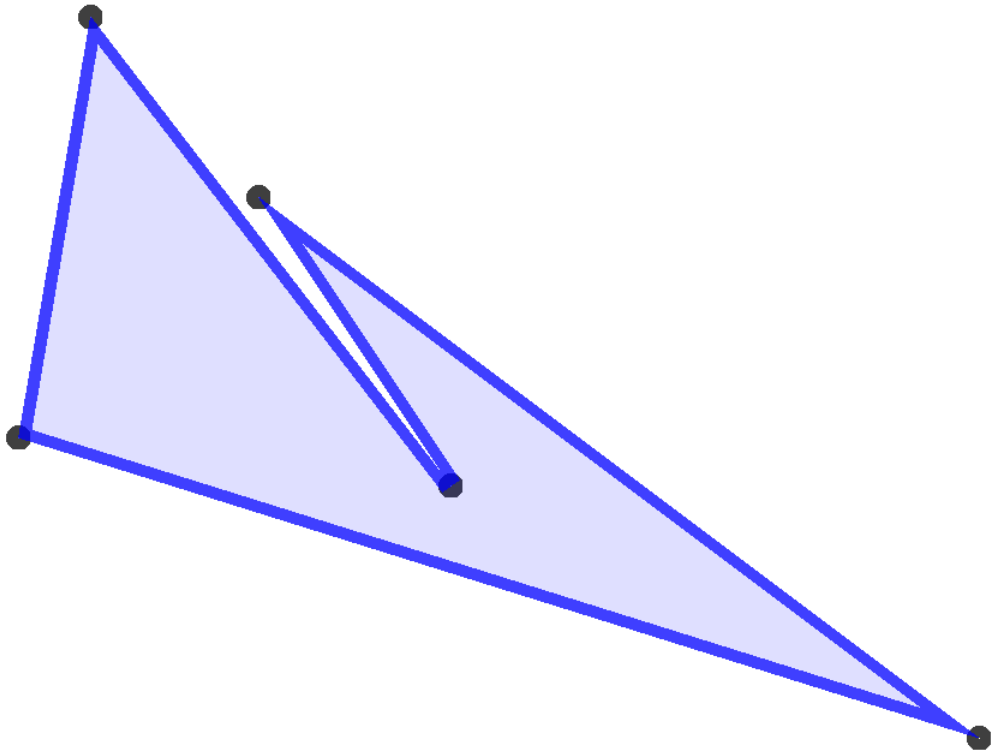
Logovanie

Transformovanie atribútov tried, aby zaznamenávali každé volanie.



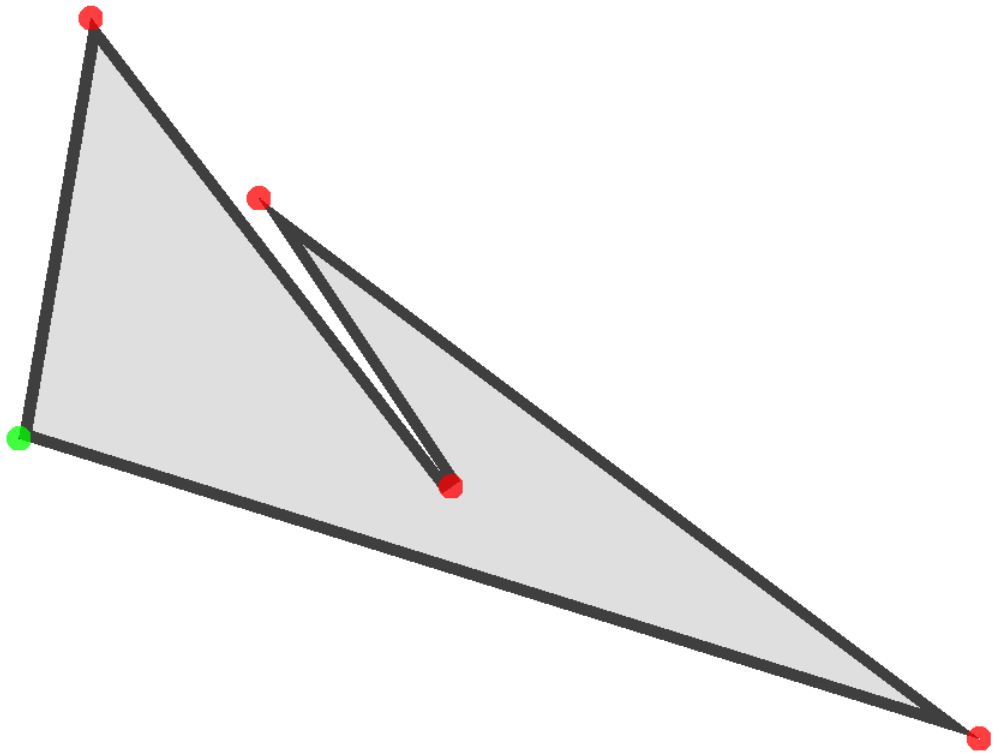
Wizer

```
('polygon', [(1.0, 4.5), (9.0, 2.0), (3.0, 6.5), (4.6, 4.1), (1.6, 8.0)])
___enter (return)
```



