ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA STAVEBNÍ, OBOR GEODÉZIE A KARTOGRAFIE

KATEDRA GEOMATIKY

název předmětu

ALGORITMY V DIGITÁLNÍ KARTOGRAFII A GIS

číslo úlohy	název úlohy				
2	Konvexní obálky a jejich konstrukce				
školní rok	**studijní skup.**	**číslo zadání**	**zpracoval**	**datum**	**klasifikace**
2019/2020	60		Tomáš Bouček, Jan Šikola	3.12. 2019	

Obsah

1. Zadání

Vstup: množina $P = \{p_1, \ldots, p_n\}$, $p_i = [x_i, y_i]$.
Výstup: konvexní obálka $H(P)$.

Nad množinou *P* implementujte následující algoritmy pro konstrukci *H(P)*.

- Jarvis Scan,
- Quick Hull,
- Sweep Line.

Vstupní množiny bodů včetně vygenerovaných konvexních obálek vhodně vizualizujte. Pro množiny $n \in \langle 1000, 1000000 \rangle$ vytvořte grafy ilustrující doby běhu algoritmů pro zvolená *n*. Měření proveďte pro různé typy vstupních množin (náhodná množina, rastr, body na kružnici) opakovaně (10x) a různá *n* (nejméně 10 množin) s uvedením rozptylu. Naměřené údaje uspořádejte do přehledných tabulek.

Zamyslete se nad problematikou možných singularit pro různé typy vstupních množin a možnými optimalizacemi. Zhodnoťte dosažené výsledky. Rozhodněte, která z těchto metod je s ohledem na časovou složitost a typ vstupní množiny *P* nejvhodnější.

2. Údaje o bonusových úlohách

V rámci bonusových úloh byla řešena konstrukce konvexní obálky metodou *Graham Scan*. Dále pak konstrukce striktně konvexních obálek pro všechny uvedené algoritmy a algoritmus pro automatické generování množin bodů různých tvarů.

3. Popis a rozbor problému

Cílem úlohy bylo vytvořit aplikaci pomocí Qt Creatoru, která by na zadané množině bodů vykreslila konvexní obálku pomocí jednoho ze zvolených algoritmů. Na množinách tvaru grid, kruh a náhodné body byla dále testována časová náročnost jednotlivých algoritmů

4. Popis algoritmů

4.1 Jarvis Scan

Prvním krokem je nalezení bodu s minimální ypsilonovou souřadnicí, tzv. pivota. Následně se postupně vyhledávají body s maximálním úhlem od posledních dvou bodů konvexní obálky. Bod s maximálním úhlem se přidá do konvexní obálky. Algoritmus běží, dokud se poslední a první bod konvexní obálky neshodují.

Výpočet úhlu ω:

$$\vec{u} = \left|p_{j-1}, p_j\right| \ ,$$

$$\vec{v} = \left|p_j, p_i\right| \ ,$$

$$\omega = \left|\operatorname{acos}\left(\frac{u * v}{|u| \cdot |v|}\right) \quad .\right|$$

kde

p_{j-1}, p_j jsou poslední dva body konvexní obálky,
p_i je bod z množiny bodů.

Algoritmus Jarvis Scan má následující podobu:

1. Nalezení pivota *q*, *q* = min(y_i)
2. Přidání *q* do konvexní obálky *H*
3. Inicializace $p_{j-1} \in X,\ p_j = q, p_{j+1} = p_{j-1}$
4. Spuštění cyklu, dokud $p_{j+1} \neq q$
5. Nalezení bodu s maximálním úhlem, $p_{j+1} = \arg\max < (p_{j-1}, p_j, p_i)$
6. Přidání bodu p_{j+1} do konvexní obálky
7. Změna indexu, $p_{j-1} = p_j;\ p_j = p_{j+1}$

4.2 Quick Hull

Základem je setřídění bodů podle souřadnice X a rozdělení množiny bodů na horní a dolní část. V setříděných bodech se vezmou body s minimální a maximální souřadnicí X a zařadí se do konvexní obálky. Tyto dva body také tvoří přímku, která množinu dělí na dvě části. Tyto dvě části se řeší separátně a na konci dojde k jejich sjednocení. Algoritmus se dělí na lokální a globální proceduru. Lokální procedura vyhledává nejvzdálenější bod od dané spojnice bodů, který se přidá do konvexní obálky. Tímto vzniknou dvě nové spojnice, od kterých se znovu vyhledává nejvzdálenější bod napravo od spojnice. Takto se postupuje, dokud se napravo od spojnic nenachází žádné body, tzn. byla vytvořena konvexní obálka.

Vzdálenost bodu o linie:

$$d(A,p) = \frac{|x_A(y_1 - y_2) + x_1(y_2 - y_A) + x_2(y_A - y_1)|}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}} \quad ,$$

kde

A je bod o souřadnicích $x_A\ a\ y_A$;
p je přímka daná body P_1 a P_2.

Algoritmus Quick Hull má následující podobu:

Lokální procedura (qh()):

1. Nalezení bodu $\bar{p}$ napravo od spojnice p_s, p_e s maximální vzdáleností.
2. Pokud $\bar{p} \neq \emptyset$
3. Zavolání funkce sama sebe; $qh(s, \bar{\imath}, S, H)$ // horní polovina
4. Přidání bodu $\bar{p}$ do konvexní obálky
5. Zavolání funkce sama sebe; $qh(\bar{\imath}, e, S, H)$ // dolní polovina

kde

S je množina bodů,
s je index počátečního bodu spojnice;
e je index koncového bodu spojnice,
$\bar{\imath}$ je index nejvzdálenějšího bodu od spojnice.

Globální procedura:

1. Inicializace: $H = \emptyset, S_U = \emptyset, S_L = \emptyset$
2. Setřídění bodů podle X, $q_1 = \min(x_i)\,, q_3 = \max(x_i)$
3. Přidání bodů q_1 a q_3 do S_U // do horní poloroviny
4. Přidání bodů q_1 a q_3 do S_L // do dolní poloroviny
5. For cyklus přes všechny body množiny
6. Pokud $p_i \in \sigma_l(q_1, q_3)$, přidání p_i do S_U // leží nalevo od spojnice
7. jinak přidání p_i do S_L
8. Přidání bodu q_3 do konvexní obálky
9. Rekurze; $qh(1{,}0, S_U, H)$
10. Přidání bodu q_1 do konvexní obálky
11. Rekurze; $qh(0{,}1, S_L, H)$

4.3 Sweep Line

Tento algoritmus využívá tzv. předchůdců a následníků bodů. Základem je setřídění bodů podle X a následné „projetí“ množiny bodů ve směru osy X. Tím se množina dělí na zpracovanou a nezpracovanou část. „Nastartování“ algoritmu je řešeno pomocí iniciálního trojúhelníku, případně iniciálního dvojúhelníku. Algoritmus se následně dělí na dvě iterativní fáze. V první fázi dochází k připojování nových bodů do již vytvořené obálky. Tím ale může dojít k porušení konvexity. V druhé fázi dochází k opravě a nekonvexní vrcholy jsou z množiny konvexní obálky odstraněny.

