VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

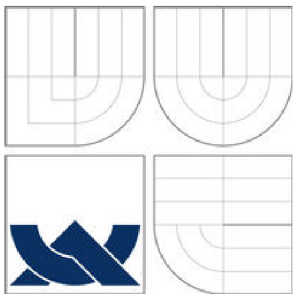
VÝUKOVÝ PROGRAM PRO DEMONSTRACI OŘEZÁVÁNÍ 2D OBJEKTŮ A VYPLŇOVÁNÍ 2D UZAVŘENÝCH OBLASTÍ

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

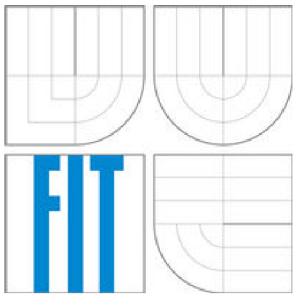
AUTOR PRÁCE
AUTHOR

LENKA NOVOTNÁ

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VÝUKOVÝ PROGRAM PRO DEMONSTRACI OŘEZÁVÁNÍ 2D OBJEKTŮ A VYPLŇOVÁNÍ 2D UZAVŘENÝCH OBLASTÍ

EDUCATION COMPUTER PROGRAM FOR DEMONSTRATION OF 2D ENTITY TRIMMING AND 2D CLOSED REGIONS FILLING

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

LENKA NOVOTNÁ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VÍT ŠTANCL

BRNO 2008

Zadání bakalářské práce/7130/2007/xnovot77

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2007/2008

Zadání bakalářské práce

Řešitel: **Novotná Lenka**
Obor: Informační technologie
Téma: **Vyukový program pro demonstraci ořezávání 2D objektů a vyplňování 2D uzavřených oblastí**
Kategorie: Počítačová grafika

Pokyny:

1. Prostudujte problematiku ořezávání 2D objektů a vyplňování 2D uzavřených oblastí v počítačové grafice.
2. Prostudujte problematiku tvorby výukových a demonstračních programů.
3. Navrhněte výukový program pro demonstraci principu ořezávání 2D objektů a vyplňování 2D uzavřených oblastí v počítačové grafice.
4. Implementujte navržený program ve vybraném jazyce (C/C++, Java, Python, C#).
5. Ve vytvořeném programu demonstrujte a implementujte popsané principy a metody.

Literatura:

- Žara J., Beneš B., Felkel P.: Moderní počítačová grafika. 1. vyd. Praha, Computer press 1998, 448 s., ISBN 80-7226-049-9

Při obhajobě semestrální části projektu je požadováno:

- Bez požadavků.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Štancl Vít, Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2007
Datum odevzdání: 14. května 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Pavel Zemčík
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Slečna

Jméno a příjmení: **Lenka Novotná**
Id studenta: 84225
Bytem: Lišov 2, 373 73 Štěpánovice u Č.Bud.
Narozena: 11. 05. 1985, České Budějovice
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP: Vyukový program pro demonstraci ořezávání 2D objektů a vyplňování 2D uzavřených oblastí
Vedoucí/školitel VŠKP: Štancl Vít, Ing.
Ústav: Ústav počítačové grafiky a multimédií
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:
tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☒ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

....

Nabyvatel Autor

Abstrakt

Tato práce se zabývá návrhem a vývojem aplikace pro demonstraci ořezávání a vyplňování 2D uzavřených oblastí. Jsou zde popsány jednotlivé algoritmy, jejich vlastnosti a vzájemné rozdíly. Aplikace byla navrhnuta tak, aby názorným způsobem ukázala právě tyto vzájemné odlišnosti a usnadnila jejím budoucím uživatelům pochopení této problematiky.

Klíčová slova

Výukový program, vyplňování oblastí, řádkové vyplňování, inverzní řádkové vyplňování, Pinedův algoritmus, semínkové vyplňování, algoritmus Sutherland-Hodgman, algoritmus Weiler-Atherton, wxWidgets.

Abstract

This bachelor`s thesis is focusing on concept and developement of educational computer program for demonstration of 2D entity trimming and 2D closed regions filling. The characteristics and mutual differences of all important algorithms are described here. The aplication was designed to ilustrate their differences and to make understandig of this topic easier for future users.

Keywords

Educational program, regions filling, line filling, inverse line filling, Pineda algorithm, seed filling, Sutherland-Hodgman algorithm, Weiler-Atherton algorithm, wxWidgets

Citace

Lenka Novotná, Výukový program pro demonstraci ořezávání 2D objektů a vyplňování uzavřených 2D oblastí, bakalářská práce, Brno, FIT VUT v Brně, 2008

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením Ing. Víta Štancla. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

……

Lenka Novotná

14.5.2008

Poděkování

V první řadě bych chtěla poděkovat vedoucímu mé bakalářské práce, Ing. Vítu Štanclovi, za jeho cenné připomínky a postřehy a vedení až ke zdárnému konci projektu.

Dále bych chtěla poděkovat všem, kteří se mnou v době psaní této práce měli trpělivost a podporovali mě.

Obsah

1 Úvod

V současné době již téměř všechny grafické editory obsahují nástroje pro vyplnění nakreslené oblasti barvou, vzorem, šrafou či jiným obrázkem. Pro běžného uživatele je způsob vyplnění a vykreslení možná nepodstatný, avšak studentům a lidem s hlubším zájmem o počítačovou grafiku se může toto téma zdát přinejmenším zajímavé.

Úkolem této práce bylo vytvořit aplikaci výukového typu, která by měla zpracovávat téma vyplňování uzavřených oblastí a ořezávání polygonů. Výchozími materiály pro zpracování teoretické části byla skripta Algoritmy počítačové grafiky [1] a přednášky předmětu Základy počítačové grafiky [2].

Práce sestává ze dvou částí. První se věnuje vysvětlení pojmů a látky, na které je založena a kterou zpracovává. Druhá část popisuje, jakým způsobem jsou algoritmy vyplňování a ořezávání zpracovány a jak jsou začleněny do aplikace, která je využívá.

2 Výukový program

V dnešní době dochází ke stále častějšímu začleňování elektronických didaktických nástrojů do běžné výuky. Umožňují studujícímu snazší utříbení nabytých informací a jejich předvedení v praxi. Úkolem každého takového programu by mělo být co nejnázorněji a nejsrozumitelněji předvést a vysvětlit látku, které se týká. Avšak na výukový program nejsou kladeny pouze pedagogické požadavky, ale také požadavky uživatelské. Rozdělme je proto do těchto dvou skupin.

2.1 Pedagogické požadavky

Z pedagogického hlediska jsou prioritní vlastnosti názornost, přehlednost a praktický přínos. Názorná je aplikace tehdy, je-li doprovázena promyšleným popisem, díky němuž se žák jednoznačně dobírá k prezentovaným myšlenkám a informacím. U výukového grafického editoru může jít například o vykreslování pomocných čar či možnost nastavení rychlosti vykreslování tak, aby bylo jasné a zřetelné, co se v tu chvíli na plátně děje.

Přehlednost aplikace vždy závisí na autorově cítění, měly by se však dodržovat jisté zažité standardy. Jako například nepoužívat příliš mnoho příliš výrazných barev, ovládací prvky s podobnou funkcí dělit do samostatných celků a jiné.

Praktický přínos aplikace by měl být zřejmý – vysvětlit látku a její principy žákovi.

2.2 Uživatelské požadavky

Často se dnes setkáváme s pojmem, že je aplikace „uživatelsky přátelská“. Taková aplikace se snaží být co nejvíce přizpůsobena intuitivnímu chápání uživatele, případně se příliš neodlišovat od již běžně rozšířených vzorů. Většina uživatelů preferuje klasické rozmístění ovládacích prvků před inovativními změnami či naprosto odlišným designem. I člověk, který se setkává s aplikací poprvé, se ve standardně navržené aplikaci snadno zorientuje a práce je pro něj příjemnější, než když jsou ovládací prvky na jiných, nečekaných místech.

Pro lepší představu je to u grafického editoru například umístění panelu s nástroji do svislého panelu v levé části aplikace a kreslící plátno v části pravé.