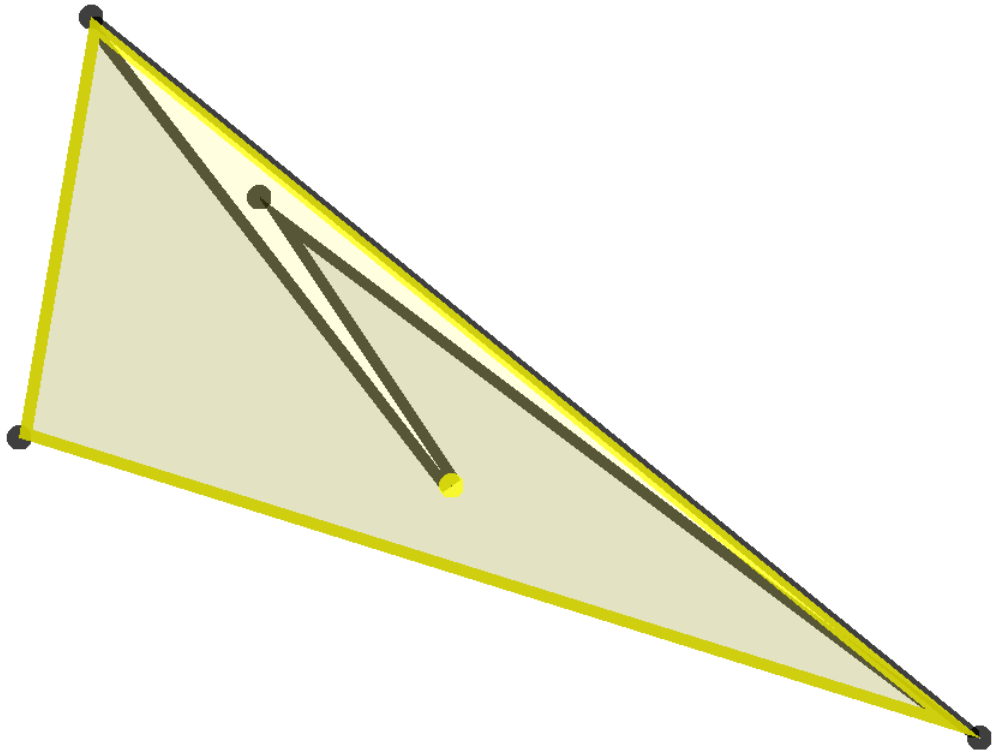
Wizer

```
function call
left_most (return)
args
 ['Point2D(1, 9/2)', 'Point2D(9, 2)', 'Point2D(3, 13/2)', 'Point2D(23/5, 41/10)', 'Point2D(8/5, 8)']
Point2D(1, 9/2)
```



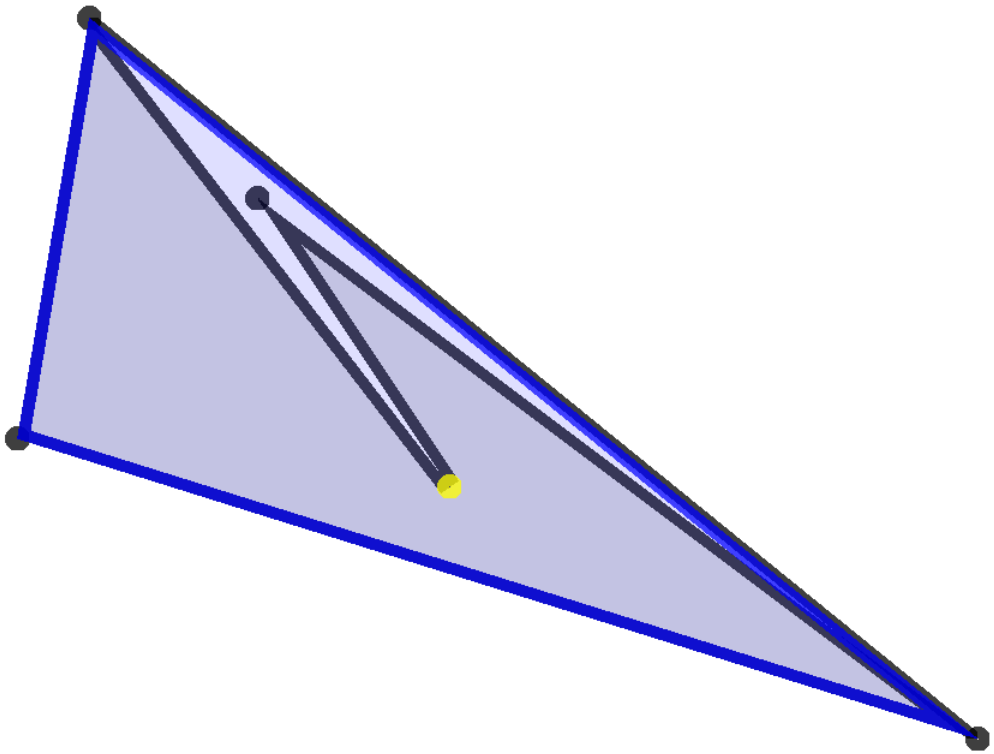
Wizer

```
function call
is_inside (call)
args
 Point2D(23/5, 41/10)
 Triangle(Point2D(8/5, 8), Point2D(1, 9/2), Point2D(9, 2))
```