Algoritmus Sweep Line má následující podobu:

1. Setřídění množiny bodů podle X
2. Inicializace vektorů bodů předchůdců *p(m)* a následníků *n(m)*, kde *m* je počet všech bodů
3. Prvotní aproximace; n[0] = 1, n[1] = 0, p[0] = 1, p[1] = 0
4. For cyklus přes všechny body; i = 2
5. Pokud $y_i > y_{i-1}$
6. p[i] = i - 1; n[i] = n[i - 1]
7. jinak
8. n[i] = i – 1; p[i] = p[i – 1]
9. spojení předchůdce a následníka i; n[p[i]] = i; p[n[i]] = i
10. Dokud n[n[i]] ∈ $\sigma_R(i, n[i])$
11. p[n[n[i]]] = i; n[i] = n[n[i]]
12. Dokud p[p[i]] ∈ $\sigma_L(i, p[i])$
13. n[p[p[i]]] = i; p[i] = p[p[i]]

4.4 Graham Scan

Základem algoritmu je setřídění bodů podle úhlu, který svírají pivot q a rovnoběžka osy x procházející pivotem. Jako pivot se volí takový bod, který má minimální souřadnici *Y*. Konvexní obálka obsahuje pivota a bod s maximálním úhlem. Dále se postoupí na předposlední bod setříděné množiny bodů a bod se vloží do obálky. Postoupí se na další bod v pořadí a zkoumá se, jestli bod leží nalevo od spojnice posledních dvou bodů konvexní obálky. Pokud ano, bod se přidá do obálky, pokud ne, tak se poslední přidaný bod do konvexní obálky vyloučí a proces se opakuje.

Algoritmus Graham Scan má následující podobu:

1. Nalezení pivota q = $\min(y_i)$,
2. Setřídění množiny bodů podle úhlu ω, $\omega = < (p_i, q, x)$,
3. Pokud $\omega_k = \omega$, vymaž bod p_k, p_l bližší ke q,
4. Inicializuj j = 2, S = ∅,
5. q a p_1 přidat do CH,
6. Opakuj pro j < n:
7. Když p_l je vlevo od p_{t-1}, p_t,
8. vlož p_j do CH,
9. j = j + 1,
10. Jinak vyřaď poslední bod z CH.

5. Problematické situace a jejich rozbor + řešení generátorů

5.1 Striktně konvexní obálka

Mezi problémové situace se řadí kolineární body ležící v konvexní obálce přímo za sebou. Problém byl ošetřen tak, že se vzaly 2 sousední body konvexní obálky a následně bylo testováno, jestli následující bod neleží na přímce tvořenou prvními dvěma body. Pokud ano, bod byl z konvexní obálky vyřazen. Takto bylo učiněno pro každou dvojici bodů.

5.2 Generátor gridu

Nejprve byl vytvořen levý horní bod. Souřadnici *X* byla přidělena polovina z rozdílu šířky a výšky okna, souřadnici *Y* byla přidělena hodnota 10. Následně byly postupně tvořeny body po řádcích s rozestupem úměrným k celkovému počtu bodů a velikosti okna.

5.2 Generátor kružnice

Nejprve byl vytvořen bod *s* reprezentující střed kružnice. Umístěn byl doprostřed okna. Následně byl vypočten úhel φ = 2π/n, podle kterého se jednotlivé body budou natáčet. Poté se postupně vytvořili jednotlivé body kružnice s konstantním rozestupem, kde úhel tvořený spojnicemi dvou sousedních bodů se středem kružnice byl roven φ.

5.3 Generátor náhodných bodů

Náhodné body byly vygenerovány pomocí funkce *QRandomGenerator*. Jako maximální možné hodnoty byly zadány rozměry okna aplikace (canvasu).

5.4 Generátor elipsy

Elipsa byla generována stejně jako kružnice. Rozdílem je, že zatímco u kružnice se brala v potaz pouze výška okna (menší z rozměrů okna), tak v tomto případě se v potaz brala i šířka. Výška tak vstupuje do generování souřadnic *X* a šířka do souřadnic *Y*. V případě, že se stane, že je okno aplikace čtvercové, výsledkem generátoru bude kružnice.

5.5 Star-shape generátor

Tento generátor je založen na generování dvou kružnic. Jedna s větším poloměrem a druhá s menším. Body jsou generovány tak, aby žádná dvojice bodů, kde jeden leží na menší kružnici a druhý na větší, neležela v jedné přímce se středem kružnic.

5.6 Generátor čtverce

Generování čtverce je podobné generování gridu. Rozdílem je, že se generují body pouze na „obvodu“. Nejprve se vygenerují body na stranách rovnoběžných s osou *X*, poté na stranách rovnoběžných s osou *Y*.

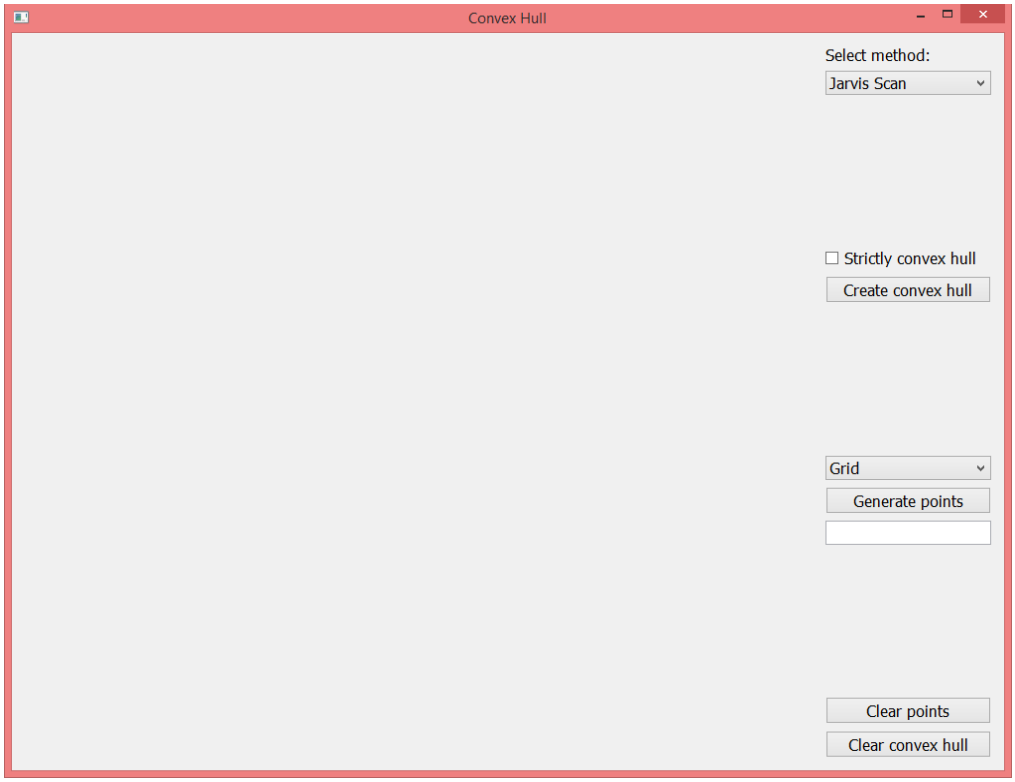
6. Vstupní data

Vstupní data jsou tvořena vkládáním uživatelem buď jako klik myši do plochy, anebo využitím generátoru množin bodů.

