

**České vysoké učení technické v Praze  
Fakulta jaderná a fyzikálně inženýrská  
Katedra fyzikální elektroniky  
Informatická fyzika**

# **Geometrické algoritmy pro počítačovou grafiku**

**Semestrální práce**

Autor práce: **Bc. Milan Holec**  
Školní rok: **2009/2010**

# **Obsah**

<b>1</b>	<b>Úvod</b>	<b>4</b>
<b>2</b>	<b>Algoritmy</b>	<b>4</b>
<b>3</b>	<b>Program</b>	<b>5</b>
<b>4</b>	<b>Používání programu</b>	<b>6</b>
<b>5</b>	<b>Ukázka</b>	<b>8</b>

## Seznam obrázků

1	Umíst'ování bodů . . . . .	6
2	Vykreslení polygonu . . . . .	8
3	Triangulace polygonu . . . . .	9
4	Trapezoidace polygonu . . . . .	9
5	Konvexní obal . . . . .	10
6	Delaunayho triangulace . . . . .	10
7	Nejmenší kružnice množiny bodů . . . . .	11

# 1 Úvod

Mezi důležité oblasti počítačové grafiky patří diskretizace obrazu na konečný počet bodů. V této práci se nebudeme zabývat samotnou prací s obrazem, ale geometrickými algoritmy pro množinu bodů, kterou můžeme obraz reprezentovat. Způsobů diskretizace je několik. Vhodným kandidátem se nabízí triangulace, která je vhodná pro strojové výpočty. Cílem bude implementovat jisté algoritmy, které zpracují zadanou množinu bodů, a jejich výstup bude možné graficky zobrazit. Pokusíme se napsat algoritmy *convex hull* a *ear clipping* použitelné pro triangulaci konvexní množiny bodů a polygonu (předpokládáme jednoduchý), respektive. Další algoritmy jsou *nalezení nejmenší kružnice množiny bodů* a *rozdělení polygonu na lichoběžníky*. Tyto zmíněné algoritmy budeme následovně kombinovat a získáme tak jednotlivé funkce programu: Delauneho triangulace, nalezení konvexního obalu množiny bodů, nalezení nejmenší kružnice množiny bodů, rozdělení polygonu na lichoběžníky a triangulace polygonu. Na závěr si ukážeme softwarovou strukturu programu a způsob, jakým ho používat.

## 2 Algoritmy

Jak už bylo řečeno v úvodu, jádrem této práce je nastudovat a následně implementovat algoritmy. Pracovat budeme s reprezentací bodů, přímk, rovin, kružnic a trojúhelníků v kartézském souřadném systému, kdy nás bude zajímat jejich vzájemná poloha. Proto část zdrojového kódu bude mít za výstup vyhodnocení situací protínání přímk, protínání ploch, protínání přímky a plochy, vzdálenost dvou bodů, vzdálenost bodu od přímky a vzdálenost bodu od roviny. Pomocí tohoto algebraického vyjádření již můžeme implementovat jednotlivé algoritmy.

Nejjednodušším z algoritmů je nalezení nejmenší kružnice obsahující množinu bodů. Tento byl implementován tzn. *hrubou silou*. Principem je vybrat trojici bodů, která jednoznačně určí polohu a poloměr kružnice. Pro ni pak zkontrolujeme, zda-li jsou všechny body uvnitř a uložíme její obsah, je-li menší než aktuální hodnota. Tento algoritmus má náročnost  $O(n^4)$ . Jako další algoritmus vezmeme rozdělení polygonu na lichoběžníky. Polygon jako geometrický útvar není jednoduchý na zpracování. Proto se často dělí na části, se kterými pracuje daleko snáze. To jsou např. lichoběžníky. Tento algoritmus jsem implementovali tak, že základny vzniklých lichoběžníků jsou vodorovné s osou  $x$ . Algoritmus pracuje tak, že přetřídí body polygonu podle souřadnice  $y$  a pak každým jedním bodem proloží přímku rovnoběžnou s osou  $x$ . Podle počtu průniků nalezne základnu lichoběžníku. Po proložení všech bodů je původní polygon rozdělen na lichoběžníky a trojúhelníky.