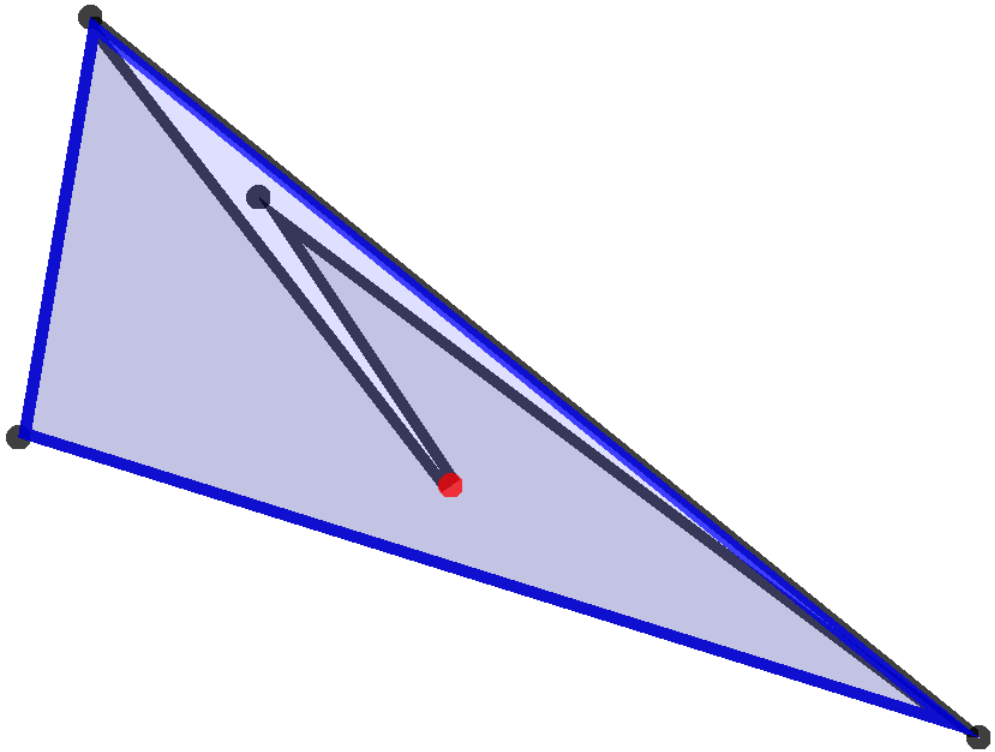
Wizer

```
('polygon', [(1.6, 8.0), (1.0, 4.5), (9.0, 2.0)])
encloses_point (call)
args
 Point2D(23/5, 41/10)
```



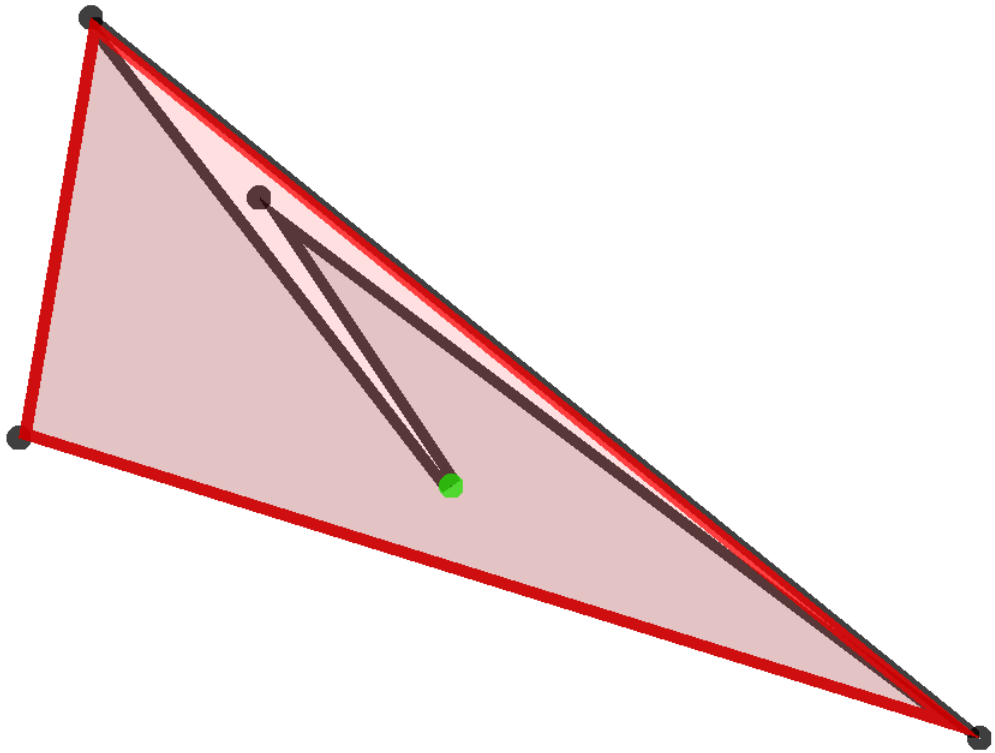
Wizer

```
('polygon', [(1.6, 8.0), (1.0, 4.5), (9.0, 2.0)])
encloses_point (return)
args
 Point2D(23/5, 41/10)
True
```



