VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

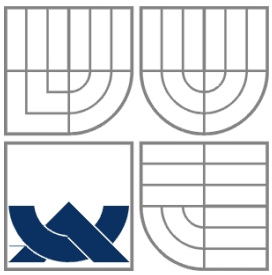
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DEMONSTRAČNÍ APLIKACE ALGORITMŮ VYPLŇOVÁNÍ UZAVŘENÝCH OBLASTÍ VE 2D

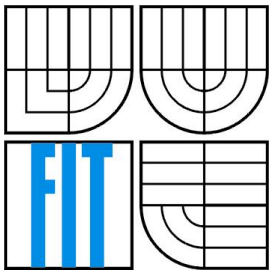
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE PAVEL HORT
AUTHOR

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

DEMONSTRAČNÍ APLIKACE ALGORITMŮ VYPLŇOVÁNÍ UZAVŘENÝCH OBLASTÍ VE 2D

DEMONSTRATION APPLICATION OF ALGORITHMS FOR CLOSED AREAS FILLING IN 2D

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

PAVEL HORT

VEDOUCÍ PRÁCE
SUPERVISOR

ING. JIŘÍ VENERA

BRNO 2008

Abstrakt

Tato práce se zabývá postupy použitými při vyplňování uzavřených oblastí ve 2D. V dokumentu jsou popsány algoritmy, které se k tomuto účelu nejčastěji využívají. Text popisuje obě hlavní skupiny těchto metod, tedy algoritmy vektorové i algoritmy rastrové. Jsou zde popsány klady a zápory jednotlivých vyplňovacích metod. Pro některé algoritmy jsou zde rovněž uvedeny optimalizace, nebo další možnosti jejich implementace. Další část tohoto textu se zabývá popisem implementace uvedeným metod s možností krokování jejich postupu. Závěr textu se věnuje popisu tříd, které jsou použité v aplikaci. Je zde také popsáno grafické uživatelské rozhraní.

Klíčová slova

počítačová grafika, Řádkové vyplňování, Inverzní řádkové vyplňování, Pinedův algoritmus, Semínkové vyplňování, Řádkové semínkové vyplňování

Abstract

This thesis describes methods used for filling closed areas in 2D. This text describes the most frequently used algorithms for this task. The first part of thesis describes both the main groups, vector algorithms and raster algorithms. Positives and negatives of these algorithms are described. For few of these algorithms is also mentioned how to optimize this algorithms or how to implement them using a different way. Next part of this text describes implementation of these algorithms for purpose of tracing them into. Last part of this text describes classes used in application. Also graphic user interface is described here.

Keywords

computer graphics, Scan-Line fill, inversive Scan-Line fill, Pined fill algorithm, Flood fill, Scan-Line Seed fill

Citace

Hort Pavel: Demonstrační aplikace algoritmů vyplňování uzavřených oblastí ve 2D. bakalářská práce, Brno, FIT VUT v Brně 2008.

Demonstrační aplikace algoritmů vyplňování uzavřených oblastí ve 2D

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Jiřího Venery
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Pavel Hort
12. května 2008

Obsah

1 Úvod

Cílem této práce je seznámit čtenáře s funkcí algoritmů využívaných pro vyplňování uzavřených oblastí ve 2D. Tyto algoritmy jsou používané při zobrazování v počítačové grafice. Počítačová grafika je oborem informatiky, který se zabývá zprostředkováním obrazové informace uživateli.

Tyto algoritmy rovněž poskytují několik způsobů vyplnění oblasti. Například jednou barvou, kombinací dvou barev, mezi nimiž je prolnutí, opakováním předem daného vzoru nebo vyplnění oblasti texturou. Oblast použití vyplňovacích algoritmů je poměrně rozsáhlá. Jejich nasazení je zřejmé ve většině způsobů zobrazování v počítačové grafice. Je tedy dobré vědět jak tyto algoritmy pracují a znát jejich výhody a nevýhody pro různá řešení.

Cílem těchto algoritmů je vyplnění nějaké oblasti ve 2D prostoru. Oblast je vždy uzavřená a její vnitřek lze vyplnit různým způsobem. Přitom není důležité, zda je oblast konvexní nebo ne. Tyto algoritmy rozdělujeme do dvou základních skupin a to sice podle způsobu reprezentace hranic vyplňované oblasti, kterou algoritmus používá. Jednu skupinu tvoří algoritmy pracující s rastrovou reprezentací hranice a druhou algoritmy, které pracují s hranicí reprezentovanou vektorově.

Vektorové oblasti bývají definovány posloupností bodů představujících mnohoúhelník, nebo posloupností geometricky popsaných čárových útvarů (přímky, úsečky, oblouky). Rastrové oblasti bývají určeny z obsahu obrazové paměti, nebo je pro určení hranice významná barva bodů, které tuto hranici tvoří popřípadě bodů tvořících oblast.