Asi nejintuitivnějším algoritmem je tzv. *ear clipping* [2]. Náhled na práci tohoto algoritmu je snadný. Představme si polygon vystřižený z papíru. Budeme postupně dokola ohýbat jeho rohy (konvexní body). Tím se stále zmenšuje počet hran až zůstanou jen tři. Algoritmus je jednoduchý, bereme po sobě jdoucí tři body polygonu, a když v takto vzniklém trojúhelníku žádný další bod neleží, tak ho "ohneme". Triangulace polygonu samozřejmě není jednoznačná.

Posledním z algoritmů je *convex hull*, určený pro hledání konvexního obalu množiny bodů. V programu je implementován pro množinu bodů v trojrozměrném prostoru. Tím se dá

jednoduchou konfigurací bodů získat i konvexní obal množiny bodů ve 2D, což je využito v programu. Hlavní výhodou ovšem přináší pro výpočet Delaunayho triangulace množiny bodů ve 2D, tj. taková triangulace, kdy je součet úhlů všech trojúhelníků maximální. Takovou konfiguraci získáme, když promítneme do roviny konvexní obal množiny 3D bodů, tak aby v obalu byl každý z bodů. V programu je každému z bodů doplněna z-tová souřadnice  $z = x^2 + y^2$ , čímž umístíme každý z bodů do konvexního obalu ve 3D [1].

Poznamenejme, že pro výpočet Delaunayho triangulace ve 3D je třeba implementovat *convex hull* ve 4D. Tím bychom získali přirozenou krystalovou strukturu, podobnou té, co vzniká v přírodě. Jenom připomeňme, že tento typ triangulace se nevyužívá pouze v počítačové grafice, ale v mnoha dalších oborech.

### 3 Program

Nyní k samotnému programu. Zdrojový kód je napsán v jazyku Java a spouštěl jsem ho na platformě Linux. Veškeré soubory lze stáhnout ve formátu zip na adrese [3].

Tento adresář obsahuje i skript RUN [4], který automaticky stáhne soubory a program spustí, a textový soubor README, kde je popsáno jak pracovat se soubory v příkazové řádce.

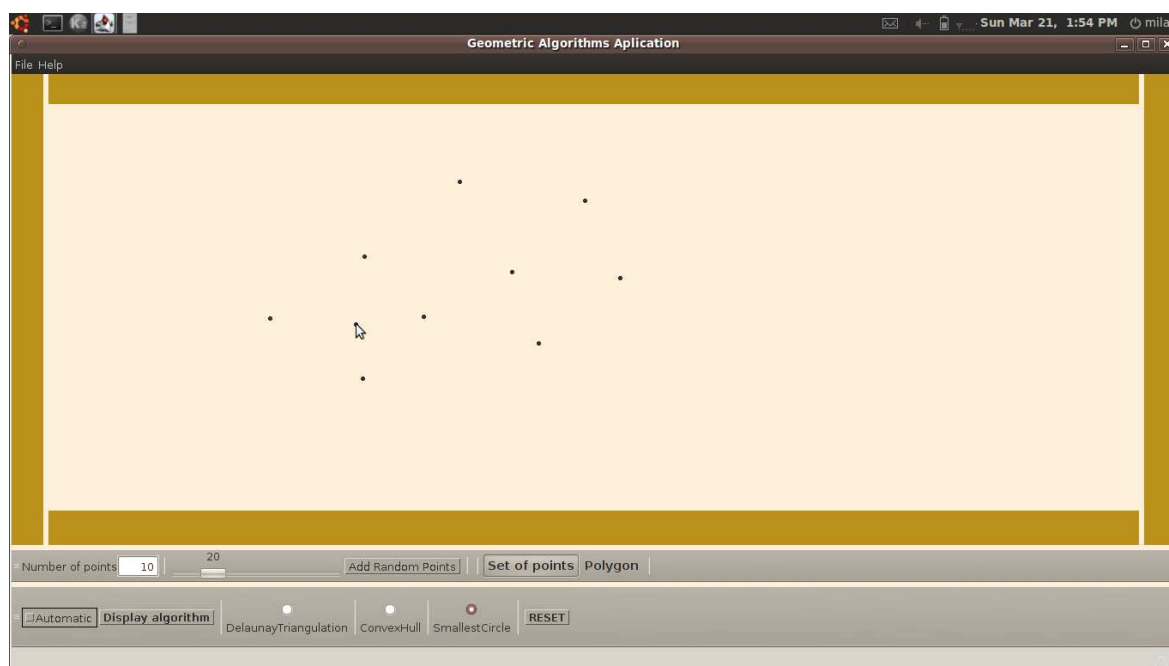
Zdrojový kód se skládá ze dvou nezávislých částí. První z nich je GUI, grafické prostředí, které jsem vytvářel v Netbeans. Druhá část obsahuje výpočetní jádro programu. Jednotlivé algoritmy nemají vzájemně objektově orientovanou strukturu. Jejich zapouzdření zprostředkovává interface, který se stará o předzpracování vstupních dat a o úpravu výstupních dat. Tento interface představuje prostor pro práci s programem, kde uživatel volá jednotlivé algoritmy.