K dalším kritérií uživatelsky přátelské aplikace patří snadná instalace aplikace nebo její dostupná a obsáhlá dokumentace.

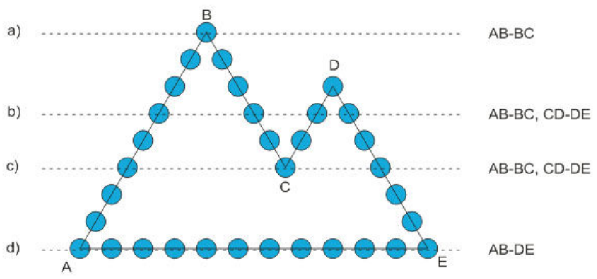
3 Vyplňování uzavřených oblastí

Oblast je vymezená část plochy, která je vůči ní jistým způsobem vyznačena. Její hranice mohou být definovány dvěma způsoby a to geometricky nebo rastrově. Geometrické určení spočívá ve využití množiny křivek pro označení dané oblasti nebo může být vymezena posloupností bodů, které definují mnohoúhelník. Pro takto vyznačenou oblast můžeme využít například vyplňovací algoritmus Pinedův, Řádkový či Inverzní řádkový. Hranice rastrově vymezené oblasti jsou určeny množinou bodů, které mají určitou vlastnost. Rozhodující je vždy buď barva hranice nebo barva vnitřních bodů. K vyplnění tohoto typu oblastí se využívá algoritmů založených na Semínkovém vyplňování. Všechny tyto algoritmy budou podrobněji popsány v následujících kapitolách.

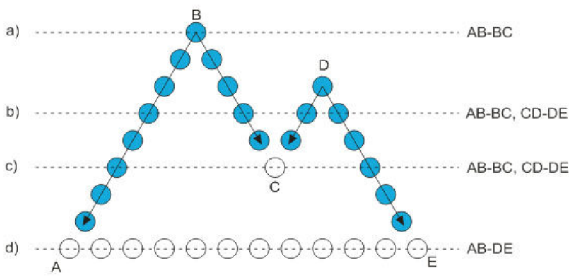
3.1 Řádkové vyplňování

Tato metoda patří mezi základní algoritmy používané pro vyplňování geometricky vyznačených oblastí, protože s ní lze vyplňovat nejen polygony (konvexní i nekonvexní), ale i jakoukoliv oblast určenou obecnou křivkou.

Každým řádkem rastru vedeme pomyslnou čáru a hledáme její průsečíky s hraniční čárou oblasti. Tyto průsečíky pak seřadíme dle hodnoty osy *x* a vždy dvojice lichého a sudého průsečíku definují úsečku ležící uvnitř ohraničené plochy (viz Obrázek 1.1 b). Je tedy nutné, abychom vždy měli k dispozici sudý počet průsečíků. Pokud se dostaneme do situace, kdy je k dispozici nemáme (Obrázek 1.1. c), je nejsnazším řešením zkrácení hranice přímky procházející daným vrcholem o jeden bod ve směru osy *y*. Provedeme-li toto zkrácení se všemi hraničními přímkami, snížíme tím počty průsečíků a tedy i pravděpodobnost komplikací. Je-li hrana vodorovná, je ze seznamu vypuštěna, protože má s rozkladovým řádkem nekonečně mnoho průsečíků. Zkrácení hran a vypuštění vodorovné hrany je zobrazeno na Obrázku 1.2. Má-li úsečka nulovou délku, je vykreslena jako jeden bod (viz Obrázek 1.1 a).



Obrázek 1.1



Obrázek 1.2

Samotný algoritmus pak vypadá takto:

1. Pro všechny hraniční úsečky ověř:
   a) je-li vodorovná, vynechej ji
   b) uprav orientaci shora dolů a zkraťjejí délku o 1 ve směru osy y
   c) aktualizuj mezní souřadnice hranice y_{max} a y_{min}
2. Pro y od y_{min} do y_{max} s krokem 1 proveď:
   a) nalezni průsečíky hraničních úseček s řádkem y
   b) uspořádej všechny průsečíky podle souřadnice x
   c) vykresli úseky mezi lichými a sudými průsečíky.
3. Vykresli hranici oblasti

Takto napsaný algoritmus však není příliš efektivní, protože všechny hodnoty počítá znovu a nevyužívá dříve vypočítaných výsledků. Největší časovou úsporu získáme, když se nám podaří urychlit hledání průsečíků s rozkladovým řádkem. Dá se předpokládat, že v následujícím průchodu bude mít rozkladový řádek s většinou hran stejný počet průsečíků a to dokonce ve stejném pořadí, jako v minulém průchodu. Z tohoto předpokladu vychází níže popsaný způsob urychlení algoritmu.

3.1.1 Seznam aktivních hran

První, co je nutné vytvořit, je tabulka hran. To je datová struktura obsahující pole seznamů, kde každá položka představuje jednu řádku. Jsou seřazeny podle maximální hodnoty souřadnice y a každý záznam obsahuje tyto položky: souřadnici y_{min} koncového bodu, souřadnici x_{max} průsečíku s rozkladovým řádkem a přírůstek Δx přechodu na další řádek.

Dále je nutné vytvořit ještě jednu datovou strukturu – Seznam aktivních hran, který bude obsahovat pouze ty hrany, které mají průsečík s právě zpracovávaným rozkladovým řádkem. Při každé nově zpracovávané řádce se aktualizuje seznam aktivních hran (vyloučí hrany ležící nad řádkem či přidá nové z tabulky hran), aktualizuje hodnoty x hran a uspořádá se podle hodnoty x. Řazení je obvykle řešeno metodou Bubble-sort.

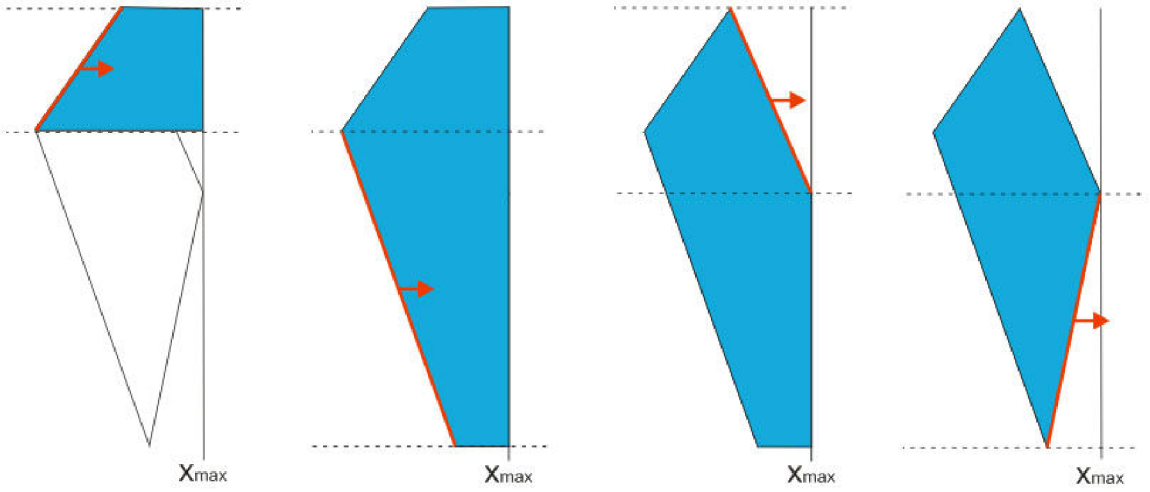
Algoritmus s použitím seznamu aktivních hran:

1. Bod jedna je stejný, jako u klasického řádkového algoritmu
2. Vytvoř tabulku hran TH a seznam aktivních hran SAH
3. Nastav $y = y_{max}$ a dokud nejsou TH a SAH prázdné, opakuj
   a) přidej do SAH hrany z TH[y] a uspořádej je podle x
   b) vykresli úseky mezi lichými a sudými průsečíky.
   c) vyřaď ze SAH hrany, které končí nad řádkem ($y_{min} = y$)
   d) sniž y o 1 a aktualizuj souřadnice průsečíků

3.2 Inverzní řádkové vyplňování

Metoda inverzního vyplňování spočívá ve zpracovávání každé hraniční přímky zvlášť. U tohoto typu algoritmu je také výhodné vynechat vodorovné hranice oblasti a všechny hranice zdola zkrátit o 1 bod ve směru osy *y*, stejně jako u Řádkového vyplňování.