Náplní této práce je přiblížit funkci těchto algoritmů a jejich vlastností. Postupně se zaměřím na princip funkce vybraných rastrových i vektorových algoritmů. Na jejich klady, zápory a další vlastnosti jejich použití. Dále budou vysvětleny detaily implementace zvolených algoritmů. V poslední části práce bude popsáno uživatelské rozhraní, které demonstruje činnost implementovaných algoritmů a umožňuje uživateli projít celým procesem vyplnění oblasti krok po kroku.

2 Vektorové algoritmy

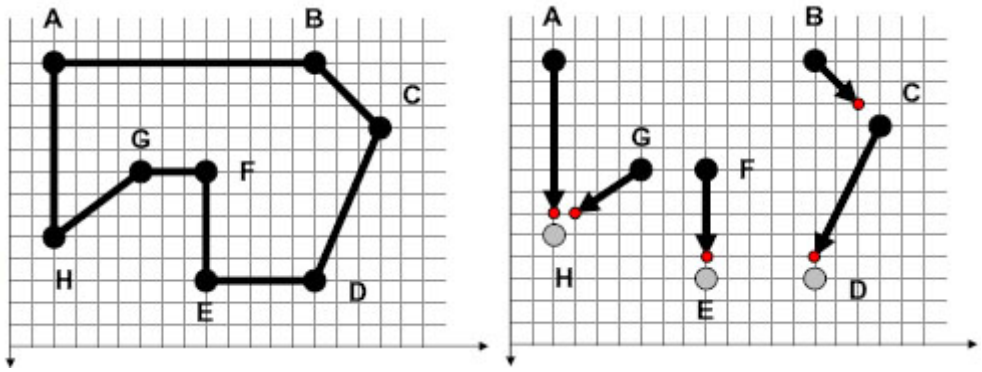
V této kapitole budou popsány principy a základní vlastnosti vektorových algoritmů, které tato aplikace demonstruje.

2.1 Řádkové vyplňování

Jde o základní algoritmus pro vyplňování obecných mnohoúhelníků. Vyplňovaná oblast Ω je popsána seznamem hraničních entit. Pokud se jedná o úsečky, postačí orientovaný seznam vrcholů těchto úseček.

Základní myšlenkou tohoto algoritmu je procházení dané oblasti Ω po řádcích výsledného rastru komplexně přes celou oblast. Úseky řádků, které jsou uvnitř oblasti, se vyplní barvou nebo vzorem. [5]

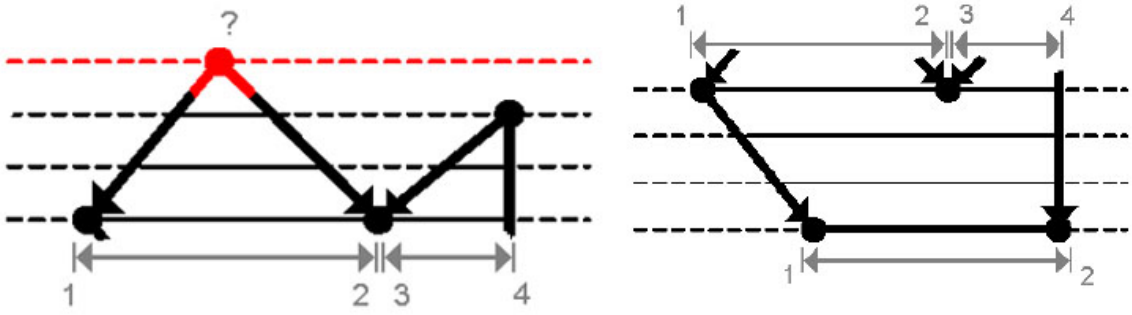
Tento algoritmus potřebuje předzpracování hranice, jelikož očekává sudý počet průsečíků na řádku. Při zpracování se také neberou v úvahu vodorovné hrany oblasti. Pro řešení problému lichého počtu průsečíků se nabízí dvě možnosti.