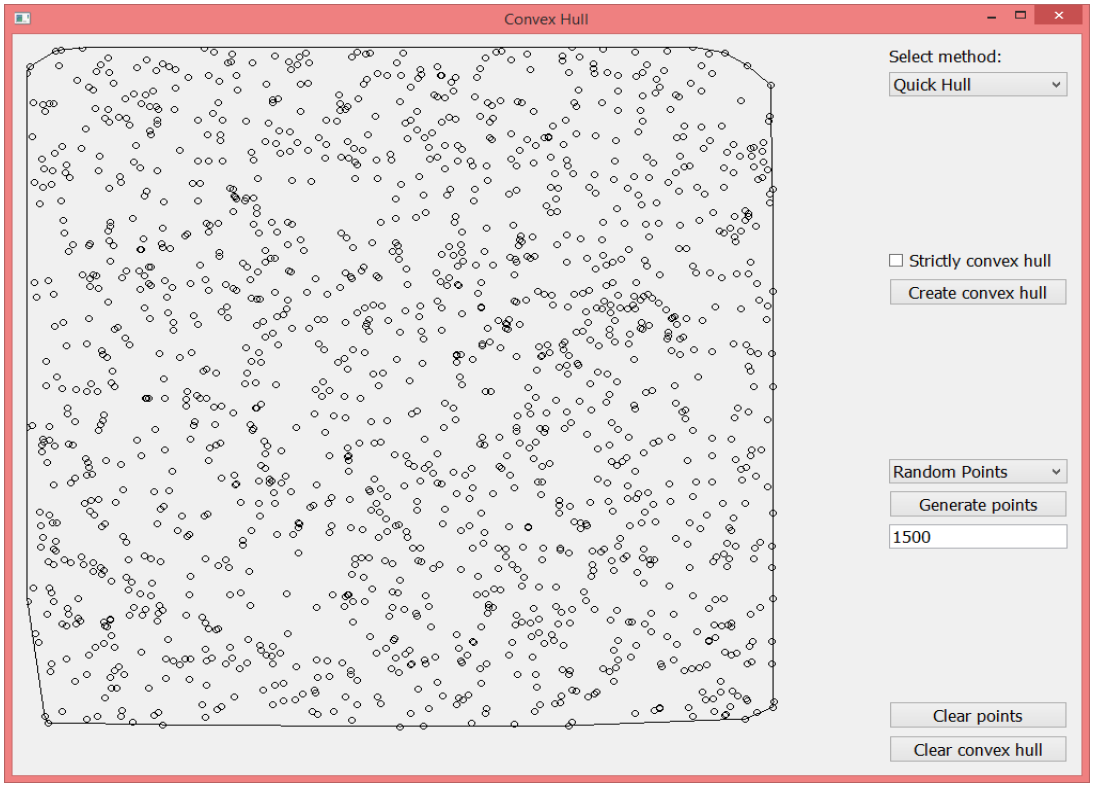
7. Výstupní data

Výstupem programu je grafické znázornění konvexní obálky.

