ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA STAVEBNÍ, OBOR GEODÉZIE A KARTOGRAFIE
KATEDRA GEOMATIKY

název předmětu

ALGORITMY V DIGITÁLNÍ KARTOGRAFII

číslo úlohy	název úlohy			
1	Geometrické vyhledávání bodů			
školní rok	studijní skupina	zpracovali:	datum	klasifikace
2020/21	60	Michal Zíma, Tomáš Lauwereys	22.10. 2020	

Zadání

Anotace:

Cílem úlohy je vytvořit aplikaci v prostředí QT, která bude umět načítat textový soubor vrchů polygonů ve formátu .txt. Dále bude analyzovat polohu bodu, který uživatel vytvoří vůči načteným polygonům. Při výsledku "bod leží uvnitř polygonu", program zvýrazní danný polygon. K analýze polohy bodu vůči polygonu má uživatel na výběr algoritmus Ray Crossing Algorithm nebo Winding Number Algorithm.

Přesné zadání:

Vstup: Souvislá polygonová mapa n polygonů $\{P_1, ..., P_n\}$, *analyzovaný bod q.*

Výstup: P_i, $q \in P_i$.

Nad polygonovou mapou implementujete následující algoritmy pro geometrické vyhledávání:

- Ray Crossing Algorithm (varianta s posunem těžiště polygonu).
- Winding Number Algorithm.

Nalezený polygon obsahující zadaný bod q graficky zvýrazněte vhodným způsobem (např. vyplněním, šrafováním, blikáním). Grafické rozhraní vytvořte s využitím frameworku QT.

Pro generování nekonvexních polygonů můžete navrhnout vlastní algoritmus či použít existující geografická data (např. mapa evropských států).

Polygony budou načítány z textového souboru ve Vámi zvoleném formátu. Pro datovou reprezentaci jednotlivých polygonů použijte špagetový model.

Bonusové úlohy

1. Ošetření singulárního případu u Winding Number Algorithm: bod leží na hraně polygonu
2. Ošetření singulárního případu u obou algoritmů: bod je totožný s vrcholem jednoho či více polygonů
3. Zvýraznění všech polygonů pro oba výše uvedené singulární případy
4. Algoritmus pro automatické generování nekonvexních polygonů

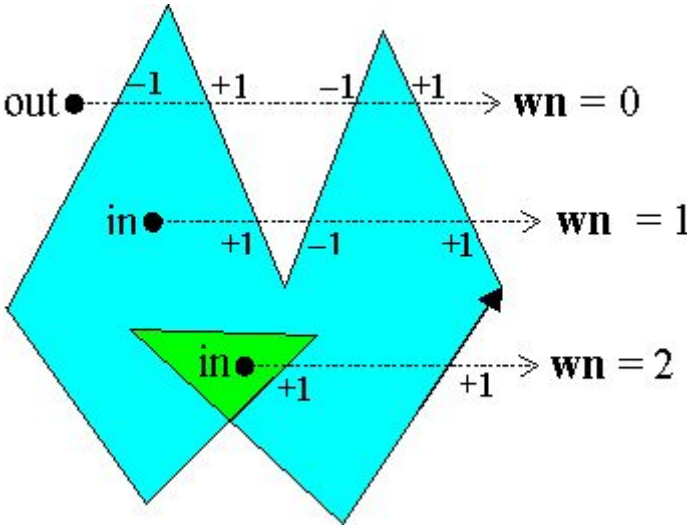
Vytvořený program obsahuje řešení 1. 2. a 3. bonusové úlohy.

Zvolené algoritmy

Ray Crossing Algorithm (s posunem těžiště polygonu)

Algoritmus je založen na hledání průsečíků polopřímky *p* s hranami polygonu *P*. Polopřímka *p* je vedena bodem *Q* - tvoří tzv. paprsek (Ray). Počet průsečíků následně určuje výsledek algoritmu.