Obrázek 2.1: Hranice oblasti před úpravou a po úpravě

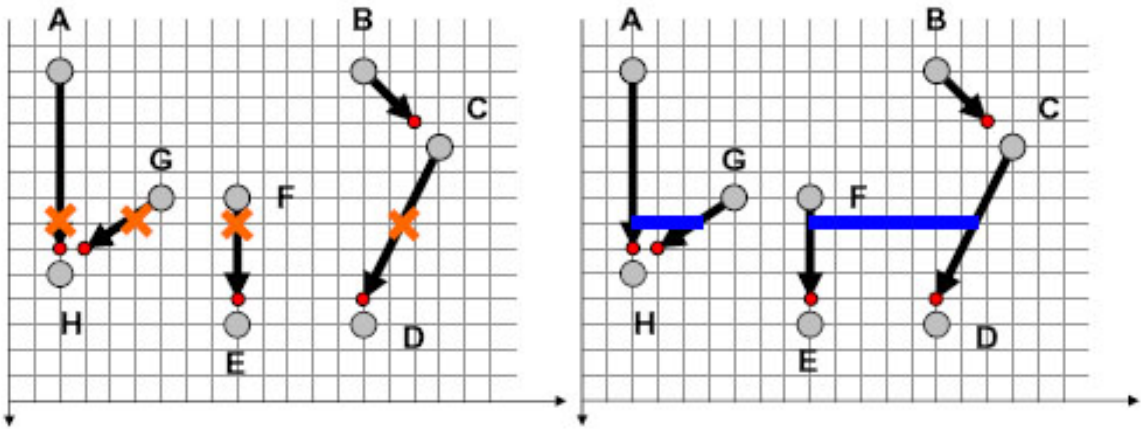
Zkrácení dolního okraje všech úseků hranice oblasti o jedna ve směru osy y. Vodorovné hrany se vypouští, což může zapříčinit zkrácení dolního okraje oblasti. Je proto nutné po ukončení činnosti algoritmu znovu vykreslit hranici oblasti. V rastru jsou koncové body hraničních úseček určené celými čísly, při různých sklonech hraniční přímky může dojít k tomu, že při zkrácení zdola o jedna nemusí vyjít koncová x-ová souřadnice koncového bodu celočíselná. Z tohoto důvodu není vhodné úsečky skutečně zkracovat, ale při výpočtu průsečíků vynechávat ty body, jež jsou koncovými body hraniční úsečky.

Druhou možností je analyzovat typy extrémů protínaných vrcholů během generování průsečíků. To však komplikuje a zpomaluje algoritmus.



Obrázek 2.2: Extrémy hranic oblasti

Algoritmus začíná svoji činnost předzpracováním hranice, následně zjistí minimální a maximální hodnotu v ose y a začne provádět vlastní vyplňování. Pro každý řádek rastru mezi maximální a minimální hodnotou v ose y vypočte průsečíky vyplňovaného řádku s hranicemi oblasti. Tyto průsečíky seřadí vzestupně podle osy x. Následuje vyplnění úseků řádku mezi lichými a sudými průsečíky. Po zpracování všech řádků ještě znovu vykreslí hranici oblasti.



Obrázek 2.3: Řádek oblasti vyplněný řádkovým algoritmem

Nevýhodou tohoto algoritmu je nutnost řazení průsečíků pro každý řádek, toto může algoritmus výrazně zpomalit. Další nevýhodou tohoto algoritmu je možnost zkrácení hranic oblasti a tedy nutnost překreslení hranic oblasti po ukončení funkce algoritmu.

Možnou optimalizací tohoto algoritmu je udržování seznamu aktivních hran oblasti. Nejprve jsou seřazeny všechny zkrácené a orientované hrany podle horní souřadnice osy y. Výsledkem je seřazení hran do tabulky hran, kde jedna položka tabulky odpovídá jednomu rozkladovému řádku a obsahuje seznam těch hran, jejichž horní bod má souřadnici y shodnou s tímto řádkem. Hrany jsou zde popsány záznamem, který obsahuje informace o tom do kolika řádků hrana zasahuje, aktuální souřadnice x průsečíku s rozkladovým řádkem a přírůstek x při přechodu na další řádek. Dále tato varianta využívá seznam aktivních hran, který obsahuje pouze hrany, které mají průsečík s právě

zpracovávaným rozkladovým řádkem. Hrany se v tomto seznamu udržují seřazené podle osy x. Tento seznam je využit pro řádkovou koherenci, kdy při přechodu na nový řádek zůstává uspořádání průsečíků ve většině případů stejné. Uspořádání je narušeno v případě protínajících se hran, nebo pokud je do seznamu přidána nová hrana z tabulky hran. Při přechodu na další řádek se provede aktualizace seznamu aktivních hran, kdy se do seznamu přidají nové hrany, nebo se odstraní hrany, které jsou již neaktivní a přičte se přírůstek v ose x (hranová koherence). [1]

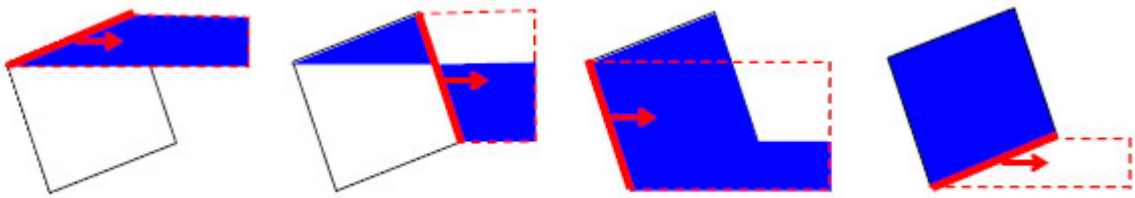
2.2 Inverzní řádkové vyplňování

Inverzní řádkové vyplňování je alternativou řádkového algoritmu. Odstraňuje nutnost třídit průsečíky pro každý řádek. Vyplňovaná oblast Ω je popsána rovněž seznamem hraničních entit, nebo orientovaným seznamem vrcholů pokud jde o úsečky.