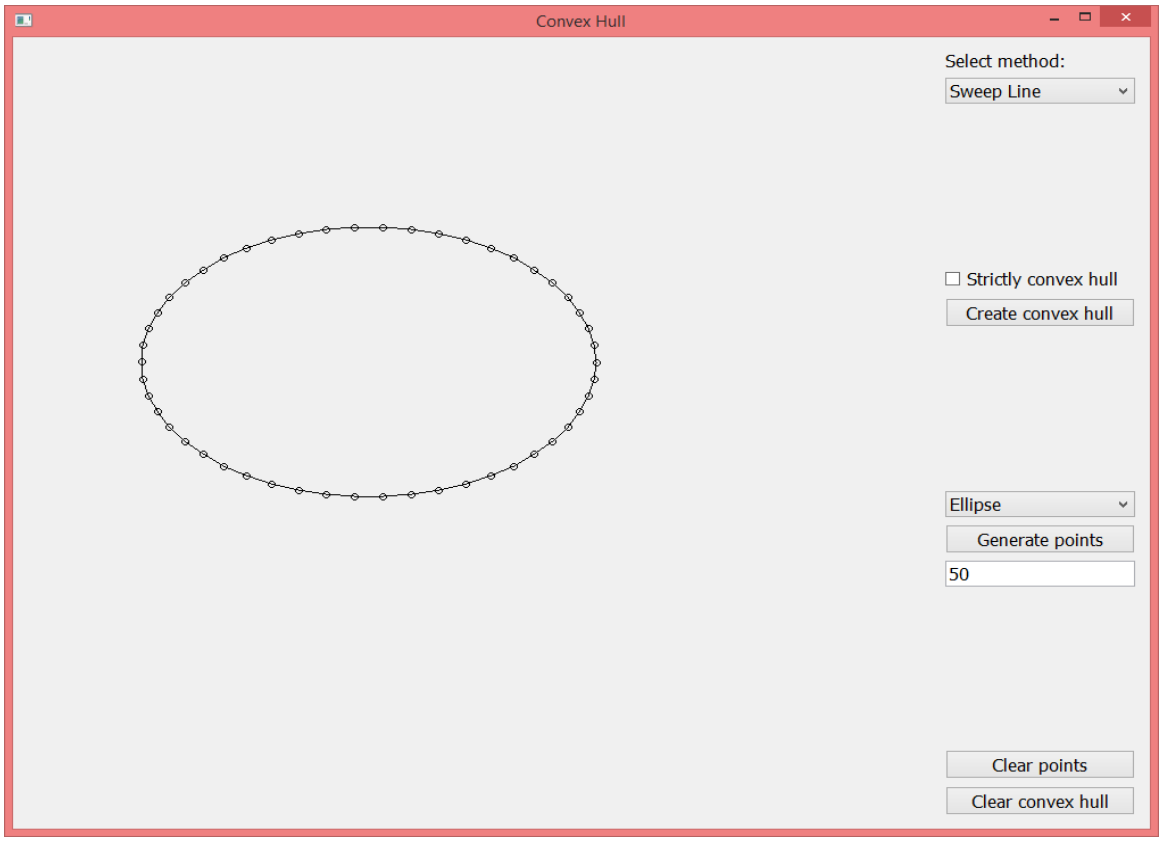
8. Obrázky vytvořené aplikace



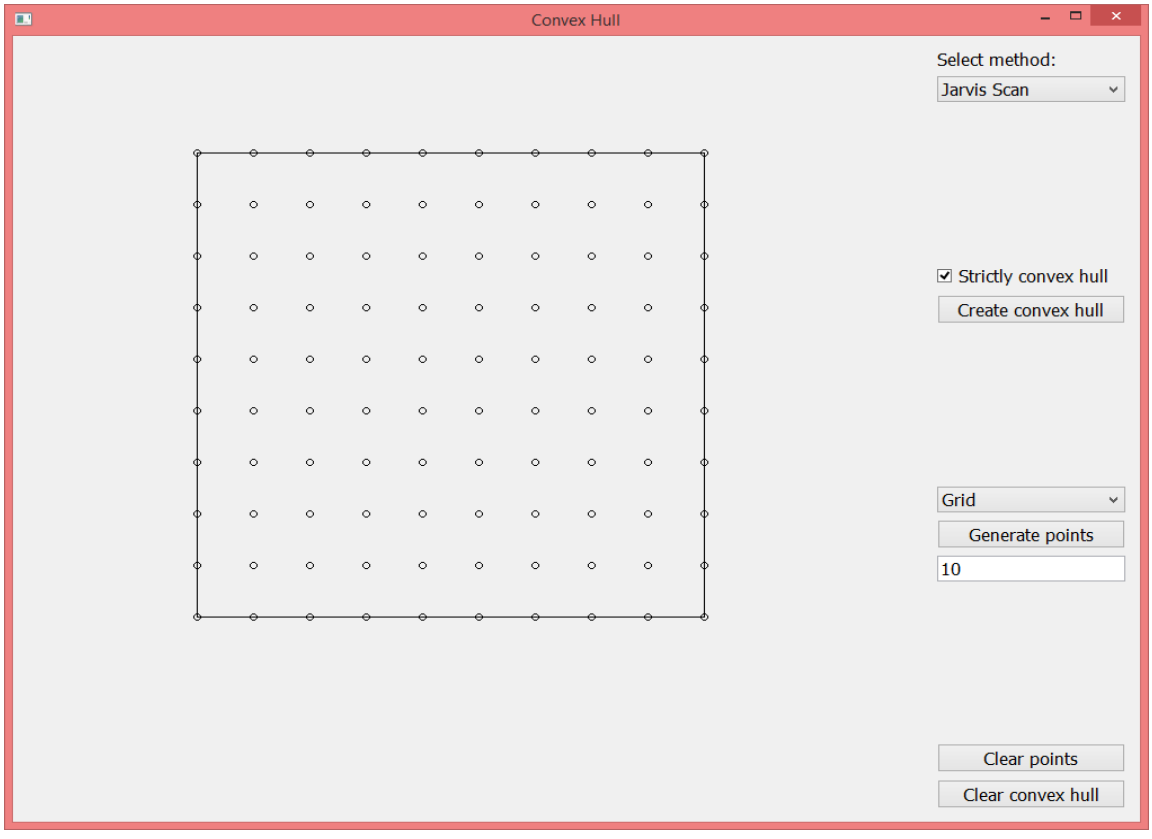
Obrázek 1: Ukázka aplikace po spuštění



Obrázek 2: Ukázka vygenerovaných 1500 náhodných bodů a jejich konvexní obálka



Obrázek 3: Ukázka vygenerované elipsy a její konvexní obálka



Obrázek 4: Ukázka vygenerovaného 10x10 gridu a jeho striktně konvexní obálka

9. Dokumentace

Třída Algorithms

Třída obsahuje funkce:

```
static double getPointLineDistance(QPoint &q, QPoint &p1, QPoint &p2);
-funkce počítá vzdálenost mezi bodem q a přímkou definovanou body p1 a p2.

static int getPointLinePosition(QPoint &q,QPoint &p1,QPoint &p2);
-funkce počítá orientaci bodu vůči přímce, jestli se nachází v pravé nebo
levé polorovině anebo jestli se bod nachází na přímce. Vstupní parametry
funkce jsou souřadnice bodu q a souřadnice koncových bodů hran polygonů.

static double getAngle2Vectors(QPoint &p1,QPoint &p2,QPoint &p3,QPoint &p4);
-funkce počítá úhel mezi dvěma vektory. Vstupní parametry funkce jsou
souřadnice bodů vektoru.

static double getPointToPointDistance(QPoint &p1, QPoint &p2);
-funkce počítá vzdálenost mezi body p1 a p2.

static QPolygon jarvisScan(QPolygon &points);
-funkce k výpočtu konvexní obálky algoritmem Jarvis Scan. Vstupem jí je
množina bodů points.

static QPolygon qHull(QPolygon &points);
-funkce k výpočtu konvexní obálky algoritmem Quick Hull. Vstupem jí je
množina bodů points.

static void qh(int s, int e, QPolygon &points, QPolygon &ch);
-funkce pro rekursivní proceduru pro algoritmus Quick Hull.

static QPolygon sweepLine(QPolygon &points);
-funkce k výpočtu konvexní obálky algoritmem Sweep Line. Vstupem jí je
množina bodů points.

static QPolygon grahamScan(QPolygon &points);
-funkce k výpočtu konvexní obálky algoritmem Graham Scan. Vstupem jí je
množina bodů points.

static QPolygon strictlyConvexHull(QPolygon &ch);
-funkce pro odstranění kolineárních bodů z konvexní obálky.
```

Třída Draw

Třída obsahuje funkce a datové položky:

```
private:
    QPolygon points;
-deklarace proměnné pro vygenerování bodů a práce s nimi.

    QPolygon ch;
-deklarace proměnné pro konvexní obálku.
```

```
public:

   void mousePressEvent(QMouseEvent *event);
-funkce pro odečet souřadnic sejmutého bodu.

   void paintEvent(QPaintEvent *e);
-funkce pro vykreslení bodů a konvexní obálky.

   void clearCH(){ch.clear(); repaint();}
-funkce pro vymazání konvexní obálky.

   void clearPoints(){points.clear(); repaint();}
-funkce pro vymazání bodů.

   QPolygon getPoints(){return points;}
-funkce pro získání souřadnic bodů.

   void setCH(QPolygon &hull){ch = hull;}
-funkce pro nastavení konvexní obálky.

   void setPoints(QPolygon Points){points = Points;}
-funkce pro nastavení bodů.

   QPolygon generateGrid(int n);
-funkce pro vygenerování gridu.

   QPolygon generateCircle(int n);
-funkce pro vygenerování kružnice.

   QPolygon generateRandomPoints(int n);
-funkce pro vygenerování náhodných bodů.

   QPolygon generateEllipse(int n);
-funkce pro vygenerování elipsy.

   QPolygon generateStarShape(int n);
-funkce pro vygenerování „star shape".

   QPolygon generateSquare(int n);
-funkce pro vygenerování čtverce.

   QPolygon getCH(){return ch;}
-funkce pro získání konvexní obálky.
```

10. Výsledky generování

Jarvis Scan

Jarvis Scan (n = 1000)												
typ množiny	jednotlivá měření [ms]										průměr [ms]	rozptyl [ms]
Grid	3	3	3	3	3	3	3	3	3	3	3	0
Circle	10	10	10	10	10	10	10	10	10	10	10	0
Random	0	0	0	0	0	1	0	0	0	0	0,1	0,3
Jarvis Scan (n = 2000)												
typ množiny	jednotlivá měření [ms]										průměr [ms]	rozptyl [ms]
Grid	8	8	8	8	8	8	8	8	8	8	8	0
Circle	24	24	24	24	24	24	24	24	24	24	24	0
Random	2	2	2	1	1	2	2	2	2	1	1,7	0,46
Jarvis Scan (n = 5000)												
typ množiny	jednotlivá měření [ms]										průměr [ms]	rozptyl [ms]
Grid	33	33	33	33	33	33	33	33	33	33	33	0
Circle	72	72	83	76	72	72	72	72	73	72	73,6	3,352610923
Random	6	5	6	5	5	5	5	5	5	7	5,4	0,66
Jarvis Scan (n = 10000)												
typ množiny	jednotlivá měření [ms]										průměr [ms]	rozptyl [ms]
Grid	92	92	92	92	93	92	93	93	92	93	92,4	0,49
Circle	151	155	156	153	151	151	151	152	153	151	152,4	1,74
Random	13	19	13	13	14	18	13	12	14	13	14,2	2,23
Jarvis Scan (n = 50000)												
typ množiny	jednotlivá měření [ms]										průměr [ms]	rozptyl [ms]
Grid	989	971	975	990	982	987	973	982	982	983	981,4	6,22
Circle	863	860	864	873	859	858	863	858	861	865	862,4	4,25
Random	199	157	252	202	211	184	170	186	220	192	197,3	25,34
Jarvis Scan (n = 100000)												
typ množiny	jednotlivá měření [ms]										průměr [ms]	rozptyl [ms]
Grid	2973	3000	2985	2990	2970	2984	2982	2983	2986	2980	2983,3	7,94
Circle	1779	1752	1752	1779	1759	1776	1750	1762	1751	1751	1761,1	11,67
Random	788	717	683	827	814	797	801	803	756	761	774,7	43,27
Jarvis Scan (n = 250000)												
typ množiny	jednotlivá měření [ms]										průměr [ms]	rozptyl [ms]
Grid	10955	10987	10846	10889	10881	10882	10871	10876	10860	10877	10892,4	41,57
Circle	4374	4425	4376	4373	4370	4369	4380	4377	4368	4414	4382,6	18,94
Random	3967	2721	5652	4134	4222	4292	4256	3526	4120	4639	4152,9	706,94
Jarvis Scan (n = 500000)												
typ množiny	jednotlivá měření [ms]										průměr [ms]	rozptyl [ms]
Grid	31032	31032	30972	31093	31010	30984	30996	31007	31003	31064	31019,3	35,05
Circle	8875	8910	8814	8852	8884	8916	8775	8847	8897	8901	8867,1	43,08
Random	16161	15852	15981	15272	16080	15801	16010	15911	15494	15531	15809,3	272,86