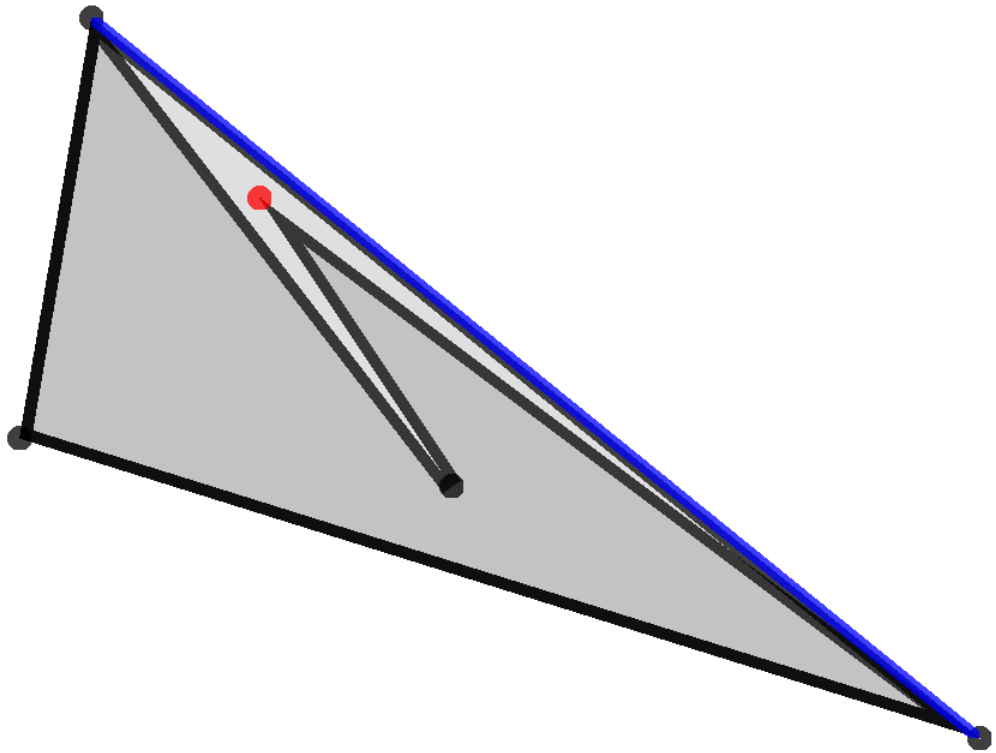
Wizer

```
function call
is_inside (return)
args
 Point2D(23/5, 41/10)
 Triangle(Point2D(8/5, 8), Point2D(1, 9/2), Point2D(9, 2))
Point2D(23/5, 41/10)
```



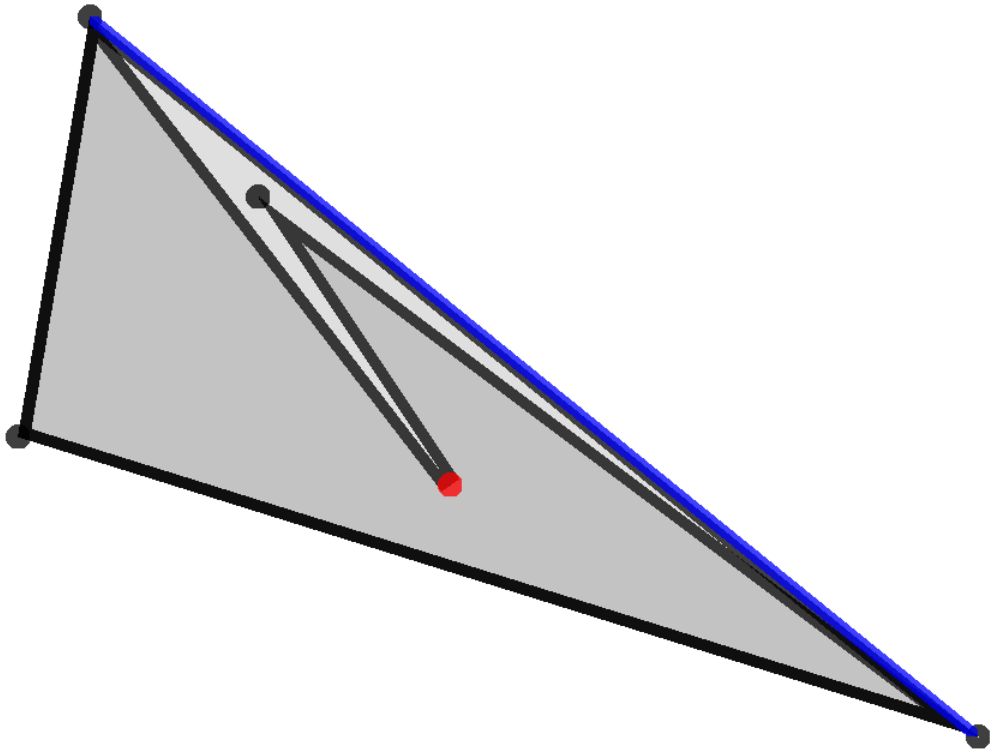
Wizer

```
('segment', ((1.6, 8.0), (9.0, 2.0)))
distance (return)
args
 Point2D(3, 13/2)
27*sqrt(2269)/4538
```