Základní myšlenkou je zde procházení oblasti po jednotlivých hranách. Od každé hrany napravo je provedeno vyplnění s inverzí aktuálních hodnot bodů.

Pro oblast Ω se nalezne maximální souřadnice v ose x. Pro každou hranu e_i z oblasti Ω se získá maximální a minimální souřadnice v ose y. Pro každý řádek mezi nalezenými souřadnicemi v ose y (vodorovné hrany se vynechávají) nalezne průsečík řádku a aktuální hrany. Následně invertuje hodnoty bodů v řádku od průsečíku po maximální souřadnici v ose x. [5]

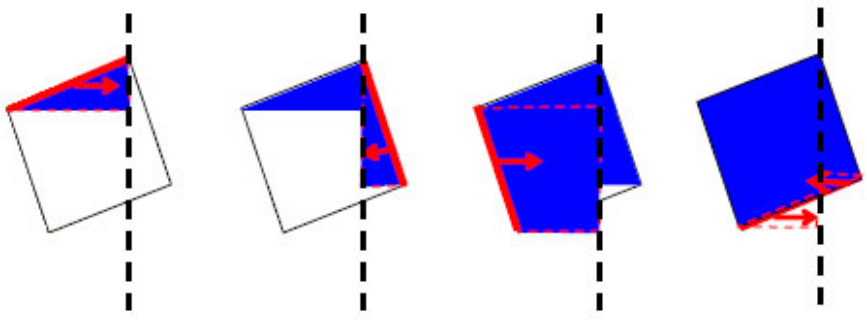
Inverzí hodnot bodů je myšleno přepnutí jejich hodnot z obarvených na neobarvené a naopak.



Obrázek 2.4: Postup inverzního vyplňování

Díky odstranění nutnosti třídit průsečíky řádku s hranicí oblasti dosahuje tento algoritmus lineární časovou náročnost v závislosti na počtu hran. Dochází ke zkreslení hranice oblasti, proto je nutné po ukončení vyplňování tuto hranici znovu vykreslit. Druhou nevýhodou toho algoritmu je, že ovlivňuje i body mimo vyplňovanou oblast a může tedy dojít k porušení obrazové informace. Proto je vhodnější provést vyplnění do pomocné paměti a až hotový výsledek zobrazit v obrazové paměti.

Plotová varianta inverzního vyplňování. Pracuje na stejném principu, ale vyplňování se neprovádí k okraji, nebo nalezenému maximu v ose x. Je zvolena svislice v některém z uzlů hranice, nejlépe uzel jehož x-ová souřadnice je nejblíže ke středu oblasti. Vyplňování se poté provádí směrem k této svislici.



Obrázek 2.5: Plotová varianta inverzního algoritmu

2.3 Pinedův algoritmus

Tento algoritmus je dalším zástupcem vektorových algoritmů. Díky svému principu pracuje pouze s konvexními mnohoúhelníky. Oblast Ω je opět popsána seznamem hraničních entit.

Základní myšlenkou algoritmu je rozdělení dané oblasti Ω na poloroviny jednotlivých hran e_i oblasti Ω. Všechny body, které leží na kladné straně všech polorovin hran e_i jsou potom uvnitř oblasti Ω a tedy jsou vyplněny. [5]

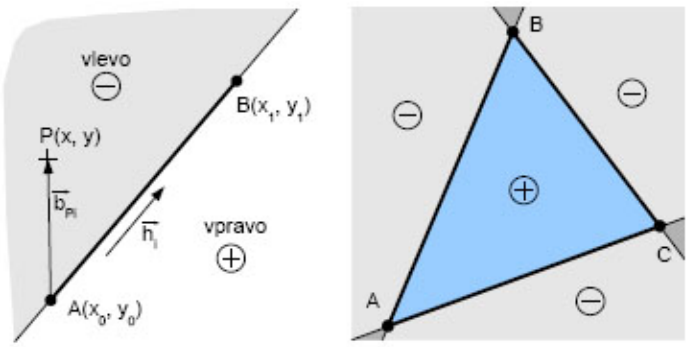
Základem algoritmu jsou hranové funkce. Jde o test polohy bodu z oblasti Ω vůči jedné její hraně e_i. Hodnotu hranové funkce $E_i(x, y)$ vypočteme podle vztahu (2.4) jako vektorový součin (2.3) vektoru $\vec{h}_1$ (2.1) hrany e_i a vektoru $\vec{b}_{pi}$ (2.2) z počátku hrany k testovacímu bodu.