Jarvis Scan (n = 750000)												
typ množiny	jednotlivá měření [ms]										průměr [ms]	rozptyl [ms]
Grid	56926	56997	57010	57132	57024	57104	57274	56708	57093	56796	57006,4	156,40
Circle	13330	13351	13354	13227	13188	13371	13300	13275	13324	13235	13295,5	58,78
Random	33153	31808	32486	32251	30953	32282	33524	31754	33054	33056	32432,1	747,21
Jarvis Scan (n = 1000000)												
typ množiny	jednotlivá měření [ms]										průměr [ms]	rozptyl [ms]
Grid	88134	87755	87216	86901	87020	87463	87555	87117	87448	87973	87458,2	386,90
Circle	17693	17827	17707	17843	17679	17690	17663	17838	17687	17750	17737,7	67,89
Random	52344	55149	52319	54337	55246	53616	55552	53228	54104	54833	54072,8	1111,68

Quick Hull

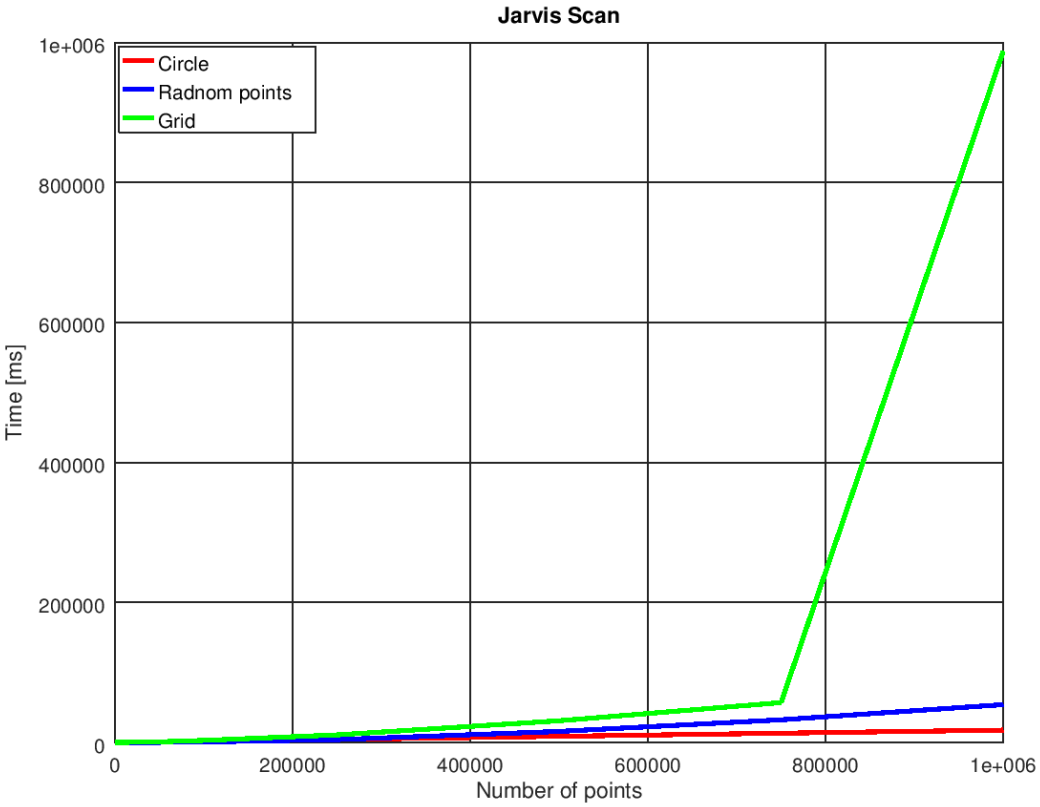
Quick Hull (n = 1000)												
typ množiny	jednotlivá měření [ms]										průměr [ms]	rozptyl [ms]
Grid	0	0	0	0	0	0	0	0	0	0	0	0
Circle	0	0	0	0	0	0	0	0	0	0	0	0
Random	0	0	0	0	0	0	0	0	0	0	0	0
Quick Hull (n = 2000)												
typ množiny	jednotlivá měření [ms]										průměr [ms]	rozptyl [ms]
Grid	0	0	0	0	0	0	0	0	0	0	0	0
Circle	1	1	1	2	1	1	2	1	1	1	1,2	0,4
Random	0	0	0	0	0	0	0	0	0	0	0	0
Quick Hull (n = 5000)												
typ množiny	jednotlivá měření [ms]										průměr [ms]	rozptyl [ms]
Grid	0	0	0	0	0	0	0	0	0	0	0	0
Circle	5	5	5	5	5	5	5	5	5	5	5	0
Random	0	0	1	1	1	0	0	1	1	1	0,6	0,49
Quick Hull (n = 10000)												
typ množiny	jednotlivá měření [ms]										průměr [ms]	rozptyl [ms]
Grid	0	0	0	0	0	0	0	0	0	0	0	0
Circle	9	9	9	9	9	9	9	9	9	9	9	0
Random	2	1	2	2	1	1	2	1	1	1	1,4	0,49
Quick Hull (n = 50000)												
typ množiny	jednotlivá měření [ms]										průměr [ms]	rozptyl [ms]
Grid	3	3	3	3	3	3	3	3	3	3	3	0
Circle	48	48	48	49	48	48	48	48	49	48	48,2	0,40
Random	8	8	8	8	8	8	9	9	7	8	8,1	0,54
Quick Hull (n = 100000)												
typ množiny	jednotlivá měření [ms]										průměr [ms]	rozptyl [ms]
Grid	6	6	6	6	6	6	6	6	6	6	6	0
Circle	122	122	123	122	122	122	122	124	120	122	122,1	0,94
Random	16	16	18	15	17	18	16	15	16	16	16,3	1,00
Quick Hull (n = 250000)												
typ množiny	jednotlivá měření [ms]										průměr [ms]	rozptyl [ms]
Grid	16	16	16	16	16	16	16	16	16	16	16	0
Circle	301	295	295	294	296	294	296	295	300	294	296	2,37
Random	39	40	38	39	41	39	36	40	43	40	39,5	1,75

Quick Hull (n = 500000)												
typ množiny	jednotlivá měření [ms]										průměr [ms]	rozptyl [ms]
Grid	33	33	33	33	33	33	37	36	33	33	33,7	1,42
Circle	590	592	596	596	595	591	598	593	598	598	594,7	2,87
Random	77	73	79	70	69	68	76	78	79	75	74,4	3,95
Quick Hull (n = 750000)												
typ množiny	jednotlivá měření [ms]										průměr [ms]	rozptyl [ms]
Grid	52	52	54	52	52	52	54	52	52	52	52,4	0,8
Circle	885	887	877	887	892	882	908	874	877	880	884,9	9,32
Random	110	119	111	109	104	115	112	114	113	125	113,2	5,44
Quick Hull (n = 1000000)												
typ množiny	jednotlivá měření [ms]										průměr [ms]	rozptyl [ms]
Grid	68	67	68	68	68	69	68	68	69	68	68,1	0,54
Circle	1200	1181	1194	1204	1185	1187	1189	1181	1201	1177	1189,9	8,94
Random	150	150	144	150	152	144	152	145	144	139	147	4,15