Jednotlivé části zdrojového kódu popisuje javadoc dokument dostupný na stránce [6], kde jsou informace o vzájemném vztahu jednotlivých tříd a detailní informace o datových složkách a funkcích jednotlivých tříd.

Zdrojový kód ke grafickému prostředí vygenerovaly samy Netbeans. Protože se nejedná o rozsáhlou aplikaci, je grafické prostředí jednoduché a intuitivní. Jeho používání bude popsáno v následujícím textu a bude doplněno screenshoty.

## 4 Používání programu

Na Obrázku 1 je zobrazeno okno zobrazivší se po spuštění programu. Jak již bylo řečeno, algoritmy pracují s množinou bodů. Tu můžeme získat dvěma způsoby. Buď je naklikáme do vyznačeného prostoru nebo je můžeme nechat vygenerovat programem. Na takové body následně aplikujeme jednotlivé algoritmy z nabídky. Dále si popíšeme jednotlivá tlačítka GUI.



Obrázek 1: Umíst'ování bodů

V horní liště okna se nachází tlačítko **File** a tlačítko **Help**. První z nich slouží ke standardnímu ukončení programu, druhé po stisknutí nabízí záložku **About...**, která zobrazí informace o programu. Přejdeme-li přes zobrazovací prostor ohraničený hnědými okraji ke spodní části okna, vidíme zde ovládací lišty programu.

Na horní liště se nacházejí zleva: pole zobrazující počet umístěných bodů, slider na určení počtu bodů, který chceme generovat (defaultně zadává deset bodů), tlačítko **Add Random Points**, po jehož stisknutí se náhodně umístí body, a dále dvě tlačítka, kterými přepínáme mezi režimy **Set Of Points** a **Polygon**.

Posledně řečeným přepínačem se mění položky na spodní liště. Na ní se ve směru zleva nachází tlačítko **Display algorithm**, po jehož stisknutí se zobrazí výsledek vybraného algoritmu na zadané body. Abychom toto tlačítko nemuseli mačkat po zadání každého nového bodu, existuje vedle něj volba **Automatic**, která zobrazí výsledek algoritmu při přidání každého dalšího bodu. V závislosti na zvoleném režimu, **Set Of Points**(Množina bodů) nebo **Polygon**, se na liště vpravo zaobrazují odpovídající algoritmy.

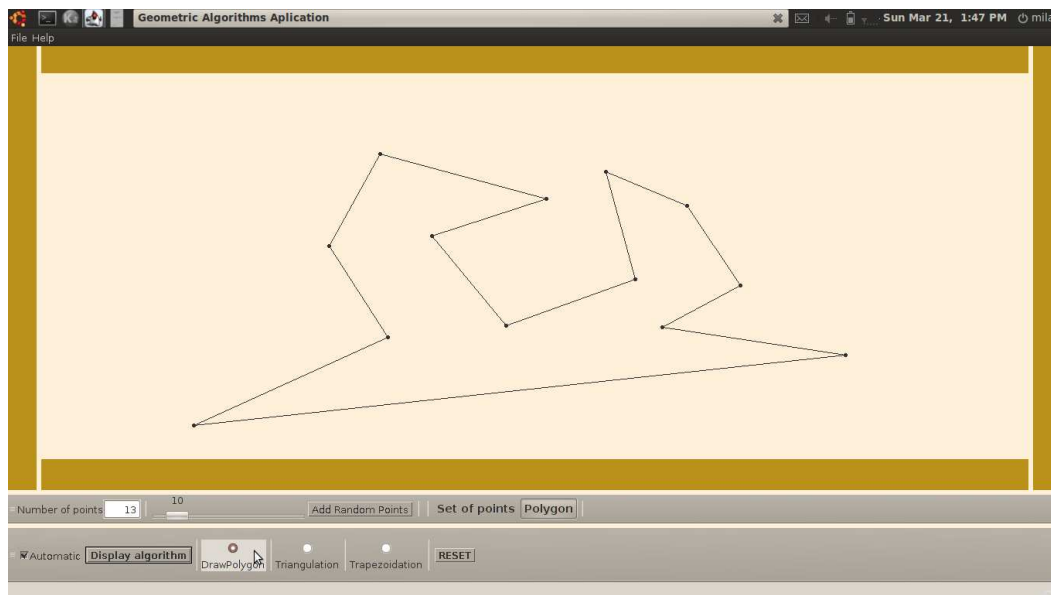
Pro případ **Set Of Points**(Množina bodů) jsou to algoritmy pro Delaunayho triangulaci,