Pro každou souřadnici *y* hraniční přímky vygenerována přímka o souřadnici *x* vedoucí až k bodu s maximální *x-ovou* souřadnicí polygonu x_{max}. Hodnota x_{max} zde slouží jako tzv. plot, která umožňuje časově efektivnější postup algoritmu. Bez něj by docházelo k vyplnění řádku od *x-ové* souřadnice hrany až po konec kreslícího plátna. Každý bod této přímky je pak operací XOR buďto nastaven na požadovanou barvu, nebo pokud již tuto barvu má, barvy zbaven. Při vykreslování dalších hraničních přímek dochází díky opětovné inverzi k smazání částí, které byly obarveny navíc, případně k dovyplnění těch částí oblasti, které dosud vyplněny nebyly.



Obrázek 2 – Inverzní řádkové vyplňování

Problém může nastat, není-li vyplňovaná oblast prázdná. Barevnou inverzí bychom totiž nedosáhli překrytí původní kresby, ale pouze negace stávajících barev. Proto se využívá tzv. šablony. To je paměťová oblast, ve které se provádí jednotlivé operace během výpočtu. Její velikost musí odpovídat velikosti vyplňované předlohy, je tedy relativně dost náročná na paměť. Vyplňovaná oblast je zde reprezentována nastavením bitů na stejnou hodnotu, zápisem do šablony dochází tedy pouze k jejich negaci. Po dokončení vyplňování v šabloně jsou podle ní zpátky do bitmapy zakresleny pouze pixely, jejichž bit v šabloně nese příznak obarvení. Výhodou je velká rychlost, protože u tohoto typu algoritmu není nutné průsečíky třídit.

Algoritmus inverzního řádkového vyplňování s využitím šablony:

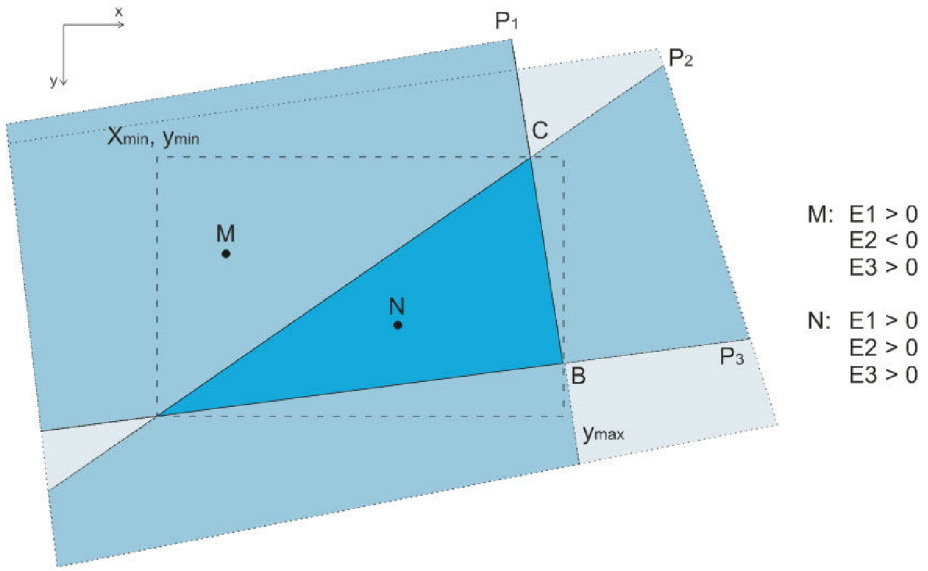
1. Pro všechny hraniční úsečky ověř:
   a) je-li vodorovná, vynechej ji
   b) uprav orientaci shora dolů a zkrať její délku o 1 ve směru osy y
2. Vynuluj šablonu
3. Pro každou nevodorovnou hranu
   a) nalezni y_{min}, y_{max} a x_{max}
   b) pro každý bod od y_{min} do y_{max} invertuj místo v šabloně
4. Vykresli hranici oblasti

3.3 Pinedův algoritmus

Tento algoritmus lze použít pouze pro vyplňování konvexních mnohoúhelníků. Pokud bychom ho chtěli využít i u nekonvexních polygonů, museli bychom ho před začátkem rozdělit do více konvexních částí. Jeho hlavní myšlenka spočívá ve využití hraniční přímky oblasti jako hraniční přímky poloroviny. Vnitřek konvexní oblasti je tedy určen jako průnik těchto polorovin. Pro určení, zda se bod nachází uvnitř nebo vně oblasti, slouží tzv. hranová funkce, kterou lze vypočítat pomocí vektorového součinu.

Vzorec pro hranovou funkci: $E(x,y) = (x - X).\Delta Y - (y - Y).\Delta X$

Bod je považován za vnitřní, je-li v daném bodě hodnota hranové funkce pro všechny orientované hraniční přímky oblasti větší než nula. Díky linearitě hranové funkce je Pinedův algoritmus velmi rychlý a snadno hardwarově implementovatelný.



Obrázek 3 – Pinedův algoritmus, průnik polorovin

Postup Pinedova algoritmu:

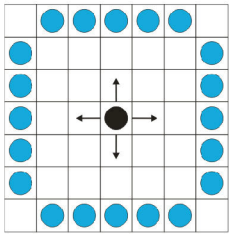
1. Nalezni minimální a maximální souřadnice oblasti x_{min}, x_{max}, y_{min}, y_{max}
2. Pro každou hranu oblasti inicializuj hranovou funkci $Ei(x_{min},y_{max})$
3. Pro všechny body v oblasti (x_{min}, x_{max}), (y_{min}, y_{max}) aktualizuj $Ei(x,y)$ a je-li tato větší než nula, vyplň bod $[x, y]$

Pokud bychom chtěli tento algoritmus ještě urychlit, nabízejí se nám dvě řešení. První z nich je založeno na tom, že neprocházíme celou oblast od maximálních do minimálních souřadnic, ale při každém dosažení vnější poloroviny se posuneme o řádek níže a obrátíme směr procházení. Při tomto postupu se však může stát, že se dostaneme rovnou dovnitř oblasti a zůstanou nám některé body, které už bychom nijak nemohli projít. Je tedy potřeba, abychom vždy po posunutí o řádek níže nejprve nalezli hranici oblasti a teprve poté otočili směr procházení. Druhý způsob spočívá v rozdělení oblasti na dvě části (například u trojúhelníku přímkou procházející vrcholem) a v procházení každé části zvlášť směrem k jejím hranicím.

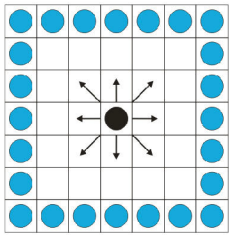
3.4 Semínkové vyplňování

Algoritmus semínkového vyplňování lze využít pouze u oblastí vymezených rastrově. Čtením z obrazové paměti získává informace o tom, kudy vede hranice oblasti. Za hranici může být považován každý bod, který nemá barvu semínka. Pokud má bod jinou barvu, než je zadaná barva hranice, jedná se o tzv. hraniční vyplňování, pokud má jinou barvu než je barva původního semínka, jedná se o tzv. záplavové vyplňování. Základem je „semínko“, což je vlastně uživatelem vybraný výchozí bod. Po obarvení semínka dochází v cyklu k obarvení jeho sousedů. Vnitřní oblast můžeme rozdělit do dvou skupin:

a) **4-okolí** je oblast, kde mezi jednotlivými dvěma body existuje cesta složená pouze z vodorovných a svislých kroků uvnitř oblasti, viz Obrázek 4.1
b) **8-okolí** je oblast, kde mezi jednotlivými dvěma body existuje cesta složená z vodorovných, svislých a diagonálních kroků uvnitř oblasti, viz Obrázek 4.2