Počet průsečíků polopřímky *q* s hranou polygonu *P*….. M
M ….. sudé - > bod *q* leží uvnitř polygonu *P*
M ….. liché - > bod *q* neleží uvnitř polygonu *P*

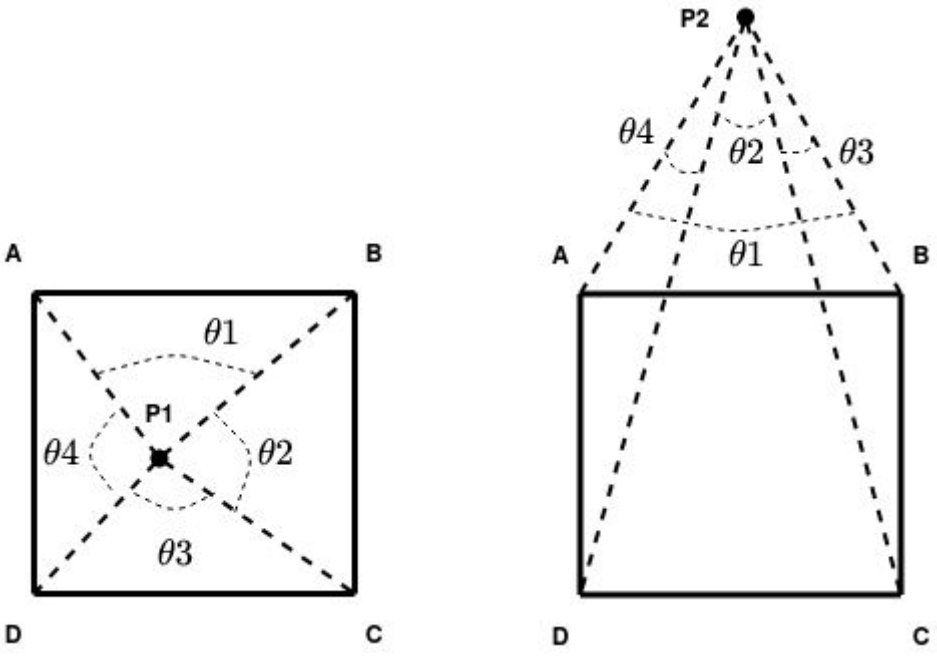


Princip Ray Crossing Algorithm [1]

https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk3.pdf slide 11

Winding number

Základ algoritmu spočívá v určení tzv. Winding number. Jedná se o součet všech rotací ω proti směru hodinových ručiček od bodu q k vrcholům polygonu. V případě, kdy je výsledný součet rotací roven 2π, zvolený bod q se nachází uvnitř polygonu. Naopak pokud není výsledný součet rotací roven 2π, bod q se nachází mimo polygon.



Princip Winding number [2]

https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk3.pdf slide 8

Problematické situace a jejich rozbor

Jako problematickou situaci považujeme tzv. singularity. To jsou případy, kdy se daný bod nachází buď na jednom z vrcholů polygonu a nebo na linii.

a) Bod ležící na vrcholu polygonu

- Tento případ lze vyřešit elegantním způsobem a to tak, že porovnáme daný bod *q* se seznamen souřadnic polygonu (polygonů) a zjistíme, zda se bod *q* nachází v těsné blízkosti (toleranci) souřadnic polygonu. V případě, že délka bodu q a alespoň jednoho z vrcholů polygonu je menší, než stanovená tolerance, algoritmus dá této situaci za pravdu.

b. Bod ležící na linii polygonu

- K řešení tohoto případu je vhodné využít orientaci bodu vůči přímce, kterou ve skriptu definuje funkce *getPointLinePosition*. Tato metoda určuje pozici bodu q vůči zadané hraně polygonu P. Vrací hodnoty 1 (bod leží v levé polorovině), 0 (bod leží v pravé polorovině) a -1 (bod leží na hraně). Ošetření případu, že bod leží na hraně, bylo vytvořeno na základě podmínky sestrojení trojúhelníku. Trojúhelník lze sestrojit tehdy pokud součet délek dvou stran je větší než délka třetí strany. Pokud podmínka neplatí, body leží v rovině. Jelikož uživatel kliká myší a né vždy klikne na hranu byla zvolena tolerance 0,2. Pokud podmínku, že leží na hraně nesplňuje, je následně vypočítán determinant vektorů p1p2 a qp1. Pokud je determinant větší než 0 funkce vrací hodnotu 1, pokud je záporný, vrací hodnotu 0.