vykreslení konvexního obalu množiny bodů a nejmenší kružnice uzavírající množinu bodů. Pro případ **Polygon** jsou to vykreslení polygonu, triangulace polygonu a trapezoidace polygonu (rozdělení polygonu na lichoběžníky). Posledním tlačítkem GUI je **RESET**, které vymaže umístěné body.

## Reference

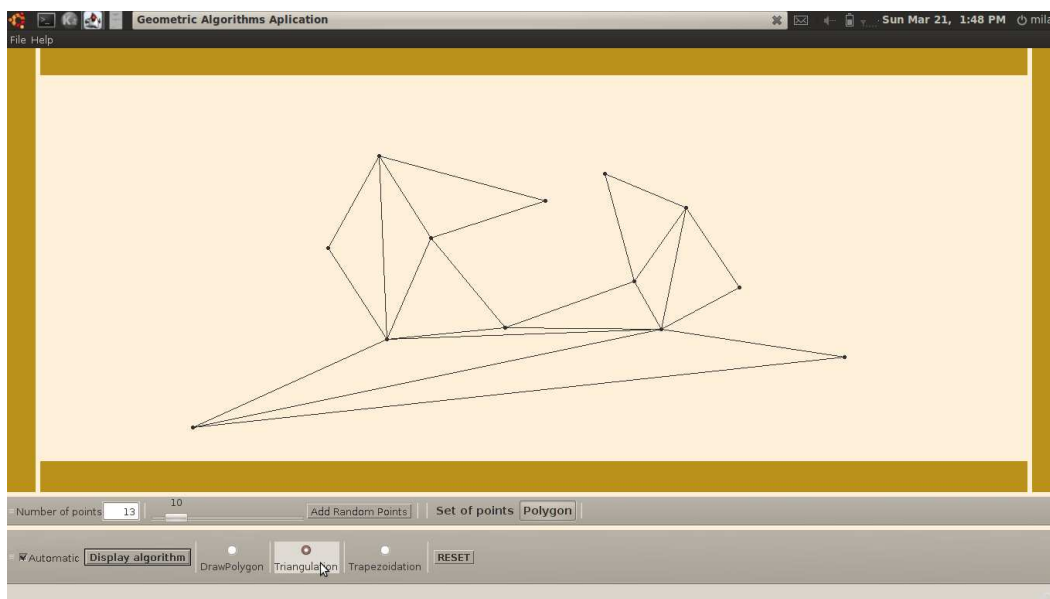
- [1] <http://www.cse.unsw.edu.au/~lambert/java/3d/delaunay.html>
- [2] <http://www.geometrictools.com/Documentation/TriangulationByEarClipping.pdf>
- [3] <http://kfe.fjfi.cvut.cz/~holec/projects/POGR1/GeometricAlgorithms.zip>
- [4] <http://kfe.fjfi.cvut.cz/~holec/projects/POGR1/RUN>
- [5] <http://kfe.fjfi.cvut.cz/~holec/projects/POGR1/README>
- [6] <http://kfe.fjfi.cvut.cz/~holec/projects/POGR1/javadoc/>