Obrázek 4.1



Obrázek 4.2

Rekurzivní algoritmus semínkového vyplňování

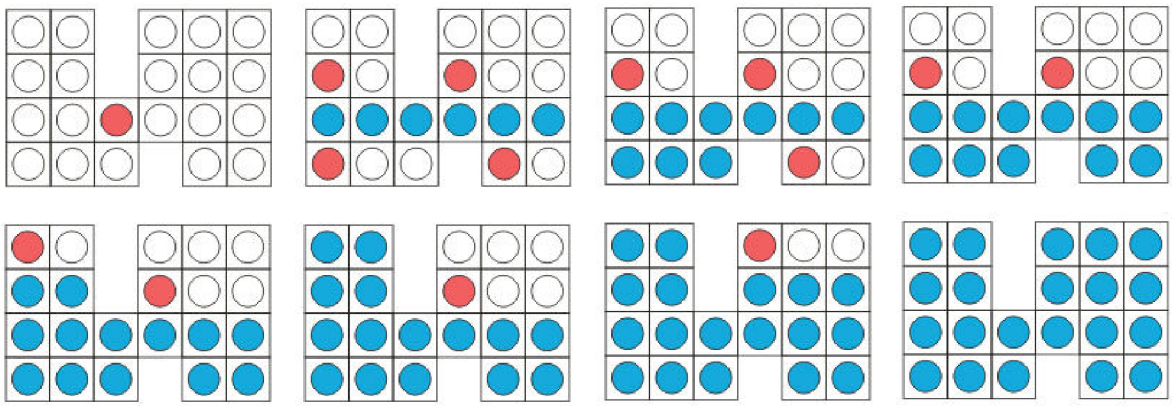
1. Otestuj, zda daný bod [*x, y*] netvoří hranici, nebo zda už nebyl vyplněn
   a) pokud ano, skonči
   b) pokud ne, obarvi jej
2. Pro všechny sousedy, tedy body o souřadnicích [*x + 1, y*], [*x – 1, y*], [*x, y + 1*] [*x, y - 1*] opakuj tento algoritmus

Zásobníkový algoritmus semínkového vyplňování

1. Vlož semínko do fronty
2. Dokud není fronta prázdná, opakuj
   a) vyber semínko z fronty a obarvi jej
   b) otestuj každého jeho souseda a pokud netvoří hranici, nebo už nebyl vyplněn, vlož ho do fronty

Algoritmus řádkového semínkového vyplňování

Rekurzivní algoritmus není v reálu příliš použitelný. Je paměťově velmi náročný a také není příliš efektivní. Při rozpínání semínek po celé oblasti a tedy určování jejich sousedů dochází k tomu, že některý bod je testován i několikrát poté, co již byl obarven. Proto došlo k modifikaci algoritmu a to tak, že v zásobníku jsou uchovávány pouze souřadnice několika málo vnitřních bodů oblasti. Vždy, když je bod ze zásobníku vybrán, jsou všechny body v tomto řádku obarveny až po nalezenou hranici oblasti. V řádku s vyšší *y* souřadnicí jsou pak hledány souvislé vnitřní úseky a z každého je uložen jeden bod na zásobník. Totéž je provedeno v řádku s nižší *y* souřadnicí.



Obrázek 5 – Řádkové semínkové vyplňování

3.5 Vyplňování vzorem, šrafování

Pro vyplnění zadané oblasti vzorem je nutné mít tento vzor někde uložen. Bývá popsán maticí o rozměrech *m* x *n*, ve které jednotlivé prvky určují barvu odpovídajících pixelů. Index prvku matice se vypočítá jako souřadnice bodu *modulo* (zbytek po celočíselném dělení) velikost matice. V případě, že je ve vzoru obsažena původní barva oblasti, může při vyplňování rastrově určené oblasti dojít k zacyklení. Tomu se dá předejít zamezením opakovaného čtení z obrazové paměti. Úprava algoritmu spočívá v rozšíření množství informací ukládaných zásobník. U řádkového semínkového vyplňování jsou to například souřadnice *y*, *xl* a *xr* vodorovného úseku a směr, odkud bylo označeno.

Šrafování geometricky určených oblastí není nijak těžké implementovat. Nejvhodnější algoritmus pro tuto operaci je algoritmus řádkového vyplňování. Postup je zde stejný jako při vyplňování jednolité plochy, avšak změníme hodnotu kroku z 1 na celé kladné číslo *m*, kde *m* je rozteč daných šraf. Pro vykreslení šikmých šraf je třeba nejprve všechny hrany otočit o daný úhel α, vyplnit oblast normálním řádkovým vyplňováním a ještě před vykreslením vše o úhel α otočit zpět. Šrafování rastrových oblastí je poměrně složitá záležitost, kterou lze řešit například použitím šablony nebo převedením šrafy na vyplňovaní vzor.

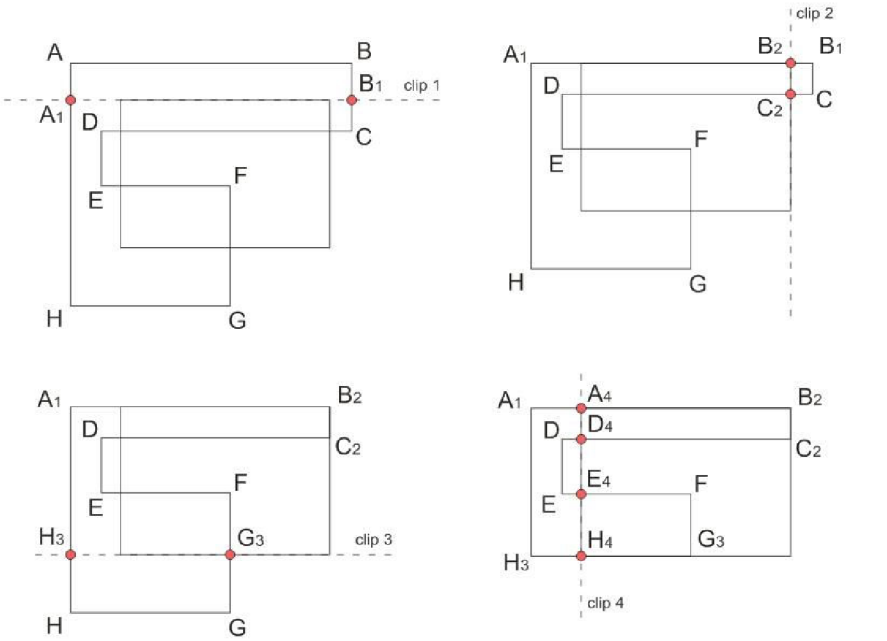
4 Ořezávání 2D polygonů

Ořezávací algoritmy nám umožňují odstranit ty části obrazu, které leží mimo oblast ořezávací plochy. Ta bývá definována jako konvexní mnohoúhelník ohraničený úsečkami, nejčastěji jako obdélník. Polygon je oblast s uspořádaným počtem vrcholů, kde první a poslední vrcholy jsou identické. Je žádoucí, aby i po oříznutí uzavřených polygonů, byly tyto oblasti uzavřené. Pro ořez polygonů slouží algoritmy Sutherland-Hodgman a Weiler-Atherton.

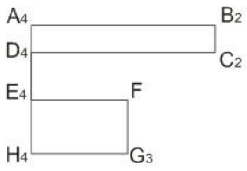
4.1 Sutherland-Hodgman

Tento algoritmus postupně ořezává polygon podle jednotlivých hranic, tedy vždy polorovinou. Nejprve si vytvoří seznam vrcholů polygonu. Pak podle jedné hrany ořezového okna vytvoří pomyslnou přímku a zjistí její průsečíky s polygonem. Ty doplní do seznamu vrcholů a tak jej aktualizuje. Stejným způsobem pokračuje u ostatních hran okna. Výstupem je posloupnost bodů tvořící hranici ořezaného polygonu.

Výhodou tohoto algoritmu je, že pro ořezávání lze použít stejný kód, pouze se vždy o 90° natočí jak polygon, tak okno. Otočení se provádí pouhou záměnou *x-ových* a *y-nových* souřadnic.