Vstupní data

Vstupní data tvoří textový soubor ve formátu .txt. Soubor tvoří jednotlivé vrcholy polygonu - řádce v pořadí *id vrcholu*, *souřadnice x* a *souřadnice y - oddělené mezerou.*

Ukázka dat:

```
1 22 11          …… 1. vrchol polygonu
2 11 11
3 11 22
4 22 22
1 22 11          …… 1. vrchol dalšího polygonu
2 33 11
3 33 33
atd.
```

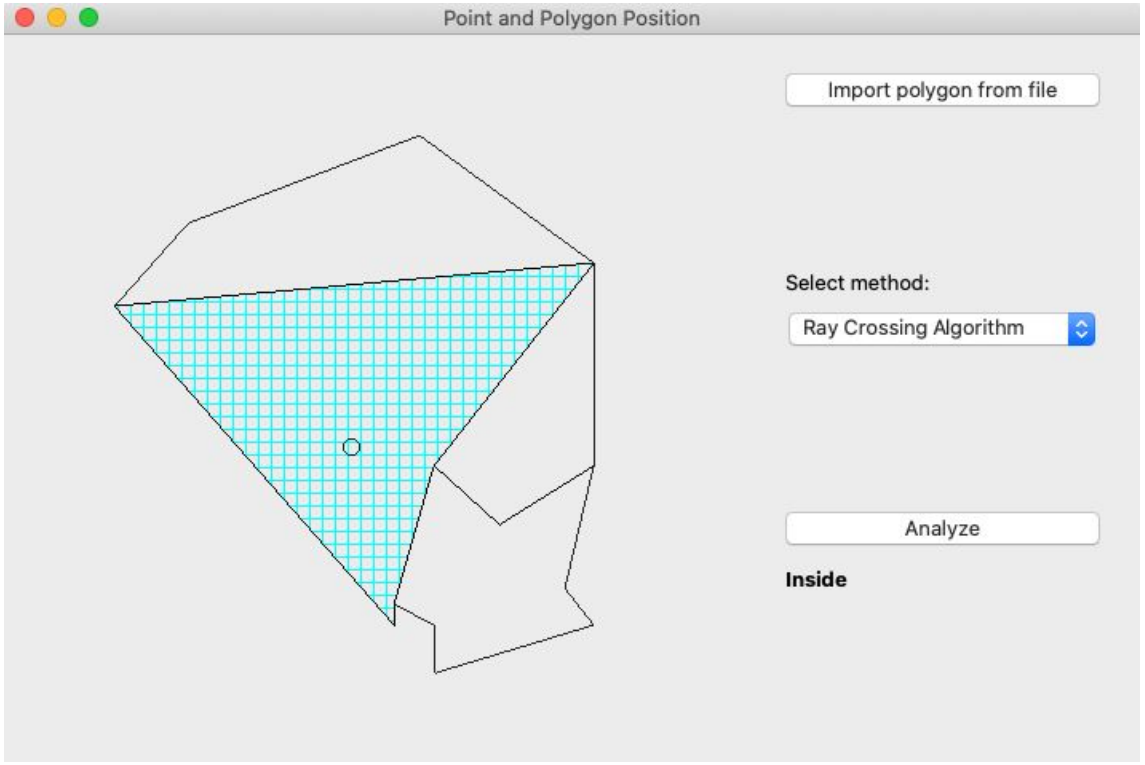
- polygon.txt
- rectangles.txt

Výstupní data

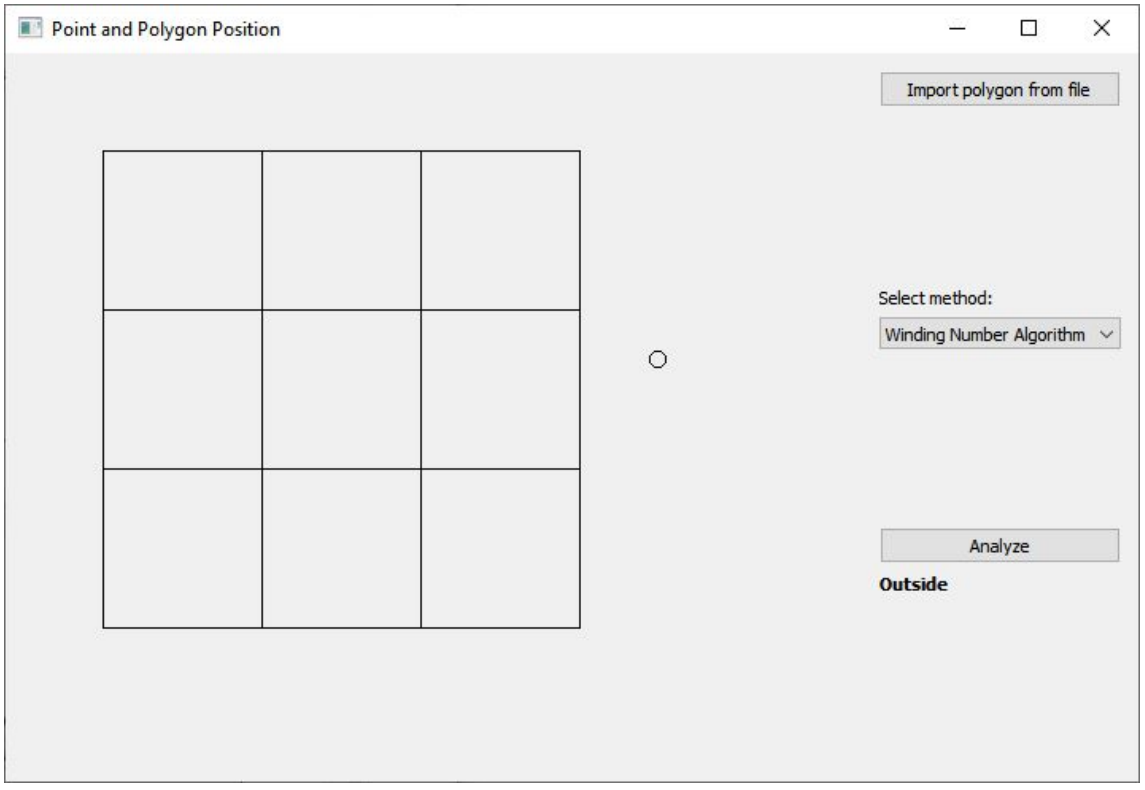
Výstupem je grafická aplikace, která slovně určí polohu bodu *q* vůči polygonu *P.* Aplikace načte textový soubor obsahující vrcholy zadaných polygonů. Následně umožní uživateli zadat bod *q*, vybrat metodu určení polohy a výsledek slovně i graficky zobrazí.