$$\vec{h}_1 = (x_{i1} - x_{i0}, y_{i1} - y_{i0}) = (\Delta x_i, \Delta y_i) \tag{2.1}$$

$$\vec{b}_{pi} = (x - x_{i0}, y - y_{i0}) \tag{2.2}$$

$$E_i(x, y) = \vec{h}_i \times \vec{b}_{pi} \tag{2.3}$$

$$E_i(x, y) = (x - x_{i0})\Delta y_i - (y - y_{i0})\Delta x_i \tag{2.4}$$



Obrázek 2.6: Princip polorovin hran oblasti u Pinedova algoritmu

Je-li pro bod P(x, y) hodnota všech hranových funkcí $E_i(x,y) \geq 0$, bod leží uvnitř nebo na hranici oblasti Ω a je tedy vyplněn. Aby algoritmus nemusel počítat kompletně všechny hranové funkce pro každý bod, je možné určit hodnotu hranové funkce na základě hodnot odpovídajících hranových funkcí pro sousední bod (viz vztahy 2.8 a 2.9). [5]

$$E_i(x+1,y) = (x+1-x_{i0})\Delta y_i - (y-y_{i0})\Delta x_i \tag{2.5}$$

$$E_i(x+1,y) = (x-x_{i0})\Delta y_i + \Delta y_i - (y-y_{i0})\Delta x_i \tag{2.6}$$

$$E_i(x+1,y) = E_i(x,y) + \Delta y_i \tag{2.7}$$

$$E_i(x\pm 1,y) = E_i(x,y) \pm \Delta y_i \tag{2.8}$$

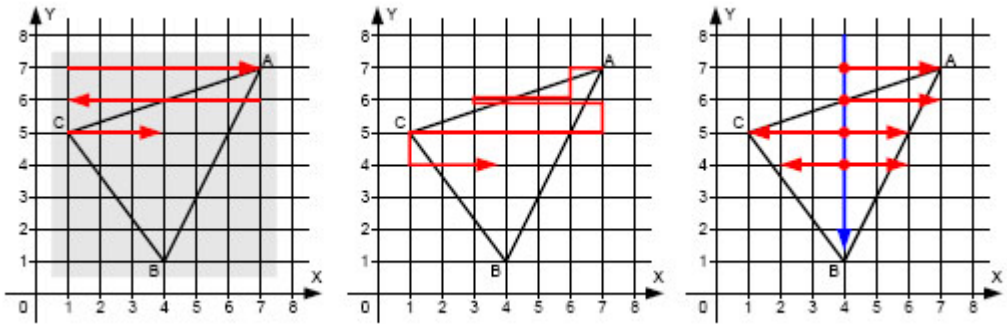
$$E_i(x,y\pm 1) = E_i(x,y) \pm \Delta x_i \tag{2.9}$$

Tento algoritmus se nejčastěji používá pro vyplňování trojúhelníků, jsou to nejjednodušší polygony a jsou vždy konvexní. Uplatňují se přitom tři základní strategie procházení oblastí Ω.

Procházení obdélníka opisujícího oblast po řádcích. Tento postup však není příliš efektivní, díky zbytečnému procházení oblasti, která je jistě prázdná.

Procházení řádků od maximálního vrcholu s obratem a přechodem na další řádek při opuštění oblasti. Tento postup je algoritmicky náročnější.

Procházení po řádcích podél svislého plotu, který spustíme z prostředního vrcholu trojúhelníku, nebo středem oblasti. Každou stranu plotu můžeme procházet odděleně což umožní paralelizaci provádění algoritmu.



Obrázek 2.7: Varianty procházení oblastí pro Pinedův algoritmus

Vyplnění trojúhelníku (velmi časté ve 3D zobrazování) je tak převedeno na testování hranových funkcí všech bodů v okolí oblasti Ω. Díky výpočtu hranové funkce na základě sousedních bodů, což znamená jednu operaci sčítaní, je algoritmus jednoduchý a snadno paralelizovatelný. Z těchto důvodů je výhodné ho implementovat přímo v hardware grafických jednotek. [5]

3 Rastrové algoritmy

V této kapitole budou popsány principy a základní vlastnosti rastrových algoritmů, které tato aplikace demonstruje.