Obrázek 6 – Sutherland-Hodgmanův algoritmus



Obrázek 7

Jeho nevýhoda se projeví při ořezávání nekonvexních polygonů. Při vytváření nové hranice polygonu je po oříznutí vytvořena nová hrana i v místech, kde my být neměla. Například máme-li polygon ve tvaru ležatého U a ořízneme-li ho v jeho pravé části, nevzniknou nám dva samostatné obdélníkové polygony, ale budou spojeny čárou, viz Obrázek 7.

Algoritmus Sutherland-Hodgman:

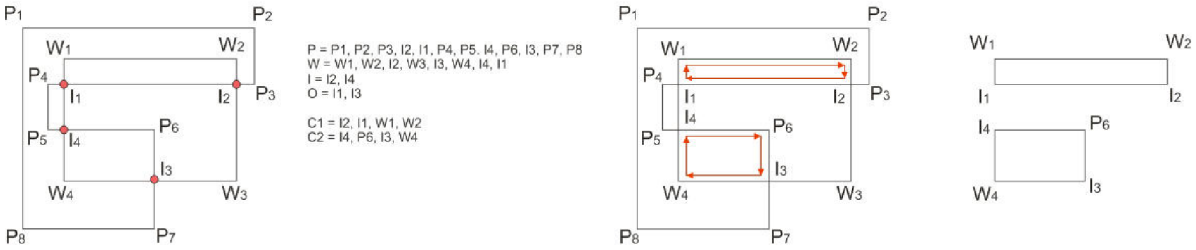
1. Vytvoř posloupnost vrcholů polygonu
2. Pro každý bod (počáteční) a bod následující (koncový) vykonej:
   a) oba body jsou uvnitř, zapiš počáteční bod do výstupní posloupnosti vrcholů
   b) počáteční bod je uvnitř a koncový vně okna, vytvoř průsečík U s hranicí a zapiš ho do výstupní posloupnosti vrcholů
   c) oba body jsou vně, nic nezapisuj
   d) počáteční bod je vně a koncový uvnitř, vytvoř průsečík V s hranicí a zapiš ho do výstupní posloupnosti vrcholů
3. Opakuj pro další hrany okna a jako vstupní posloupnost vrcholů použij výstupní posloupnost z tohoto algoritmu

4.2 Weiler-Atherton

Velmi vhodný je tento algoritmus pro ořezávání nekonvexních polygonů, ořezávací okno může být také nekonvexní. Oba objekty, okno i polygon, jsou reprezentovány posloupností vrcholů, která je uzavřená a jednosměrně orientovaná. Tato posloupnost tvoří hranici, tedy i pomyslnou smyčku. Algoritmus využívá 3 seznamů, jeden pro vrcholy polygonu, druhý pro vrcholy okna, ve třetím jsou uloženy průsečíky těchto dvou objektů. Průsečíky si nejprve podle orientace polygonu a okna označíme jako vstupní nebo výstupní. Tyto průsečíky se nezapisují pouze do seznamu pro ně určeného, ale při průchodu po vrcholech polygonu i okna do jejich seznamů, vždy mezi příslušné vrcholy. Místa jejich zařazení jsou pak propojena obousměrným ukazatelem a tím vznikne ořezaný polygon s novými vrcholy.

Algoritmus Weiler-Atherton pro zachování polygonu uvnitř okna:

1. Vytvoř orientovaný seznam P vrcholů a průsečíků s oknem ležícím na polygonu
2. Vytvoř orientovaný seznam W vrcholů a průsečíků s polygonem ležícím na okně
3. Vytvoř orientovaný seznam I průsečíků a označ si, zda je vstupní (I) nebo výstupní (O)
4. Přečti první průsečík ze seznamu I:
   a) je-li vstupní, odeber ho ze seznamu I a pokračuj v seznamu P. Ulož všechny prvky seznamu až po další průsečík do výsledné posloupnosti, průsečík odeber ze seznamu I a pokračuj seznamem W.
   b) je-li výstupní, odeber ho ze seznamu I a pokračuj v seznamu W. Ulož všechny prvky seznamu až po další průsečík do výsledné posloupnosti, průsečík odeber ze seznamu I a pokračuj seznamem P.



Obrázek 8 – Postup Weiler-Athertonova algoritmu

Pokud chceme polygon oříznout tak, aby nám zbyly části, které leží mimo okno, stačí jen zaměnit pořadí procházení jednotlivých seznamů. Je-li první průsečík ze seznamu I vstupní, budeme dále číst ze seznamu W, a je-li výstupní budeme pokračovat v seznamu P.

5 Implementace výukové aplikace

Aplikace byla vytvořena primárně pro studenty 2. ročníku bakalářského studia na Fakultě informačních technologií VUT v Brně. Jako výuková pomůcka by jim měla pomoci pochopit základní principy a rozdíly mezi jednotlivými algoritmy vyplňování a ořezávání, které jsou součástí probírané látky předmětu Základy počítačové grafiky.

5.1 Vývojové prostředí

Celá aplikace byla napsána v jazyce C++ s využitím volně dostupného souboru knihoven wxWidgets, který umožňuje snadnou implementaci GUI a obsahuje veliké množství funkcí zajišťujících vykreslování a obsluhu ovládacích prvků. Právě díky této knihově bylo možné vytvořit program, jehož většina zdrojového kódu se týká samotného postupu algoritmů a pouze malá část popisu uživatelského prostředí. Dalšími důvody, proč byla použita knihovna wxWidgets, jsou, že se jedná o volně dostupný nástroj a že vytvořená aplikace není vázána na konkrétní platformu.

Platformou aplikace je operační systém MS Windows, je psána a kompilována ve vývojovém prostředí Microsoft Visual Studio 2005 s verzí wxWidgets 2.8.7.

5.1.1 wxWidgets

wxWidgets je soubor knihoven umožňující tvorbu uživatelského rozhraní multiplatformních aplikací. Aplikace, kterou jeho pomocí napíšete, si po přenesení na jinou platformu převezme její vzhled a nebude se tak příliš lišit od ostatních aplikací. Původně byla tato knihovna vytvořena pouze pro Unix a Windows, avšak v současné době je podporována i platformou MacOS.

wxWidgets byly implementovány v jazyce C++. Mohou být využity širokou škálou programovacích jazyků, jako například C++, Python, Pearl, Java a jiné. Jsou šířeny pod licencí L-GPL, s výjimkou, že zkompilovanou knihovnu lze linkovat bez omezení. wxWidgets jsou tedy volně dostupné, stejně dostupná je i poměrně rozsáhlá dokumentace a v roce 2006 byla vydána i kniha věnovaná pouze jim [3]. Poslední stabilní verze je verze 2.8.

Možnosti wxWidgets

- Obsahuje základní prvky pro práci s GUI jako jsou tlačítka, sizery, dialogová okna; dále obsahuje i nestandardní prvky, například tabulky, kalendáře a jiné. Je implementováno velké množství funkcí pro dialogová okna, umožňující jejich specializaci. Mezi klasické dialogy patří (wxFileDialog, wxDirDialog, wxMessageBox a wxColourDialog.

- Zprávy jsou zpracovávány přes tzv. události. Ovládací prvky jsou odvozeny od třídy wxEventHandler a to umožňuje řešit jejich obsluhu přes tabulku událostí, které lze dokonce řetězit.
- Umožňují zapisování a následné čtení do běžných grafických formátů (BMP, JPG, GIF, PNG a další) a využívání ikon.
- Podporují různé SQL databáze, obsahují totiž sadu ODBC tříd.
- Usnadňují programování síťových aplikací díky implementovaným socketům, vláknům, FTP a HTTP protokolu.

5.2 Návrh aplikace

Teoretický návrh aplikace byl zpracován jako Semestrální projekt a tato bakalářská práce z něj vychází. Jedná se o jednoduchý grafický editor se speciálně implementovanými funkcemi. Základní rozdělení aplikace je do tří celků: pro vyplňování rastrových oblastí, vektorových oblastí a ořezávání polygonů. Podle těchto tří skupin je také mírně odlišné prostředí editoru, který součástí aplikace, a jeho možnosti.