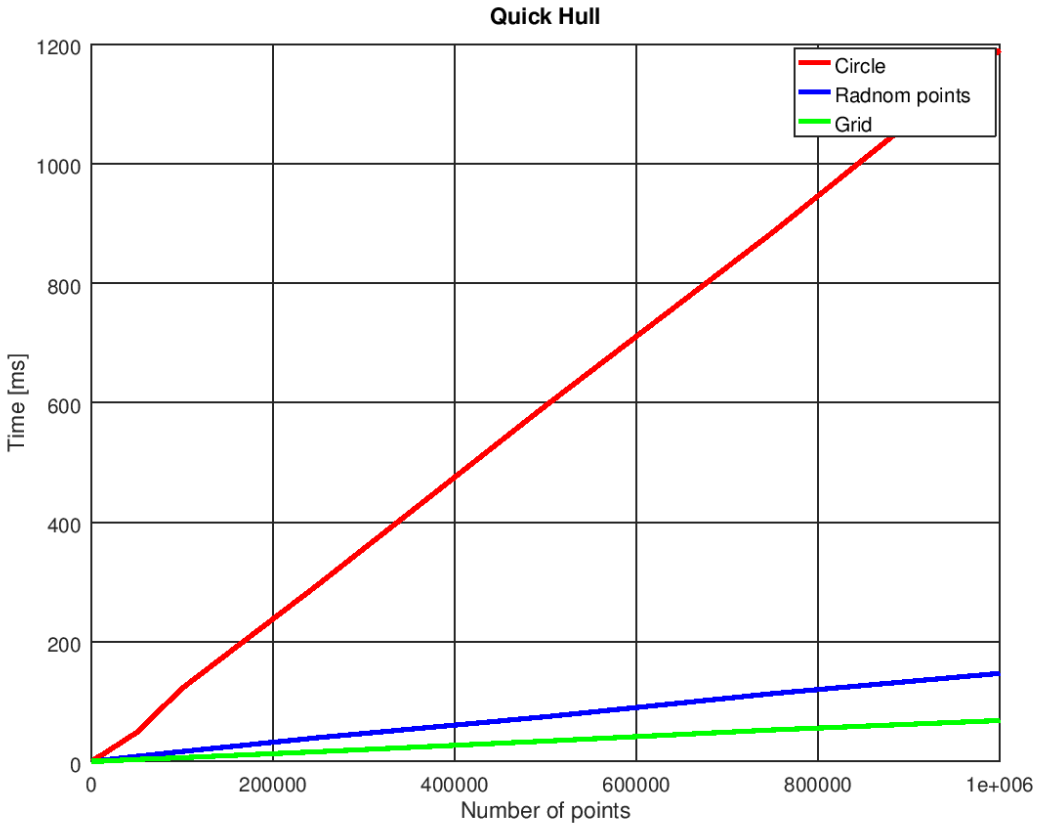
Sweep Line

Sweep Line (n = 1000)												
typ množiny	jednotlivá měření [ms]										průměr [ms]	rozptyl [ms]
Grid	0	0	0	0	0	0	0	0	0	0	0	0
Circle	0	0	0	0	0	0	0	0	0	0	0	0
Random	0	0	0	0	0	0	0	0	0	0	0	0
Sweep Line (n = 2000)												
typ množiny	jednotlivá měření [ms]										průměr [ms]	rozptyl [ms]
Grid	0	0	0	0	0	0	0	0	0	0	0	0
Circle	0	0	0	0	0	0	0	0	0	0	0	0
Random	0	0	0	0	0	0	0	0	0	0	0	0
Sweep Line (n = 5000)												
typ množiny	jednotlivá měření [ms]										průměr [ms]	rozptyl [ms]
Grid	0	0	0	0	0	0	0	0	0	0	0	0
Circle	0	0	0	0	0	0	0	0	0	0	0	0
Random	0	0	0	0	0	0	0	0	0	0	0	0
Sweep Line (n = 10000)												
typ množiny	jednotlivá měření [ms]										průměr [ms]	rozptyl [ms]
Grid	0	0	0	0	0	0	0	0	0	0	0	0
Circle	0	0	0	0	0	0	0	0	0	0	0	0
Random	0	0	0	0	0	0	0	0	0	0	0	0
Sweep Line (n = 50000)												
typ množiny	jednotlivá měření [ms]										průměr [ms]	rozptyl [ms]
Grid	4	4	4	4	4	4	4	4	4	4	4	0
Circle	3	3	3	3	3	3	3	3	3	3	3	0
Random	4	5	4	4	4	4	5	5	4	4	4,3	0,46

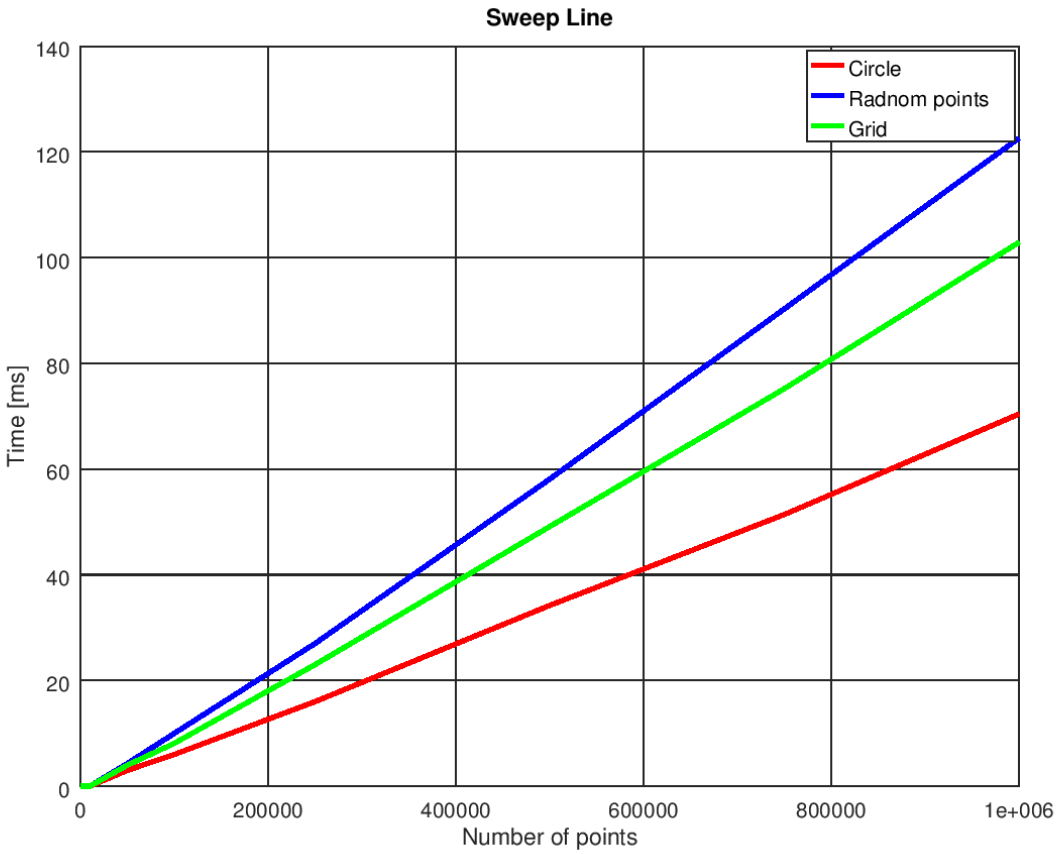
Sweep Line (n = 100000)												
typ množiny	jednotlivá měření [ms]										průměr [ms]	rozptyl [ms]
Grid	8	8	8	8	8	8	9	8	8	8	8,1	0,3
Circle	6	6	6	6	6	6	6	6	6	6	6	0
Random	10	10	10	10	10	10	10	10	10	10	10	0
Sweep Line (n = 250000)												
typ množiny	jednotlivá měření [ms]										průměr [ms]	rozptyl [ms]
Grid	23	23	23	23	23	23	23	23	23	23	23	0
Circle	16	16	16	16	16	16	16	16	16	16	16	0,00
Random	27	27	27	27	27	27	27	27	27	27	27	0,00
Sweep Line (n = 500000)												
typ množiny	jednotlivá měření [ms]										průměr [ms]	rozptyl [ms]
Grid	49	49	49	49	49	49	49	49	49	50	49,1	0,3
Circle	34	34	34	35	34	34	35	34	34	34	34,2	0,40
Random	58	58	58	58	59	58	58	58	58	58	58,1	0,30
Sweep Line (n = 750000)												
typ množiny	jednotlivá měření [ms]										průměr [ms]	rozptyl [ms]
Grid	75	74	75	76	75	76	75	76	75	75	75,2	0,60
Circle	51	52	51	52	51	51	51	51	53	51	51,4	0,66
Random	91	91	90	90	90	90	90	90	90	91	90,3	0,46
Sweep Line (n = 1000000)												
typ množiny	jednotlivá měření [ms]										průměr [ms]	rozptyl [ms]
Grid	103	102	104	103	103	103	103	103	102	103	102,9	0,54
Circle	68	69	70	76	72	70	69	69	72	69	70,4	2,24
Random	123	122	124	122	122	122	123	123	122	123	122,6	0,66