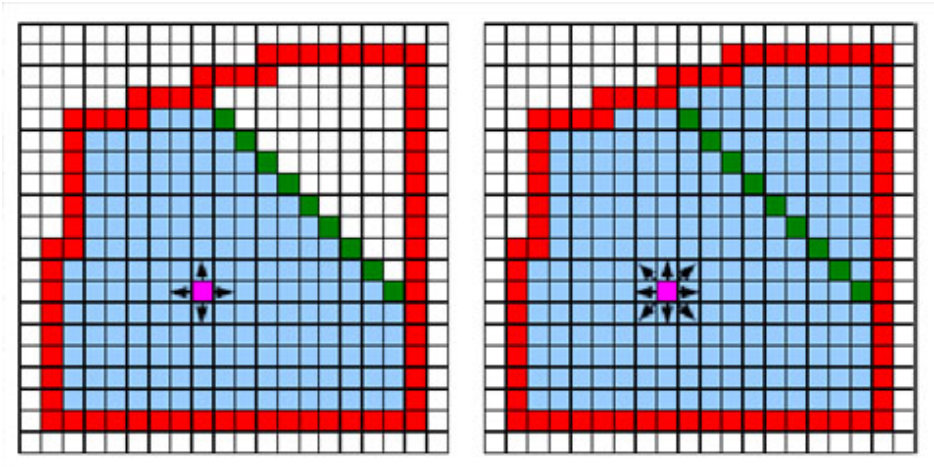
3.1 Semínkové vyplňování

Klasický algoritmus pro vyplňování rastrových oblastí. Vyplňovaná oblast Ω je popsaná v rastrové matici, nejčastěji přímo v obrazové paměti, množinou podobných bodů tvořících hranici oblasti, nebo vlastní oblast. Jelikož není k dispozici seznam hraničních entit, algoritmus neví co je uvnitř a co vně oblasti. Proto musí uživatel na začátku zadat startovní bod (semínko).

Základní myšlenkou algoritmu je obarvování sousedních bodů (semínek) od aktuálního (startovního) bodu. Obarvené body se stávají rekurzivně dalšími semínky. Body, které jsou hranicí oblasti nebo nemají vlastnosti oblasti se vyřazují. [5]

Možných implementací semínkového vyplňování je více. Podle způsobu reprezentace hranice oblasti a podle způsobu obarvování okolí.

Podle způsobu obarvování okolí od aktuálního semínka rozlišujeme dvě varianty vyplňování. Šíření do 4-okolí, kdy bereme v potaz 4 sousedy pro každé semínko, nebo do 8-okolí, kdy se bere v potaz 8 sousedů pro každé semínko.



Obrázek 3.1: Šíření do 4-okolí a do 8-okolí

Tyto dva druhy okolí vedou na dva druhy hranice. Pro 4-okolí je hranice uzavřená pokud její hraniční body sousedí po hraně. Pro 8-okolí je hranice uzavřená pokud její hraniční body sousedí po hranách i přes vrchol.

Podle způsobu definice oblasti nebo její hranice rozlišujeme následující čtyři varianty řízení vyplňování:

- Hraniční, kdy je oblast definována spojitou hranicí z bodů určité barvy. Testuje se barva vyplňovaných bodů, jestli není barvou hranice.
- Záplavové, kdy je oblast zadána množinou bodů se stejnou barvou. Testuje se barva bodu, jestli je barvou oblasti.
- Měkké, oblast je definována spojitou hranicí z bodů daného rozptylu barev. Testuje se barva bodu, jestli je barvou uvnitř daného rozptylu barev.
- Prahové, oblast je zadána spojitou množinou vnitřních bodů jejichž barva leží v daném rozmezí okolo barvy prvního semínka. Testuje se barva bodů jestli je v rozmezí barvy prvního semínka. [5]

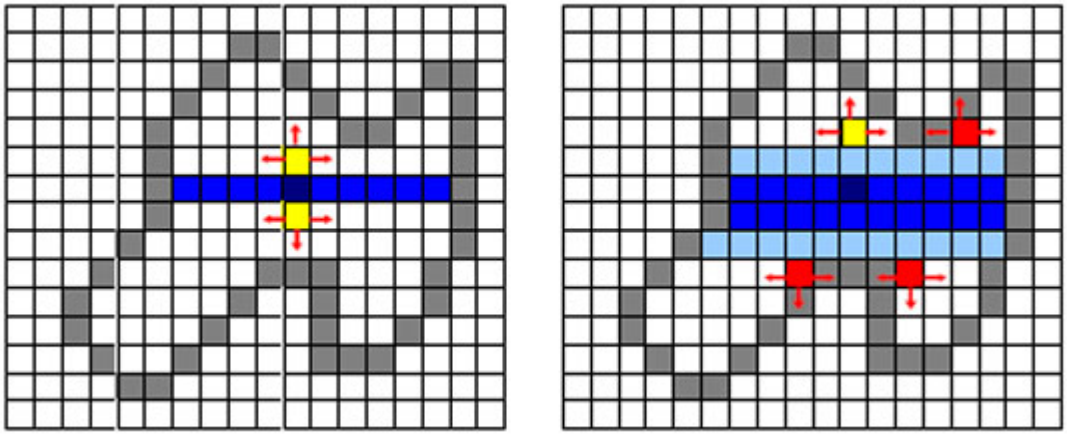
Pokud je algoritmus implementován rekurzivně hrozí, že dojde místo na systémovém zásobníku, je proto výhodnější tento algoritmus implementovat s vlastním zásobníkem nebo frontou a raději nepoužívat rekurzivní implementaci tohoto algoritmu.