5.3 Implementace rozhraní

5.3.1 Inicializace programu

O prvotní inicializaci programu se stará třída `FillAndClipApp::OnInit()`. Je vytvořena základní třída aplikace `MyFrame{}`, která vytvoří dialogorvé okno aplikace, její menu a funkce pro jeho obsluhu. Funkce `MyFrame::CreateControls()` vytvoří záložky a `sizery` pro třídy, které připraví ovládací prvky každé ze tří výše zmíněných skupin. `Sizery` jsou objekty umožňující správu ovládacích prvků na ploše okna.

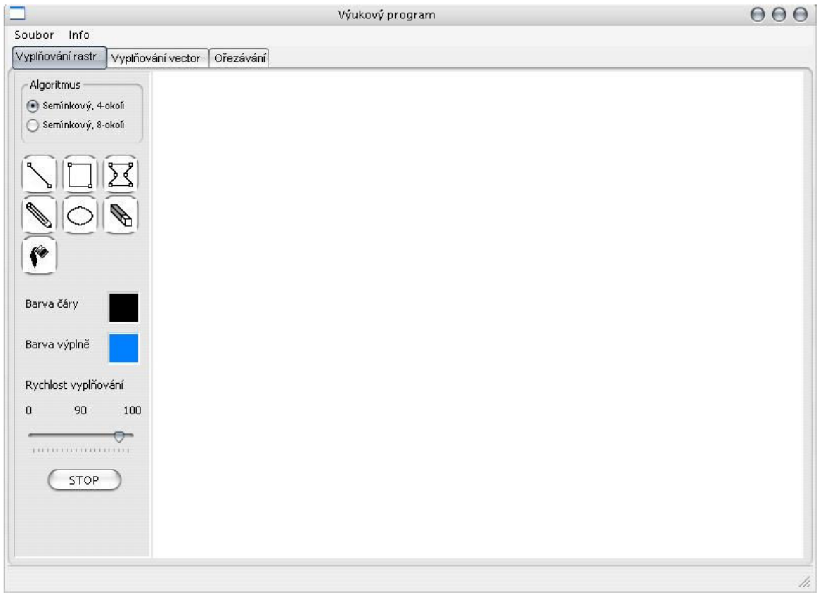
Třídy s názvem končící na `-Page` při inicializaci vytvoří levý panel se seznamem implementovaných algoritmů, ovládací tlačítka a posuvník pro nastavení rychlosti vykreslování a funkce k nim náležící.

Třídy s názvem končícím na `-Canvas` vytvoří kreslící plátno a bitmapu, do které se bude zapisovat.

5.3.2 Uživatelský vstup

Celé ovládání programu je vytvořeno tak, aby k jeho ovládání stačila uživateli převážně pouze myš. Funkce grafické knihovny umí události vznikající jejím použitím (stisknutí tlačítka, pohyb myši) zpracovávat. Aplikace reaguje na stisknutí levého či pravého tlačítka myši, například při stisknutí

levého tlačítka vznikne událost `wxEVT_LEFT_DOWN` a funkce `OnLeftDown(wxMouseEvent& event)` vykoná kód. Vstup z klávesnice je využíván pouze při zadávání jména souboru při ukládání vytvořeného obrázku.



Obrázek 9 – Aplikace po inicializaci

5.4 Implementace jednotlivých metod

Abychom měli nějaké plochy či polygony k vyplňování, musíme si je nejprve vytvořit. O jejich samotné vykreslování se téměř nemusíme starat, protože knihovna wxWidgets obsahuje prostředky, které toto řeší snadno a efektivně. Pouze kreslíme-li polygon pro algoritmy využívající vektorový zápis oblasti, je nutné si ukládat jeho vrcholy, protože jsou nezbytné k další práci s ním.

V části aplikace zabývající se vyplňováním si uživatel vybere algoritmus, jehož postup chce znázornit, a použije nástroj Plechovka. Dle vybraného algoritmu je vytvořena třída s jeho implementací. Aby bylo možné názorně ukázat, jak se jednotlivé algoritmy liší, je v panelu nástrojů umístěn posuvník rychlosti vyplňování. Ten využívá metodu `OnTimer(wxTimerEvent &event)`, která každou jednotku času, která je nastavena, vykoná jeden krok. Je-li posuvník na maximu, znamená to, že není nastavena žádná jednotka a oblast je vyplněna v reálném čase bez zpomalení. Dalším výukovým doplňkem je například zobrazování aktivních průsečíků u řádkového a inverzního vyplňování, zobrazování právě procházeného pixelu u Pinedova algoritmu a jiné.

V části zabývající se ořezáváním si uživatel také nejprve vybere algoritmus, který chce použít, a stiskne tlačítko Oříznout. Dle vybraného algoritmu je volána funkce obsahující jeho implementaci. Pro názornost je v pozadí obrázku po oříznutí znázorněn původní polygon i ořezové okno. U metody Weiler-Atherton dojde také k označení všech použitých bodů a vypsání obsahů všech seznamů.

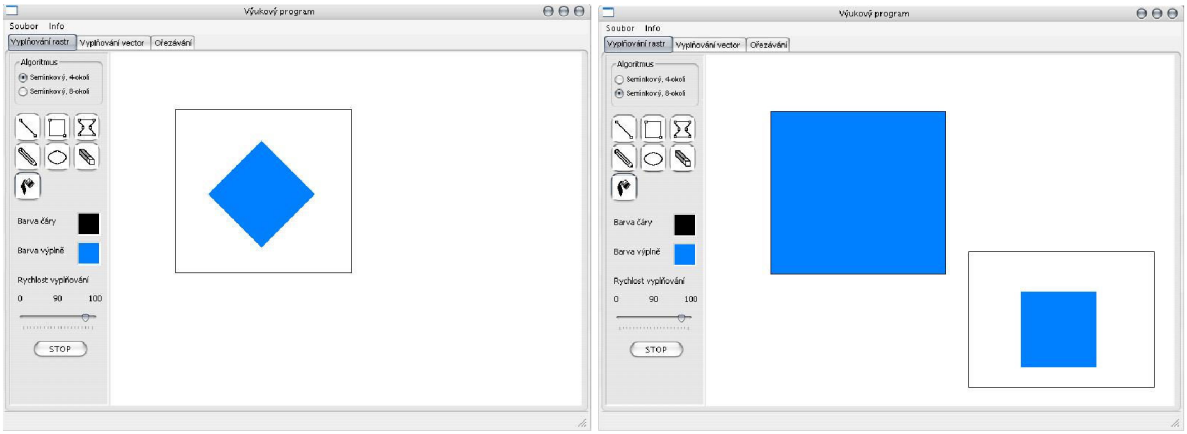
5.4.1 Semínkové vyplňování, 4-okolí

Při inicializaci třídy `Seed4fill{}` získáme barvu pixelu, který byl myší označen. Je-li barva totožná s barvou výplně, vyplňování končí. Pokud je ale barvy jiné, je vložen do seznamu `uncoloured`, který obsahuje ty body, které čekají na obarvení.

V každém dalším kroku časovače je spuštěna funkce `Seed4fill::Step()`. Nejprve vybarví všechny body ze seznamu `uncoloured`. Vytvoří druhý seznam `to_be_coloured`, do kterého se budou vkládat další body k obarvení. Ještě jednou projde seznam `uncoloured` a zjišťuje, jak jsou na tom čtyři sousední body těchto pixelů. Má-li bod jinou barvu než tu, kterou vyplňujeme, je přidán do seznamu `to_be_coloured`. Tak ale dochází k několikanásobnému vkládání téhož bodu. Proto pro snížení paměťové náročnosti jsou vkládány pouze ty body, které již nejsou v seznamu obsaženy. Nakonec se obsahy obou seznamů prohodí, tzn. obsah seznamu `uncoloured` je zahozen a v příštím kroku časovače se pokračuje s obsahem seznamu `to_be_coloured` v seznamu `uncoloured`. Vyplňování končí v okamžiku, je-li tento seznam prázdný.