## 5 Ukázka

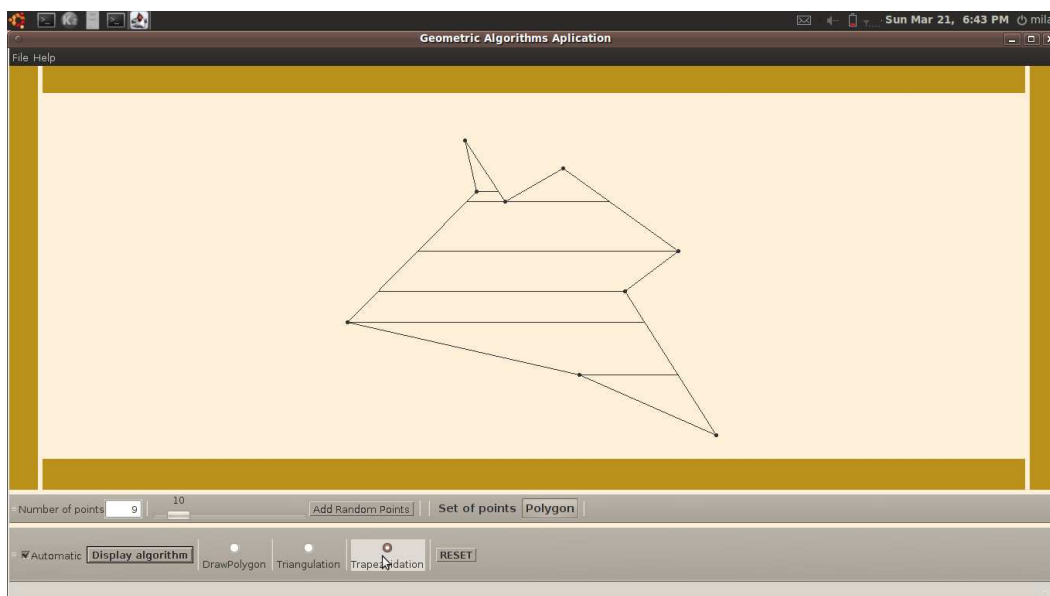


Obrázek 2: Vykreslení polygonu

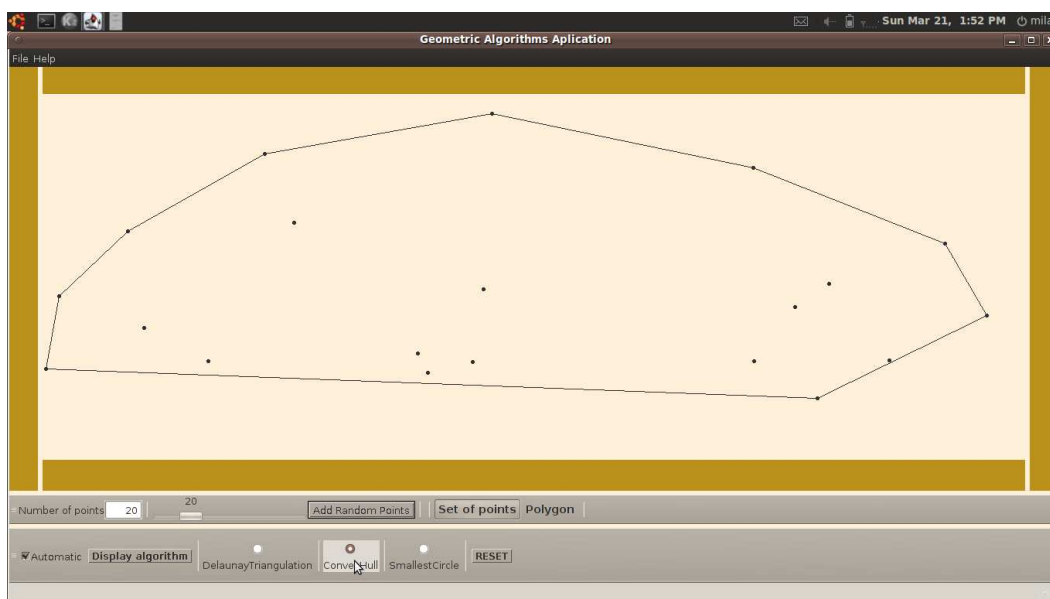




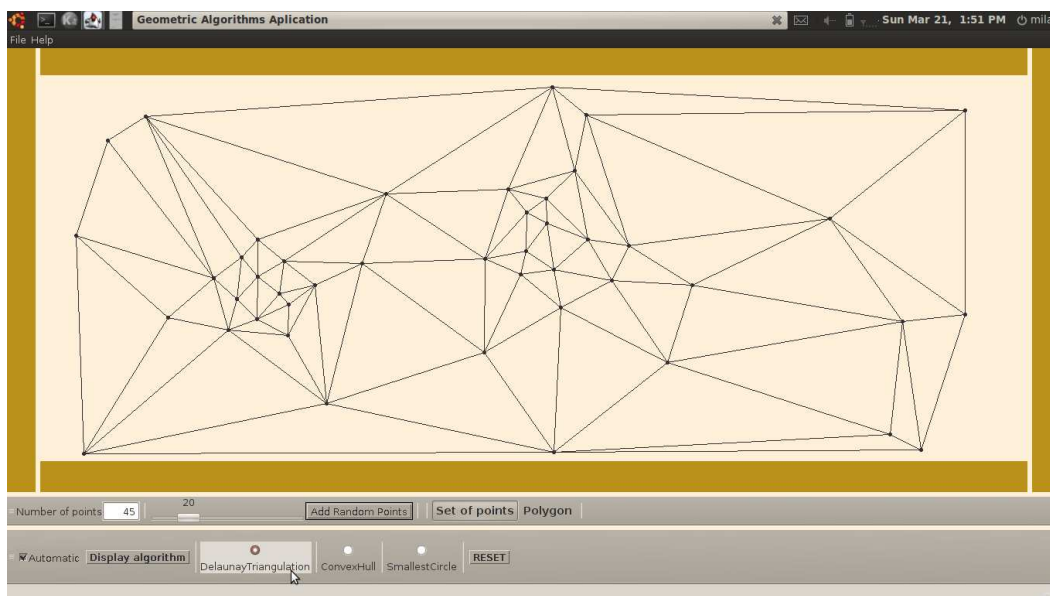
Obrázek 3: Triangulace polygonu