Obrázek 5: Jarvis Scan nad různými množinami bodů



Obrázek 6: Quick Hull nad různými množinami bodů



Obrázek 7: Sweep Line nad různými množinami bodů

11. Závěr

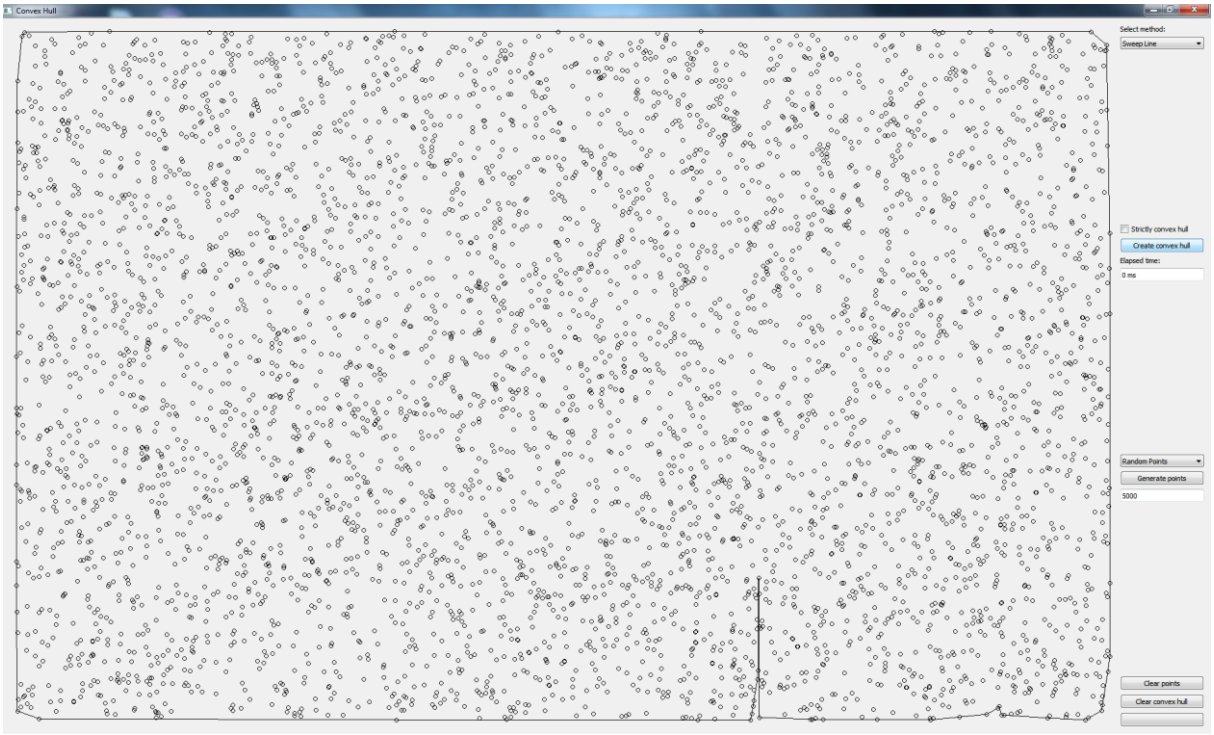
Byla vytvořena aplikace pro výpočet konvexní obálky nad náhodnou množinou bodů. Množinu bodů je možné získat klikáním myši do aplikace anebo využít generátor náhodných tvarů. Dále byl zkoumán čas, za který se provede určitý algoritmus nad náhodnou množinou *n* bodů.
Z výsledků je patrné, že nejvhodnějším algoritmem je *Sweep Line*, který i při velkém množství bodů zvládl vygenerovat konvexní obálku velmi rychle, nehledě na tvar množiny bodů. Nicméně se ukázalo, že algoritmus nejspíš obsahuje chybu, protože u vyššího počtu bodů se konvexní obálka vygenerovala chybně nad množinami *random points* a *circle* (viz. přílohy). Špatně si nevedl ani *Quick Hull*, který konvexní obálku nad „gridovou“ množinu bodů zvládl vygenerovat ještě rychleji než *Sweep Line*. Nad kruhovou množinou oproti *Sweep Line* trochu zaostával. *Jarvis* Scan si úspěšně poradil se všemi množinami, nicméně rychlost výpočtu při vyšším počtu bodů značně zaostává za předchozími algoritmy.

V přílohách jsou ukázány chybně vygenerovány konvexní obálky metodou *Sweep Line*.