5.4.2 Semínkové vyplňování, 8-okolí

Implementace tohoto algoritmu je téměř stejná jako u předchozího, liší se pouze v tom, že jsou při druhém průchodu seznamu `uncoloured` testovány i body na úhlopříčku.

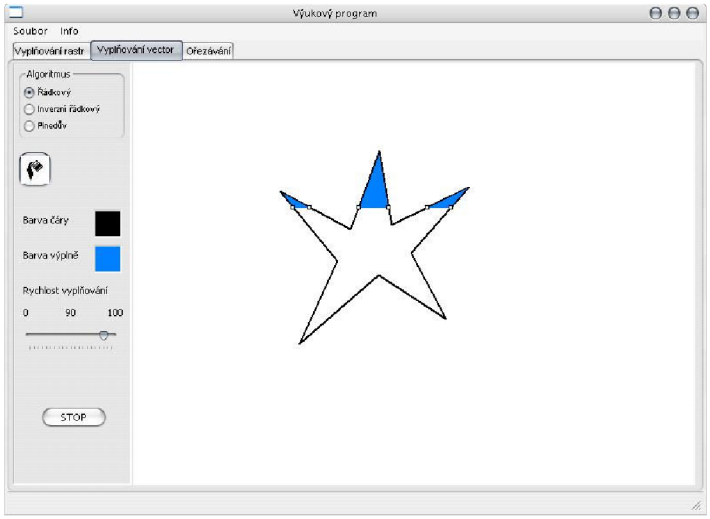


Obrázek 10 – Ukázka Semínkového vyplňování se 4 i 8 okolím

5.4.3 Řádkové vyplňování

Třída `Linefill{}` má oproti `Seed4fill{}` jednu zvláštnost: pracuje totiž se dvěmi bitmapami. Jedna je určená pro zakreslování výplně polygonu, druhá slouží pouze k edukačním účelům. V tomto případě jsou do ní při každém kroku pro názornost zakreslovány průsečíky právě vyplňovaného řádku s hranami polygonu. Během inicializace této třídy dojde pouze k nastavení hodnot x_{min}, x_{max}, y_{min} a y_{max} dle daného polygonu.

Každým krokem časovače je spuštěna funkce `Step()`. Vždy pro právě procházenou řádku jsou vypočítány průsečíky s hranami, vloženy do seznamu a seřazeny podle souřadnice x. Pak jsou vybarveny pixely ležící mezi sudým a lichým průsečíkem. Toto je zakreslováno do výsledné bitmapy, která se zkopíruje, a do kopie jsou zakreslovány ony průsečíky. V posledním kroku časovače je na obrazovku vykreslena výsledná bitmapa, nehrozí tak že by nám zůstaly viditelné vykreslované průsečíky. Vykreslení výsledné bitmapy na obrazovku se provede i po stisknutí tlačítka Stop.



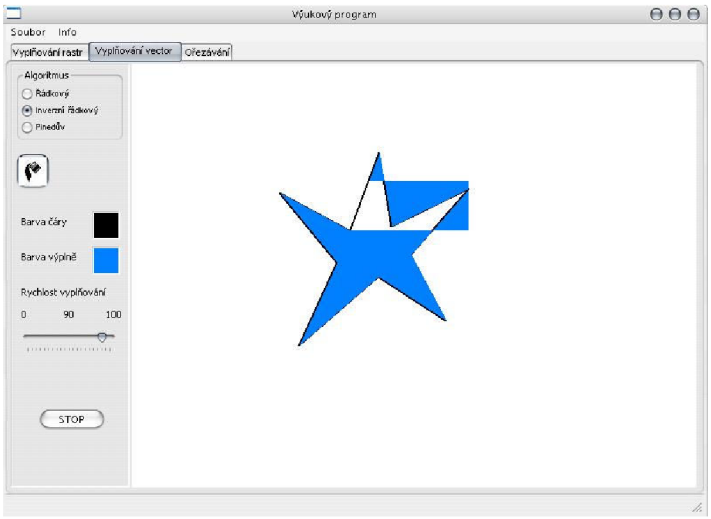
Obrázek 11 – Ukázka Řádkového vyplňování

5.4.4 Inverzní řádkové vyplňování

Inicializace třídy `InvFill{}` probíhá stejně jako inicializace předchozí třídy, navíc je však ještě akolována bitová maska a všechny její prvky jsou nastaveny na hodnotu 0.

Hrany polygonu jsou zpracovávány postupně, funkce `Step()` se v každém kroku zabývá jejím jedním řádkem. Z koncových bodů je vypočítána *x-ová* počáteční souřadnice a od ní je v cyklu až do bodu se souřadnicí x_{max} procházena bitová maska. Pokud je pro právě zpracovávaný bod nastavena maska na hodnotu 0, obarvíme jej barvou výplně. Pokud je má maska hodnotu 1, znamená to, že bod byl ve dřívějším průchodu obarven, a měla by se vrátit barva původního pozadí. Tento problém je řešen tak, že máme k dispozici dvě bitmapy. Do jedné se vykresluje vybarvovaný polygon a ve druhé máme uložené původní pozadí. Dle hodnoty masky je tedy buď pixel obarven barvou výplně, nebo je přes ukazatel do původní bitmapy zjištěna barva pixelu a ta použita k vybarvení.

Poté, co jsou tímto způsobem zpracovány všechny hrany, je bitmapa s původním pozadím přepsána bitmapou, do které se zakreslovala výplň polygonu. Stejná akce se provede i po stisknutí tlačítka Stop.

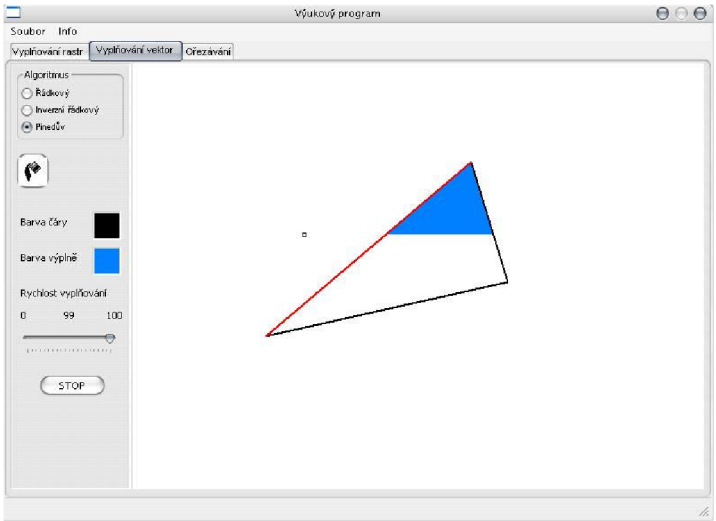


Obrázek 12 – Ukázka Inverzního řádkového vyplňování

5.4.5 Pinedův algoritmus

Při inicializaci třídy `PinedaFill{}` dojde také k určení maximálních a minimálních souřadnic polygonu, stejně jako u předchozích metod. Navíc se zde ale počítá jeho polarita. Zadáme-li totiž body polygonu a následně s jedním nebo několika body pohneme tak, že se změní jeho orientace (např. byly-li body původně zadány ve směru hodinových ručiček, změní se orientace do protisměru), může dojít k záměně znaménka hranové funkce a následnému špatnému vykreslení.

Funkce `Step()` prochází oblast kolem polygonu metodou Minimax, tedy od minimálních do maximálních souřadnic. Pomocí funkce `LeziUvnitr()` testuje, zda bod je uvnitř nebo vně polygonu. Je-li bod uvnitř, je obarven, je-li vně, je červeně vyznačena ta hrana, díky níž je hranová funkce záporná.