Obecný postup algoritmu pro semínkové vyplnění rastrové oblasti Ω s počátečním bodem S[x,y] a vlastním zásobníkem je následující. Testování sousedů aktuálního bodu (4-okolí, nebo 8-okolí podle typu vyplňování), jestli jsou neobarvené a patří do oblasti Ω. Obarvení vyhovujících bodů a jejich uložení na zásobník. Získání nového aktivního bodu ze zásobníku. Algoritmus končí pokud je zásobník prázdný.

3.2 Řádkové semínkové vyplňování

Tento algoritmus je obdobou semínkového vyplňování a pracuje na stejném principu. Také se může šířit do 4-okolí, nebo 8-okolí. Rovněž řízení algoritmu může pracovat na totožném principu. Je opět nutné zadání počátečního bodu (semínka).

Základní myšlenou algoritmu je, vyplňování souvislých vodorovných úseků a prohledávání intervalů nad a pod těmito úseky. Každá vodorovná řada vnitřních bodů nad, nebo pod daným úsekem tvoří nový úsek, který je rekurzivně zpracován. [1]



Obrázek 3.2: postup vyplňování řádkového semínkového algoritmu

Obecný postup algoritmu je nalezení hranice x_L a x_R na řádku aktuálního bodu (semínka) [x, y] v jeho nejbližším okolí. Zakreslení úsečky [x_L, y]-[x_R, y]. Hledání souvislých úseků na vyšší a nižší úsečce [x_L, y ± 1]-[x_R, y ± 1]. Pro každý nalezený úsek je vložen jeden bod (semínko) do zásobníku. Algoritmus končí po vyprázdnění zásobníku.

Výhodou řádkového semínkového vyplňování je, že snižuje počet přístupů do obrazové paměti. Také výrazné snížení počtu bodů na zásobníku.

4 Úprava algoritmů pro aplikaci

Kvůli požadavku na možnost krokování algoritmu v aplikaci, není možné použít algoritmy v základní podobě. Je nutné je upravit tak, aby bylo umožněno přerušení provádění algoritmu s možností pokračování od místa kde k přerušení došlo. Rovněž nemá smysl algoritmus přerušovat pokud zatím nedošlo ke změně obrazové paměti, protože toto nebude mít žádný efekt pro pozorovatele. Jelikož všechny algoritmy, které tato aplikace demonstruje pracují iteračně, je vhodné algoritmus přerušovat před přechodem do další iterace (kroku) algoritmu. Toto je totiž ve většině případů okamžik, kdy mohlo dojít ke změně obrazové informace, kterou uživatel pozoruje. Rovněž jsou také všechny vnitřní a stavové proměnné algoritmu vypočteny a je tedy možné je zálohovat pro účel obnovy stavu algoritmu pro další krok výpočtu. Způsob tohoto zálohování se liší u vektorových a rastrových algoritmů díky odlišnostem ve způsobu jejich činnosti. Všechny algoritmy je možné spustit ve dvou režimech. V prvním se provede vyplnění oblasti bez přerušování a uživateli je zobrazen výsledek algoritmu po vyplnění oblasti. Při opakovaném spouštění ve druhém režimu se provede vyplnění po krocích. Toto řešení umožňuje volný přistup k vnitřnímu stavu algoritmu, proto je možné uživateli zobrazit tento vnitřní stav, pro lepší pochopení funkce algoritmu.

Tato varianta ovšem není jedinou možností řešení tohoto problému. Je zde také například možnost použít pro běh vyplňovacího algoritmu jiné programové vlákno, a toto pozastavovat v průběhu vykonávání algoritmu. Při použití tohoto řešení by algoritmy mohly zůstat ve své základní podobě, bylo by však nutné řešit přístup k proměnným algoritmu běžícího v jiném programovém vlákně.

4.1 Implementace vektorových algoritmů

V této kapitole budou popsány varianty vektorových algoritmů, které jsou implementovány v aplikaci. Bude zde rovněž uvedeno jaké pomocné funkce algoritmy využívají pro svoji činnost.

4.1.1 Řádkové vyplňování

Algoritmus řádkového vyplňování využívá pro svoji činnost pomocných funkcí například pro nalezení průsečíků s řádkem a jejich setřídění. Jednou z nich je funkce PrepareBorder, která připraví hranici oblasti pro další práci algoritmu. Na začátku tohoto algoritmu je provedeno zjištění minimálních a maximálních hodnot v ose y, mezi kterými se bude následující vyplňování provádět. Toto testování probíhá nad již připravenou hranicí. Další funkcí kterou algoritmus využívá je funkce GetCross, která pro aktuálně zpracovávaný řádek zjistí průsečíky. Tyto je nutné následně setřídit podle hodnoty v ose x. Poté jsou obarveny úseky mezi lichými a sudými průsečíky. V tomto místě je vloženo přerušení činnosti algoritmu a pokud jej uživatel spouští v režimu pro krokování, je

provádění algoritmu ukončeno. Předpřipravená hranice je již uložená ze začátku činnosti algoritmu, zálohuje se aktuální pozice řádku, na kterém se bude pokračovat. Při opětovném spuštění algoritmu v režimu krokování se tato pozice obnoví a pokračuje se od této pozice. Po ukončení jednoho kroku algoritmu je uživateli zobrazen aktuální stav vyplňované oblasti a vnitřních proměnných algoritmu.