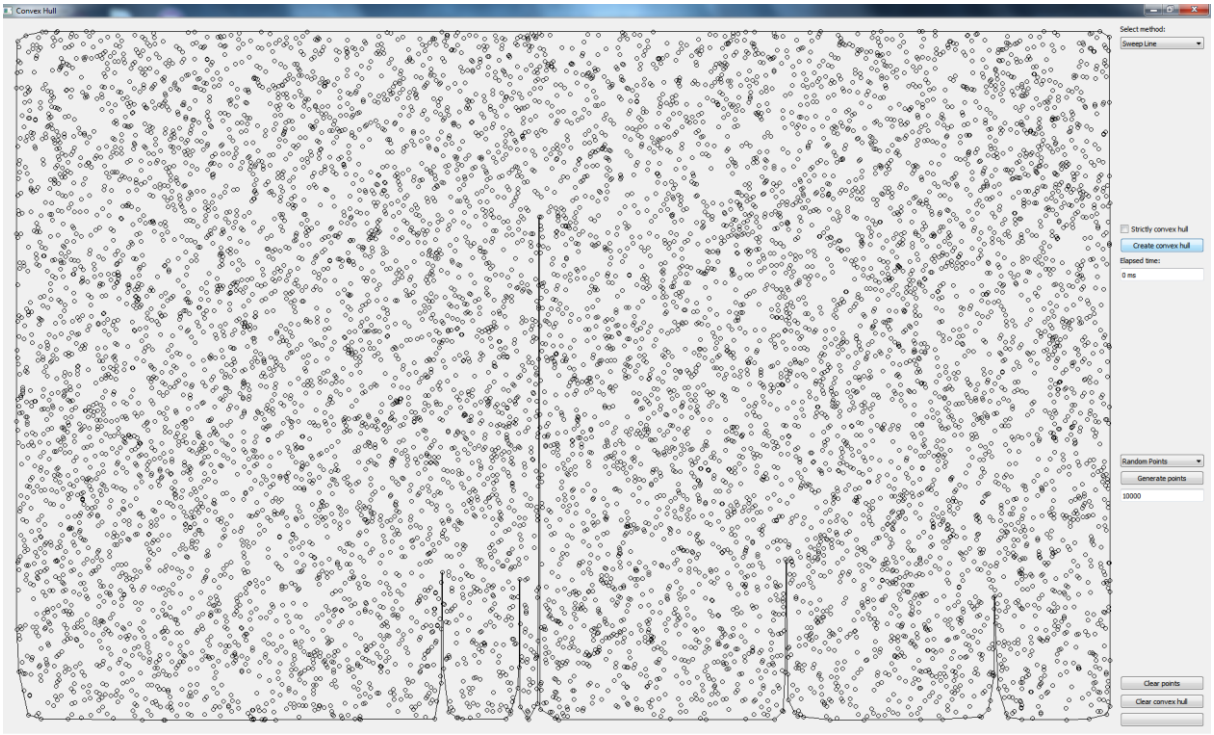
11.1 Náměty na vylepšení

- vyřešení chyby ve *Sweep Line Algoritmu*

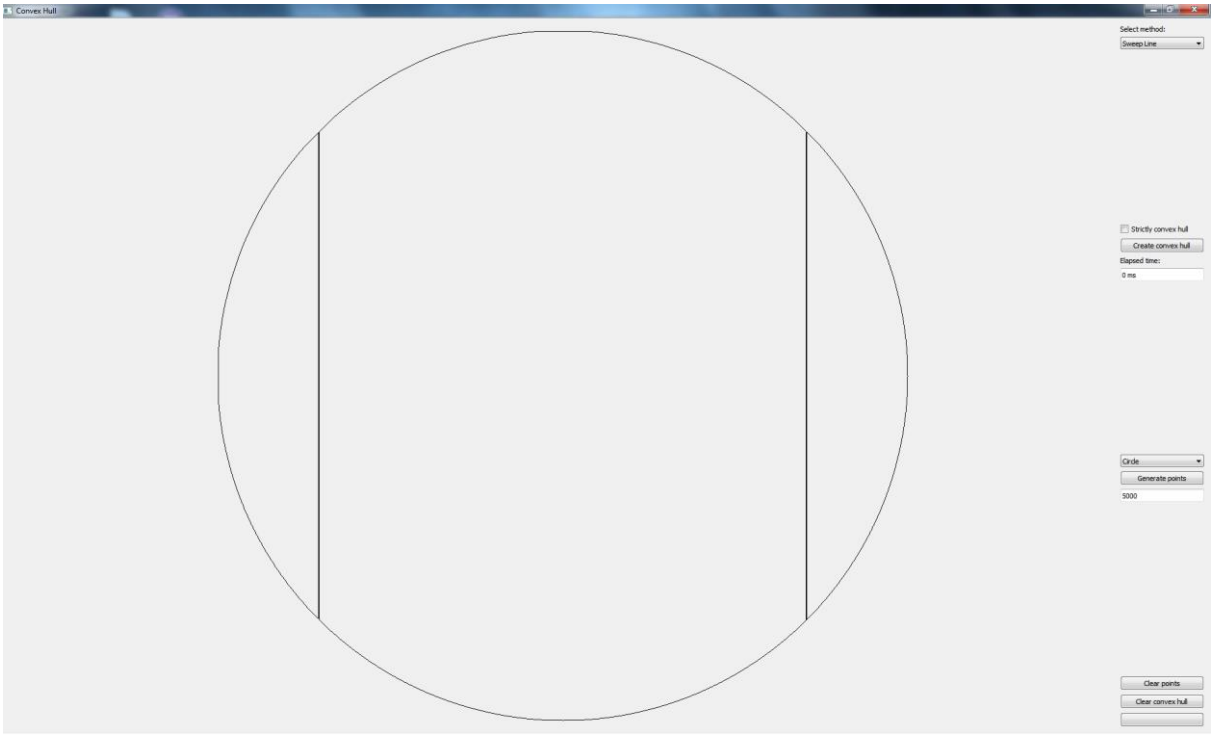
12. Přílohy



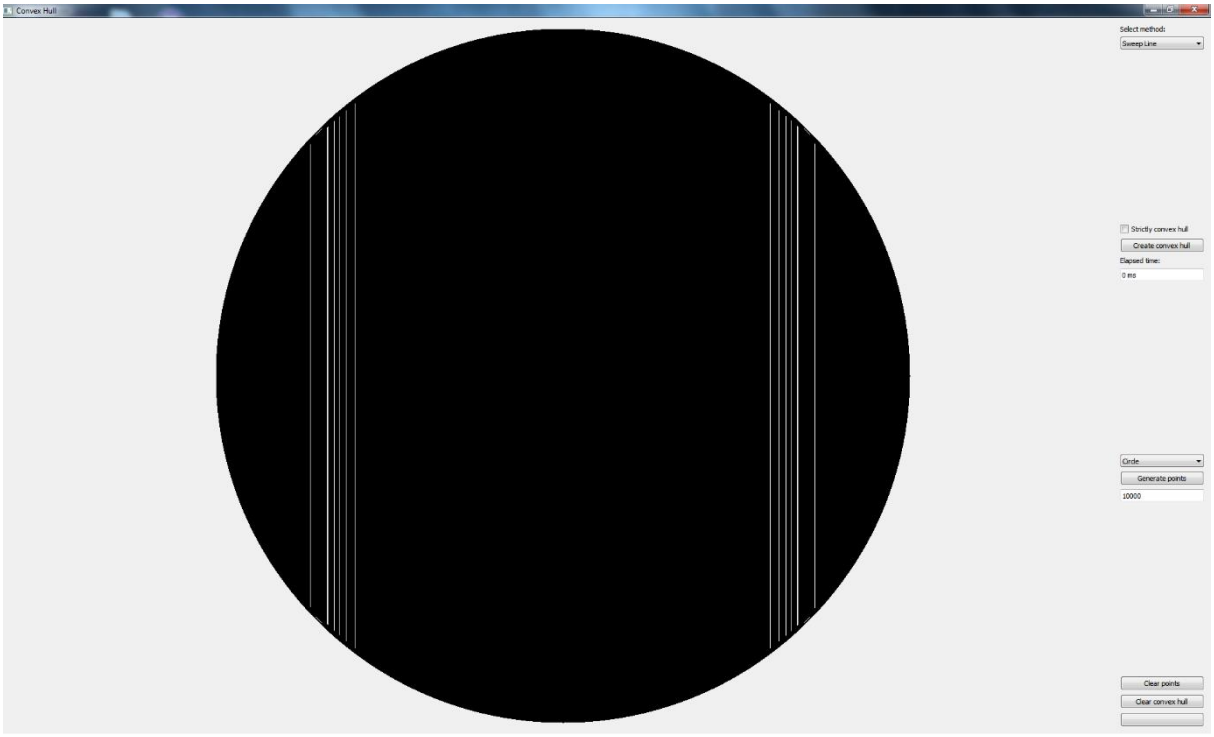
Obrázek 8: Chyba v konvexní obálce při použití Sweep Line nad množinou Random Points o 5000 bodech



Obrázek 9: Chyba v konvexní obálce při použití Sweep Line nad množinou Random Points o 10000 bodech



Obrázek 10: Chyba v konvexní obálce při použití Sweep Line nad množinou Circle o 5000 bodech



Obrázek 11: Chyba v konvexní obálce při použití Sweep Line nad množinou Circle o 10000 bodech