Vytvořená aplikace

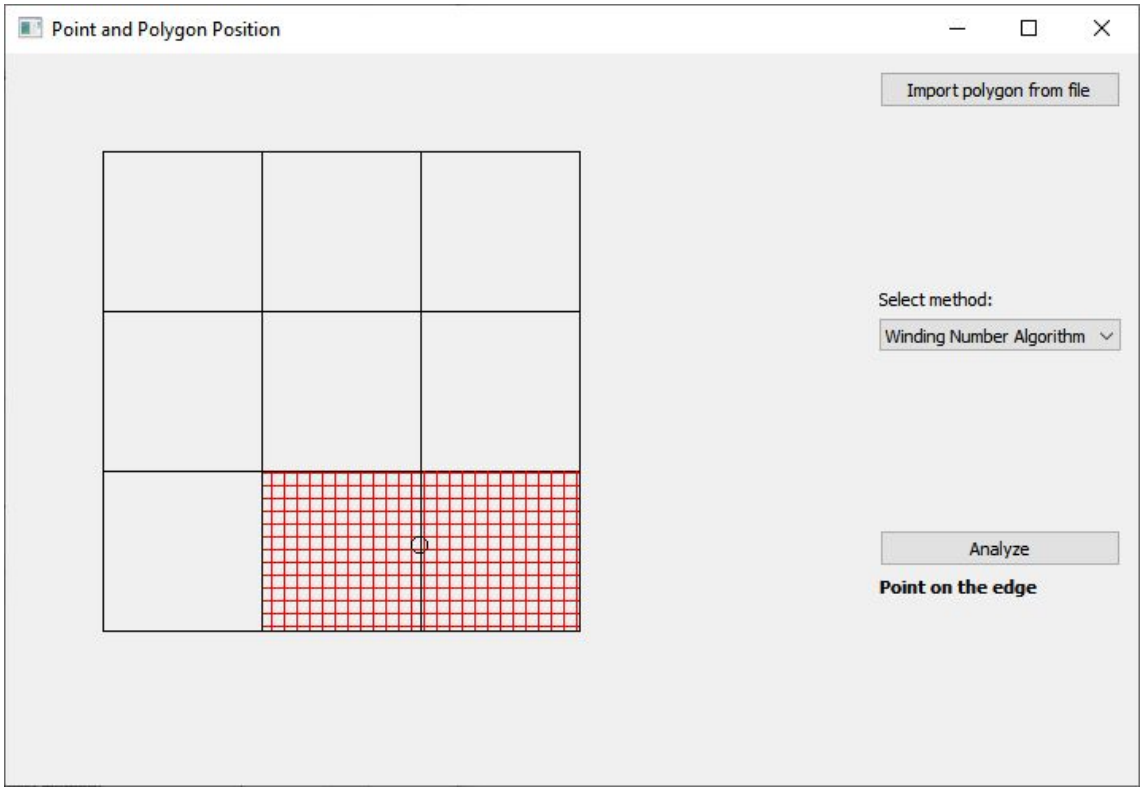
Následující snímky vytvořené aplikace zobrazují řešení daných situací