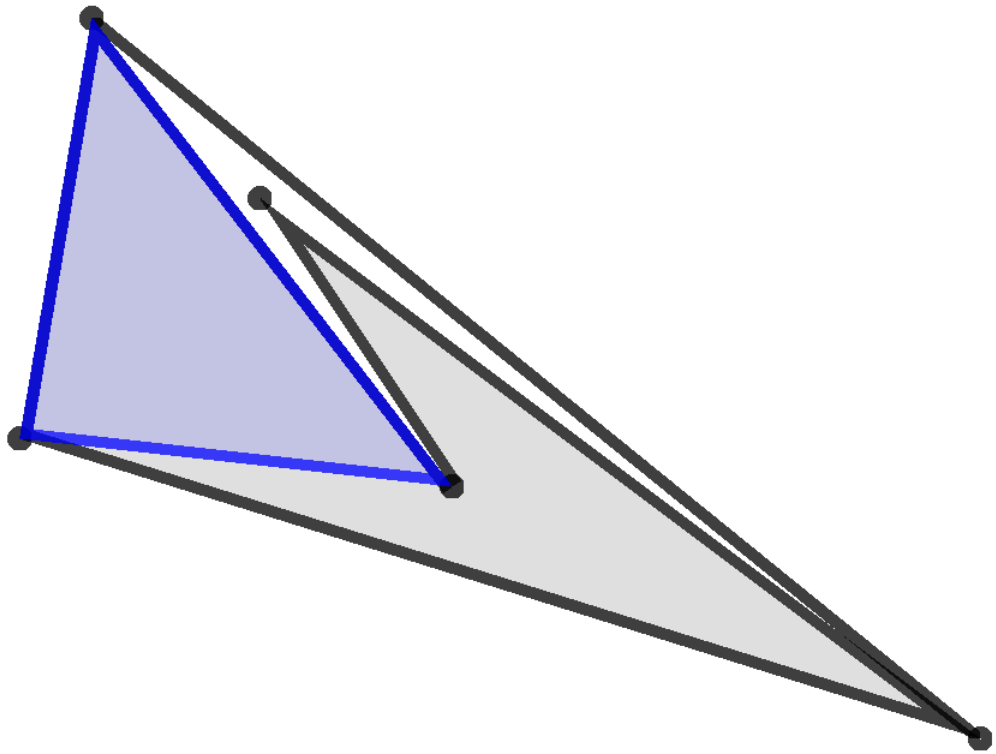
Wizer

```
('segment', ((1.6, 8.0), (9.0, 2.0)))
distance (return)
args
 Point2D(23/5, 41/10)
543*sqrt(2269)/22690
```



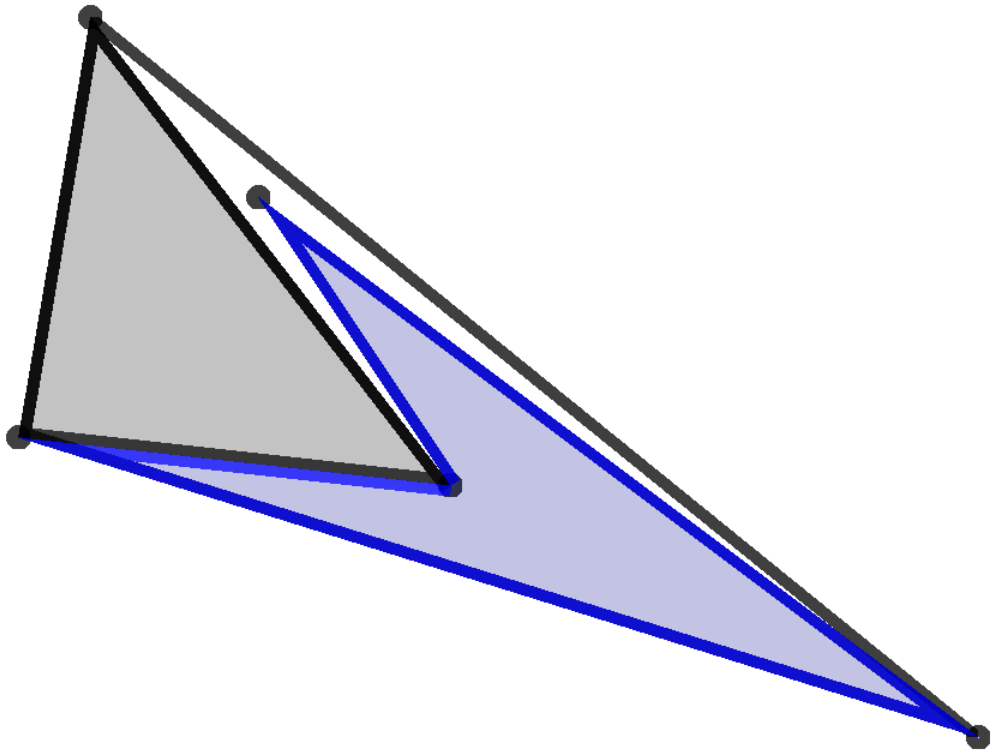
Wizer

```
('polygon', [(1.0, 4.5), (4.6, 4.1), (1.6, 8.0)])
___enter (return)
```