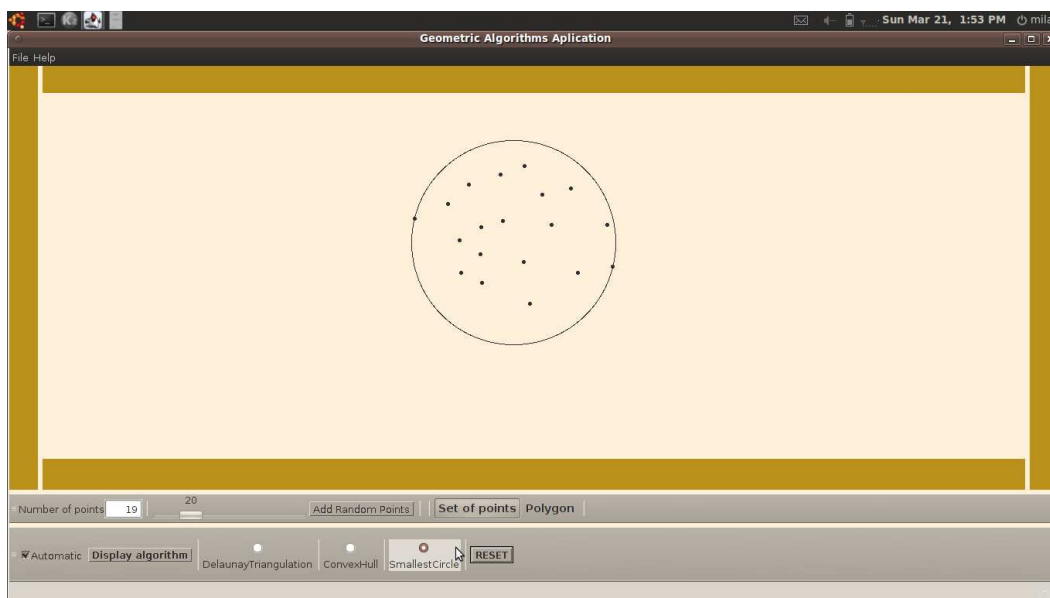
Obrázek 4: Trapezoidace polygonu



Obrázek 5: Konvexní obal



Obrázek 6: Delaunayho triangulace



Obrázek 7: Nejmenší kružnice množiny bodů