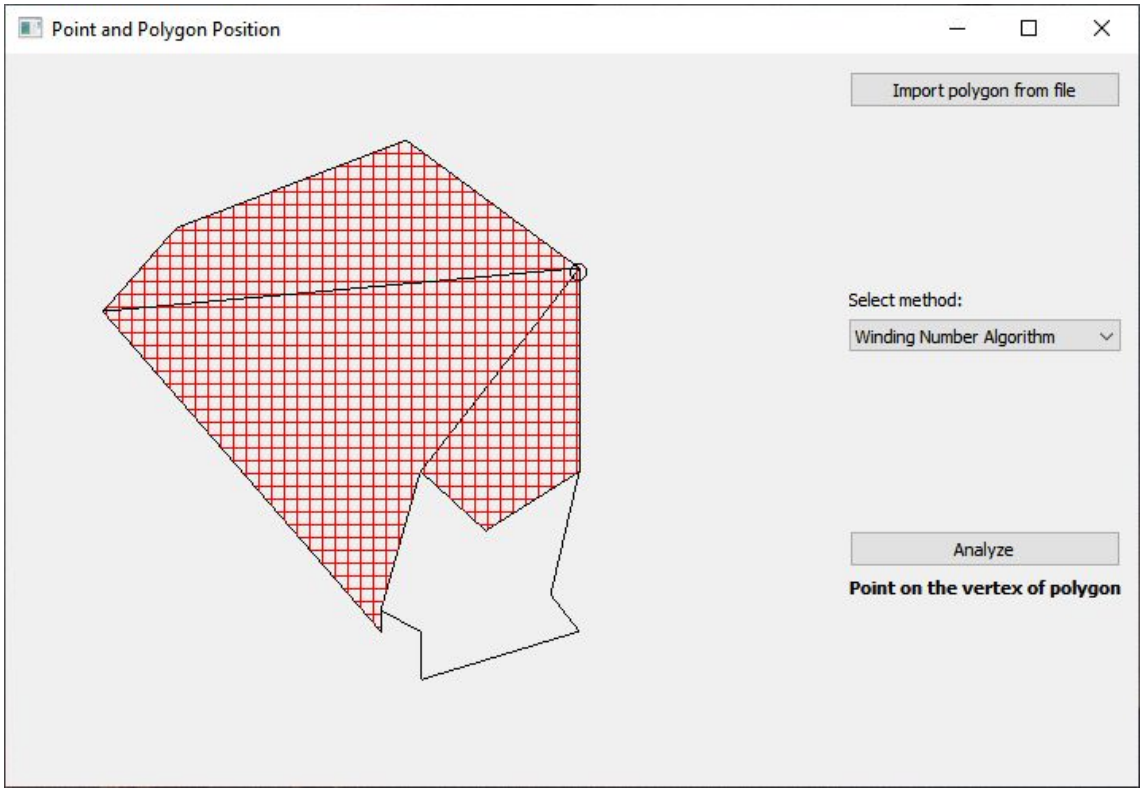
Bod leží uvnitř polygonu



Bod leží mimo polygon



Bod leží na hraně polygonu



Bod leží na hraně polygonu

Dokumentace

Třídy, datové položky a metody

Aplikace obsahuje tři třídy - Algorithms, Draw a Widget. Každá třída je zastoupena hlavičkovým souborem a zdrojovým souborem. V hlavičkových souborech jsou definovány společně se třídou její proměnné a metody.

- **Třída Algorithms**

  Třída Algorithms obsahuje celkem čtyři metody, které jsou použity pro vyřešení zadaného problému. Datovými typy metod byly QPointF a QPolygonF, oba s plovoucí desetinnou čárkou.

  `int getPointLinePosition(QPointF &q, QPointF &p1, QPointF &p2);`

  Tato metoda určuje pozici bodu *q* vůči zadané hraně polygonu *P*. Vrací hodnoty 1 (bod leží v levé polorovině), 0 (bod leží v pravé polorovině) a -1 (bod leží na hraně). Ošetření případu, že bod leží na hraně, bylo vytvořeno na základě podmínky sestrojení trojúhelníku. Trojúhelník lze sestrojit tehdy pokud součet délek dvou stran je větší než délka třetí strany. Pokud podmínka neplatí, body leží v rovině. Jelikož uživatel kliká myší a né vždy klikne na hranu byla zvolena tolerance 0,2. Pokud podmínku, že leží na hraně nesplňuje, je následně vypočítán determinant vektorů p1p2 a qp1. Pokud je determinant větší než 0 funkce vrací hodnotu 1, pokud je záporný, vrací hodnotu 0.