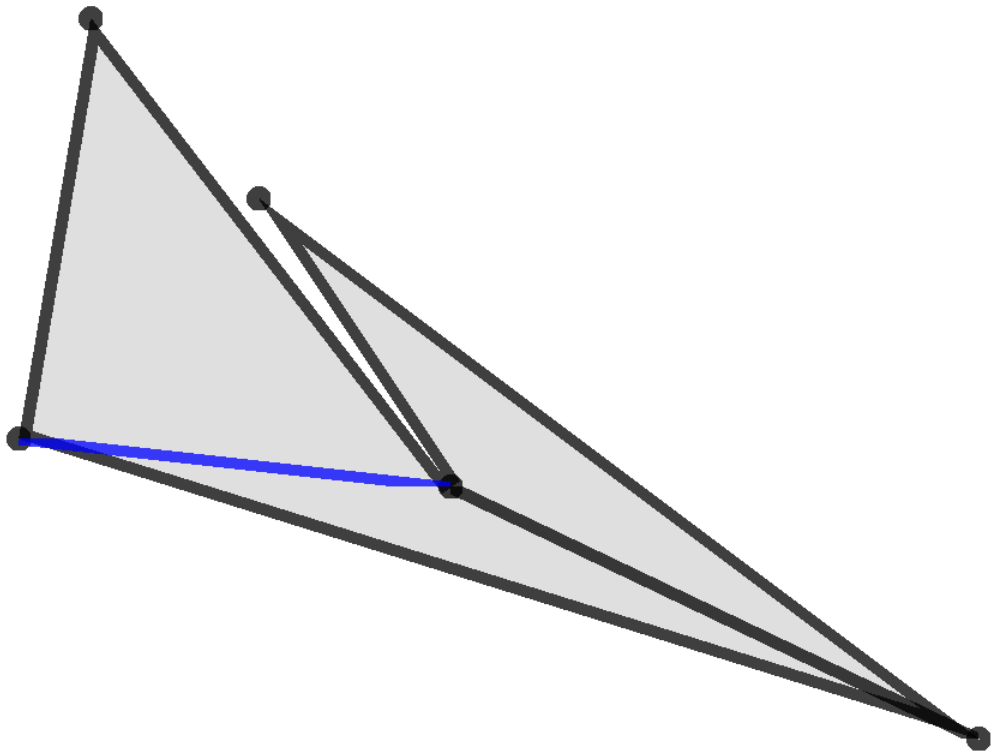
Wizer

```
('polygon', [(1.0, 4.5), (9.0, 2.0), (3.0, 6.5), (4.6, 4.1)])
___enter (return)
```



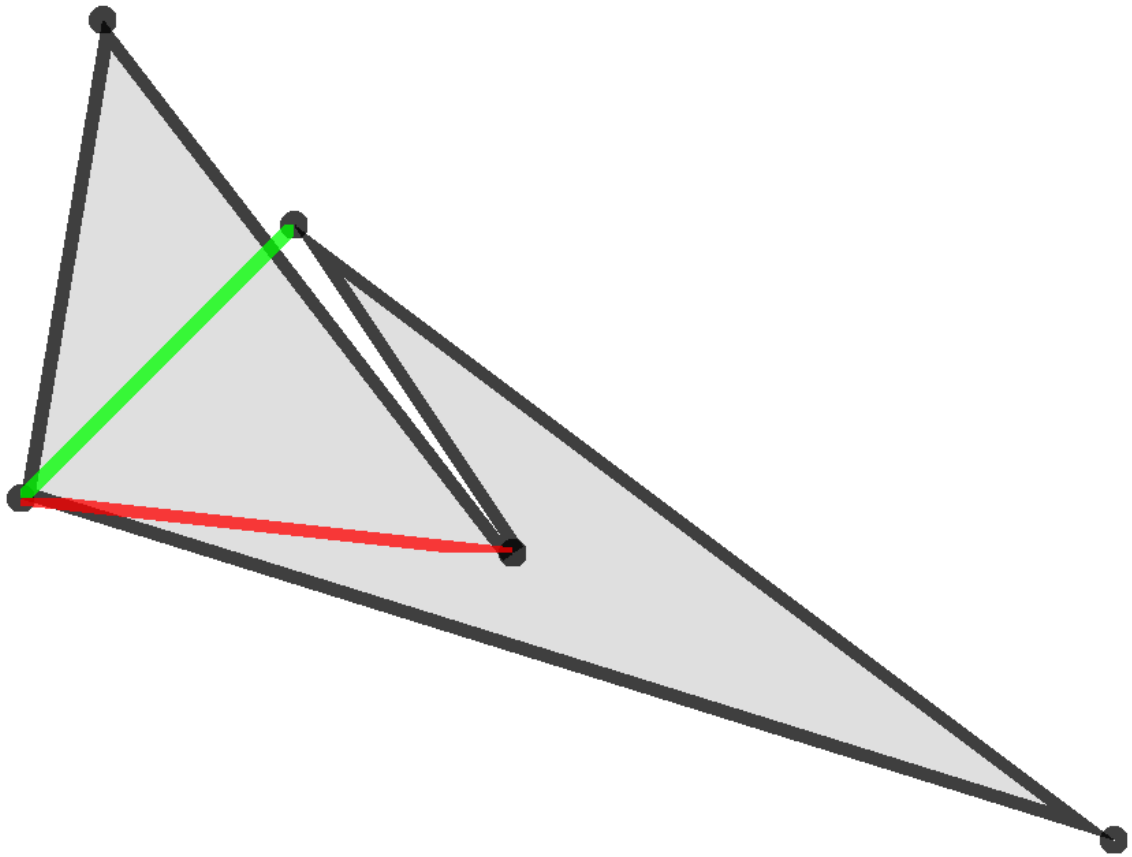
Wizer

```
('segment', ((1.0, 4.5), (4.6, 4.1)))
___enter (return)
```