Obrázek 13 – Ukázka vyplňování Pinedovým algoritmem

5.4.6 Sutherland-Hodgman

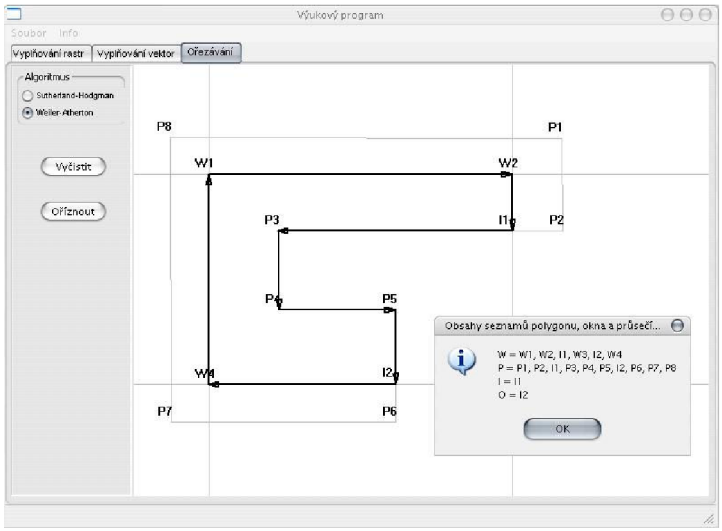
Koncepce je u tohoto algoritmu trochu jiná, než tomu bylo u předchozích souborů. Není zde vytvořena celá nová třída, ale pouze funkce `SHclipping()`, která po svém zavolání volá čtyřikrát funkci `SHstep()`, vždy pro jednu hranu ořezového okna.

Díky funkcím vytvořené struktury `Plane{}`, jako je výpočet vzdálenosti bodu od roviny `Vzdalenost()`, vzájemná poloha bodu a roviny `Pozice()` a výpočet souřadnice průsečíku `Prusecik()`, nemá funkce `SHstep()` příliš mnoho práce. Pouze otestuje polohu dvou bodů a dle toho, zda jsou oba uvnitř, venku nebo je některý uvnitř, plní výstupní polygon body.

5.4.7 Weiler –Atherton

Koncepce zpracování tohoto algoritmu je podobná jako u předchozího případu. Také není vytvářena nová třída, ale je volána pouze funkce `WAclipping()`, která pomocí dalších pomocných funkcí zařídí správné ořezání polygonu.

Aby bylo možné zajistit správný průběh algoritmu, využívá funkce `WAclipping()` více drobnějších funkcí pro práci s průsečíky polygonu a okna, i několik dalších menších tříd a jejich funkcí. Nejprve je polygon testován, zda nemá opačnou orientaci; obecně se předpokládá orientace bodů ve směru hodinových ručiček. Pak jsou naplněny seznamy polygonu, okna i průsečíků. Ošetřena je i varianta, kdy mají dva body s přímkami ohraničujícími okno dva průsečíky (např. je-li jeden vrchol vysoko nad horní hranou okna a druhý hodně vlevo od levé hrany okna). Pak je již pouze naplněn výstupní polygon vhodnými body a vykreslen. V závěru jsou formou informačního okna vypsány obsahy všech seznamů.



Obrázek 14 – Ořezávání polygonu algoritmem Weiler-Atherton

5.5 Struktura zdrojových kódů

Většina souborů existuje ve dvojici souborů s příponou .cpp a .h. Hlavičkové soubory obsahují definici konstant a proměnných dané třídy a výčet hlaviček funkcí, které tato třída volá a využívá. Soubory .cpp pak obsahují těla těchto funkcí.

Další skupinu souborů tvoří hlavičkové soubory jednotlivých vyplňovacích a ořezávacích metod, které však nemají svůj protějšek ve tvaru .cpp.

5.5.1 main.cpp

Tento soubor obsahuje funkce pro vytvoření samotné aplikace, jejích panelů, menu a funkce pro jejich obsluhu. Jeho úkolem je inicializovat program.

5.5.2 StdAfx.cpp, StdAfx.h

Hlavičkový soubor StdAfx.h obsahuje standardní systémové knihovny, které jsou často využívány, ale jejich obsah se nemění. Využití tohoto souboru je výhodné zvláště proto, že šetří čas při kompilaci a usnadňuje tak práci.

5.5.3 point2D.h

Hlavičkový soubor, který byl vytvořen pro definici struktury bodu. Obsahuje také definici operací plus a mínus nad touto strukturou a funkci, která vrátí vzdálenost dvou bodů.

5.5.4 ColorBox.cpp, ColorBox.h

Soubory pro práci s barevnou paletou. Zde jsou obsaženy funkce zajišťující vytvoření dialogového okna palety barev a navrácení uživatelem vybrané barvy čáry a výplně.

5.5.5 fillAlgorithm.h

Hlavičkový soubor s definicí obecné třídy, která je dále využívána všemi vyplňovacími algoritmy.

5.5.6 rasterFillPage.cpp, rasterFillPage.h

Soubory obsahující deklaraci a definici třídy a jejích funkcí, které jsou nutné pro vytvoření ovládacích prvků nástrojového panelu pro vyplňování rastrovými algoritmy, tedy Semínkovým se 4 a 8 okolím.

5.5.7 rasterCanvas.cpp, rasterCanvas.h

Dvojice souborů deklarující a definující funkce pro obsluhu ovládacích prvků nástrojového panelu, funkce pro vykreslování vybraných grafických objektů a funkce pro zpracování vstupu pomocí myši od uživatele. Jedná se zde převážně o správu událostí. V závislosti na tom, jaký vstup přijde od uživatele se volají jednotlivé funkce obsluhy.

5.5.8 vectorFillPage.cpp, vectorFillPage.h

Soubory obsahující deklaraci a definici třídy a jejích funkcí, které jsou nutné pro vytvoření ovládacích prvků nástrojového panelu pro vyplňování vektorovými algoritmy, tedy Řádkovým, Inverzním řádkovým a Pinedovým.

5.5.9 vectorCanvas.cpp, vectorCanvas.h

Dvojice souborů deklarující a definující funkce pro obsluhu ovládacích prvků nástrojového panelu, funkci pro vykreslení polygonu a funkce pro zpracování vstupu pomocí myši od uživatele. Jedná se zde také převážně o správu událostí.

5.5.10 clippingPage.cpp, clippingPage.h

Soubory obsahující deklaraci a definici třídy a jejích funkcí, které jsou nutné pro vytvoření ovládacích prvků nástrojového panelu pro ořezávání polygonů.

5.5.11 clippingCanvas.cpp, clippingCanvas.h

Dvojice souborů deklarující a definující funkce pro obsluhu ovládacích prvků nástrojového panelu, funkce pro vykreslování vybraných grafických objektů a funkce pro zpracování vstupu pomocí myši od uživatele. Dále obsahují funkce implementující ořezávací algoritmy Sutherland-Hodgman a Weiler-Atherton a jiné drobnější pomocné funkce.

5.5.12 seedFill.h, lineFill.h, inverseFill.h, pinedaFill.h

Všechny tyto knihovny mají obdobnou funkci. Obsahují třídy a jejich funkce implementující jednotlivé algoritmy vyplňování.

5.5.13 WAclip.h

Tato knihovna obsahuje třídu a její funkce umožňující práci s ořezovým oknem.

6 Závěr

Výsledkem této práce je jednoduchá aplikace, která by měla pomoci studentům snáze pochopit rozdíly mezi vyplňovacími a mezi ořezávacími algoritmy. Byla navržena tak, aby bylo předvedení algoritmů co nejnázornější i za cenu relativně pomalé rychlosti vykreslování.

Budoucí rozšíření aplikace vidím právě v optimalizaci kódů, přidáním možnosti vyplňování vzorem nebo šrafování či rozšíření ořezávacích algoritmů o algoritmy ořezávající speciálně přímky. Další drobnosti, které by mohly aplikaci v budoucnu vylepšit, jsou možnost nahrání vlastního obrázku do kreslící plochy a přidání položky do menu, která by obsahovala teoretické zpracování algoritmů.

Literatura

[1] Sochor, J., Žára, J., Beneš, B.: Algoritmy počítačové grafiky, Skriptum vysoké učení technické v Praze, Vydavatelství ČVUT, Praha, leden 1998.

[2] Kršek, P., Základy počítačové grafiky, prezentace přednášek, FIT VUT v Brně, 2006

[3] Smart, J., Hock, K., Csmor, S.: Cross-Platform GUI programming with wxWidgets, Prentice Hall - Professional Technical Reference, ISBN No: 0-13-147381-6