  `double getAngle(QPointF &p1,QPointF &p2,QPointF &p3,QPointF &p4);`

  Metoda vrací hodnotu úhlu mezi dvěma vektory.

  `int getPositionWinding(QPointF &q, std::vector<QPointF> &pol);`

  Tato metoda určuje polohu bodu *q* vůči zadanému polygonu *P* metodou Winding Number Algorithm. Pokud je výstupem 1, bod *q* leží uvnitř polygonu *P,* pokud -1,bod *q* leží na hraně polygonu *P* a pokud se výsledek rovná 0, bod *q* leží mimo polygon *P*. Metoda k výpočtům využívá metody *getAngle* a *getPointLinePossition,* které jsou deklarovány ve stejné třídě.

  `int getPositionRay(QPointF &q, std::vector<QPointF> &pol);`

  Metoda *getPositionRay,* podobně jako metoda *getPositionWinding,* vrací hodnotu 1 (bod leží uvnitř polygonu) a 0 (bod leží mimo polygon). Výsledné hodnoty počítá na základě počtu průsečíků polopřímky s počátkem v bodě *q* a jednotlivými hranami polygonu *P*.

- **Třída Draw**

  `void mousePressEvent(QMouseEvent *e);`

  Tato metoda ukládá souřadnice bodu *q*, které uživatel zadá kliknutím myši do kreslícího pole.

  `void paintEvent(QPaintEvent *e);`

  Touto metodou se vykreslují načtené polygony, bod *q* a vybarvení polygonů při situaci bod leží na hraně (červeně) a bod leží uvnitř polygonu (tirkisově). Zvýraznění polygonů bylo vytvořeno pomocí funkce setStyle() z knihovny <QWidget> [3]

  `QPoint & getPoint(){return q;}`

  Metoda (getter) vrací souřadnice bodu *q*.

`void loadPolygon(std::string &path);`
Metoda, která načte vrcholy polygonů z textového souboru.

`void setPolygon(std::vector<int> res){result=res;}`
Metoda (setter), která ukládá řešení pro jednotlivé vrcholy polygonu.

`std::vector<QPolygonF> getPolygons(){return polygons;}`
Metoda (getter), která ukládá načtené vrcholy polygonů do proměnné typu vector<QPolygonF>.

- **Třída Widget**
  void on_pushButton_clicked**();**
  Po kliknutí na tlačítko *pushButton* (Import polygon from file) metoda vrací hodnotu cesty, kde se textový soubor nachází.

  void on_pushButton_2_clicked**();**
  Po kliknutí na tlačítko *pushButton_2* (Analyze) metoda analyzuje polohu bodu *q* vůči polygonu P, podle algoritmu, který uživatel vybral z nabídky. Výstupem metody je slovní odpověď analýzy, která je zobrazena v textovém poli *label.*

Závěr

Výstupem této úlohy je aplikace, která umí analyzovat polohu bodu vůči polygonu. Aplikace načte textový soubor obsahující souřadnice vrcholu a vykreslí polygon(y). V kreslícím okně může uživatel zadat analyzovaný bod *q*. Následně po výběru metody (Ray Crossing Algorithm nebo Winding Number Algorithm) a zmáčknutí tlačítka Analyze program slovně vyjádří výsledek analýzy. Výsledkem analýzy jsou situace, kdy uživatel klikl do polygonu, mimo polygonu nebo na vrchol polygonu. Algoritmus Winding Number zvládne vyřešit i situaci, kdy uživatel klikne na hranu polygonu. V rámci úlohy nebyla z časových důvodů řešena poslední bonusová část.

V Praze dne 22.10.2020

Bc. Michal Zíma,
Bc. Tomáš Lauwereys

Seznam literatury a zdrojů

[1] Obrázek - princip Ray Crossing Algorithm
https://www.researchgate.net/figure/Schematic-illustration-of-the-Ray-Crossings-algorithm_fig4_267197000

[2] Obrázek - princip Winding number
https://towardsdatascience.com/is-the-point-inside-the-polygon-574b86472119

[3] Funkce setStyle()
https://doc.qt.io/archives/qtjambi-4.5.2_01/com/trolltech/qt/core/Qt.BrushStyle.html