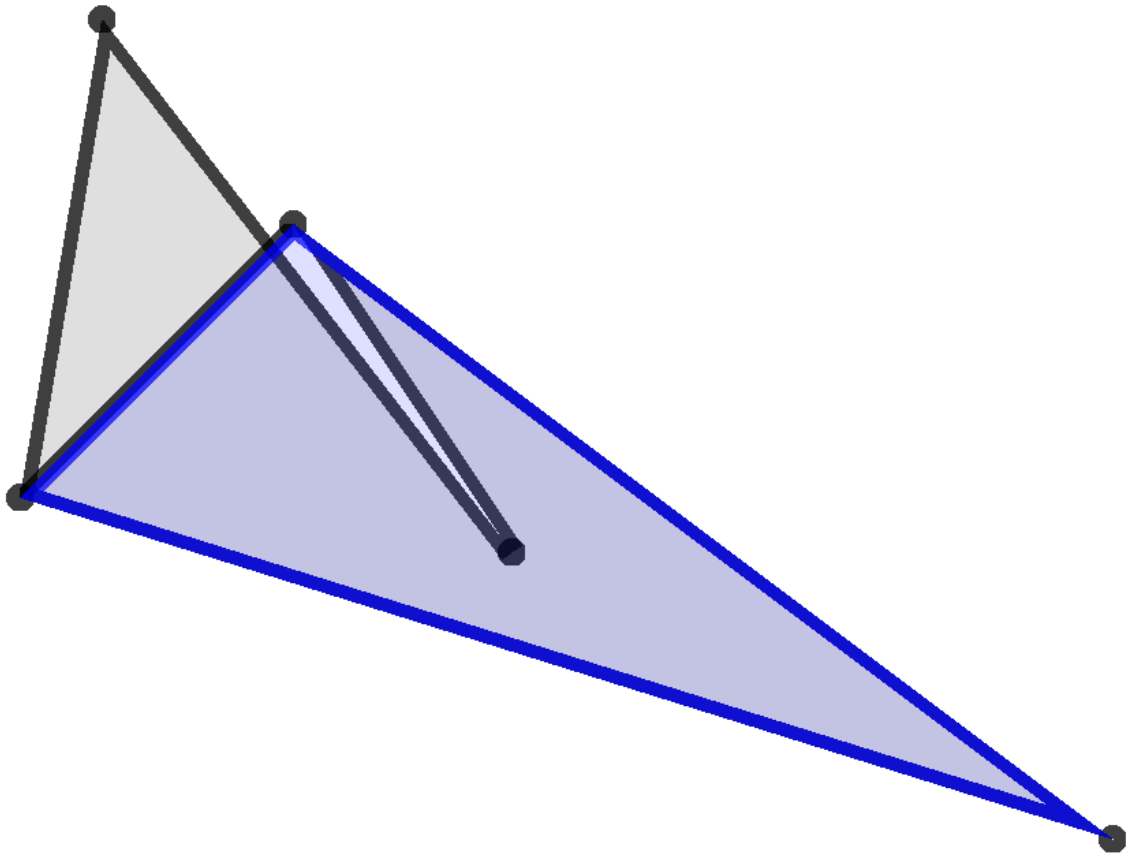
Wizer

```
function call (return)
.compare_lengths
args
  Segment_2(PointC2(1, 4.5), PointC2(4.6, 4.1))
  Segment_2(PointC2(1, 4.5), PointC2(3, 6.5))
Segment_2(PointC2(1, 4.5), PointC2(3, 6.5))
```



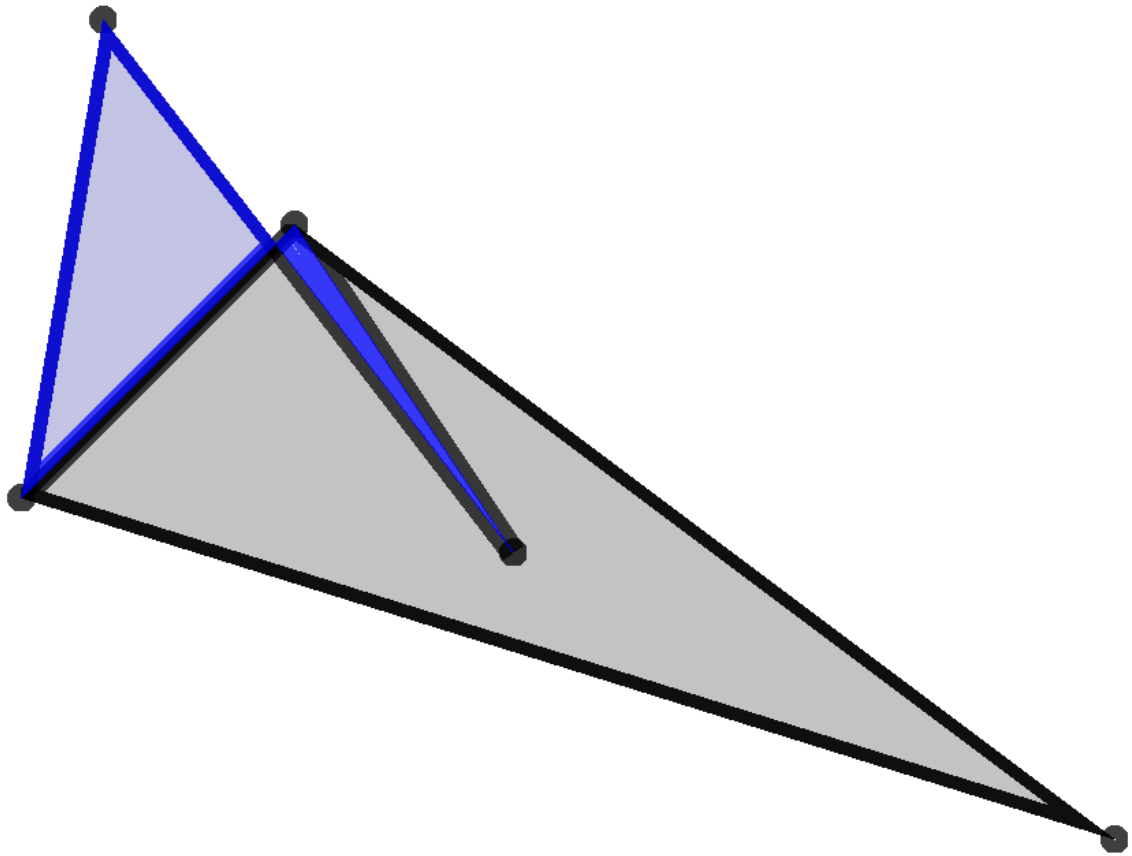
Wizer

```
('polygon', [[1.0, 4.5], [9.0, 2.0], [3.0, 6.5]])
.___enter
```



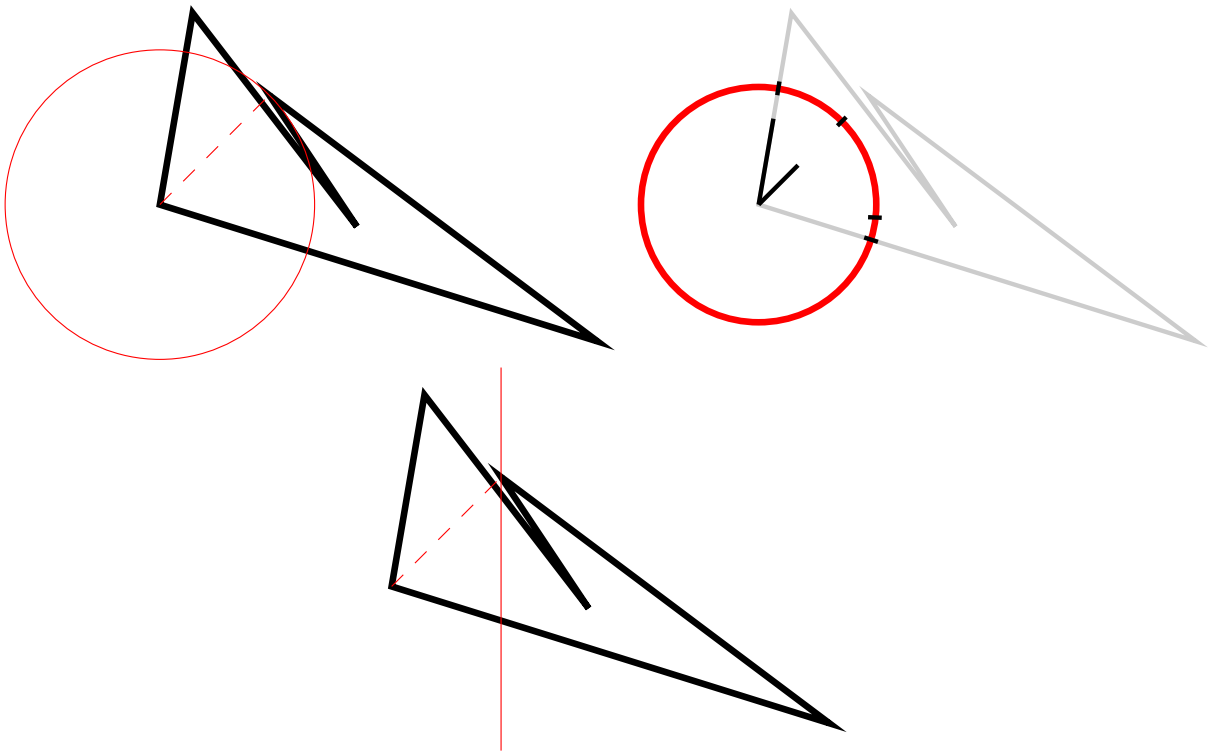
Wizer

```
('polygon', [[3.0, 6.5], [4.6, 4.1], [1.6, 8.0], [1.0, 4.5]])
.___enter
```



Záver

Vysvetlili sme problém triangulácie a niektoré nesprávne a správne riešenia.



Vytvorili sme prostredie na vizualizáciu algoritmov na mnohouholníkoch.