4.1.2 Inverzní řádkové vyplňování

Algoritmus inverzního řádkového vyplňování využívá podpůrné funkce stejně jako algoritmus řádkového vyplňování. Před začátkem provádění algoritmu je provedena úprava hranice funkcí PrepareBorder. Dále se zjistí maximální souřadnice v ose x. Poté se pro každou hranu e_i z upravené množiny hran zjistí minimální a maximální souřadnice v ose y. Dále algoritmus provede zjištění průsečíků hrany s řádky mezi minimální a maximální souřadnicí v ose y pomocí funkce GetLineCross. Tyto průsečíky se však nemusejí dále třídit. Je provedena změna obarvení bodů od průsečíku vpravo po maximální souřadnici v ose x. Po provedení inverze pro všechny průsečíky je vloženo přerušení algoritmu, které se provede v případě , že je algoritmus volán v režimu krokování. Zálohují se pozice hrany v seznamu hran, kde bude algoritmu pokračovat. Při volání algoritmu v režimu krokování se obnoví pozice v seznamu hran a zpracuje se aktuální hrana. Po ukončení jednoho kroku algoritmu je uživateli zobrazen aktuální stav vyplňované oblasti.

4.1.3 Pinedův algoritmus

Pinedův algoritmus je v aplikaci implementován ve variantě která prochází plochu obdélníka který opisuje zadaný trojúhelník. Na začátku algoritmu jsou zjištěny hrany tohoto obdélníka a jsou vypočítány hranové funkce. Oblastí se prochází po řádcích s připočítáváním přírůstků hranových funkcí při přechodu na sousední bod. Přerušení algoritmu je vloženo po provedení testu zda aktuální bod náleží do vyplňované oblasti a jeho případném vyplnění. Algoritmus uchovává pro potřeby krokování hodnoty hranových funkcí a informaci o pozici, na které se uvnitř oblasti nachází. V případě, kdy je algoritmus volán způsobem pro krokování je provedena obnova stavu těchto hodnot a následně je provedena iterace algoritmu a nové uložení stavových hodnot. Po opuštění algoritmu po každém z kroků výpočtu je uživateli zobrazen aktuální bod, v kterém se algoritmus nachází, stav oblasti a také ohodnocení hranových funkcí pro aktuální bod.

4.2 Implementace rastrových algoritmů

Pro rastrové algoritmy implementované v této aplikaci je tento problém jednodušší, jelikož není nutné uchovávat vnitřní stav algoritmu, ale postačuje zálohovat stav zásobníku bodů, které algoritmus bude zpracovávat.

4.2.1 Semínkové vyplňování

Semínkové vyplňování je v této aplikaci implementováno pro šíření do 4-okolí i do 8-okolí. Tato volba se uživateli naskytne při výběru tohoto algoritmu pro demonstraci. Hranice je reprezentována hraniční formou s jistou odlišností v uložení bodů, která bude popsána v následující kapitole. V jiném směru je algoritmus oproti základní variantě prakticky nezměněn. Pouze při spuštění v režimu pro krokování se provede kontrola okolí aktuálního bodu vybraného ze zásobníku. Provede se uložení nevyřazených bodů na zásobník a na konci iterace se algoritmus ukončí. Po ukončení kroku algoritmu je uživateli zobrazen stav zásobníku a aktuální stav vyplňované oblasti.

4.2.2 Řádkové semínkové vyplňování

Řádkové semínkové vyplňování je implementováno v podstatě v základní podobě. Šíření bylo zvoleno pouze pro 4-okolí. Hranice je reprezentována hraniční formou se stejnou změnou jako u semínkového vyplňování. Pro každý bod vybraný ze zásobníku se provede nalezení levého a pravého průsečíku s hranicí oblasti. Řádek mezi těmito body je vyplněn a je provedeno nalezení spojitých úseků nad a pod právě obarveným řádkem. Z každé nalezené spojité oblasti, pokud již není obarvená, je uložen jeden bod na zásobník. Pokud je algoritmus spuštěn pro krokování, je poté provádění algoritmu ukončeno. Uživateli je po každém kroku zobrazen aktuální stav bodů na zásobníku a stav vyplňované oblasti.

5 Hierarchie tříd

Z povahy implementačního jazyka vyplývá rozdělení aplikace do tříd. Hlavní třídou aplikace je třída Mainform, která definuje vzhled uživatelského rozhraní a v jeho událostech používá i funkčnost dalších tříd. Například funkce třídy Imaging, pro zobrazování grafických informací. Třída Algorithm obsahuje algoritmy vyplňovaní, které využívají třídu Data jako datovou vrstvu pro svoji práci. Třídy Loader a XmlWorker udržují informativní část uživatelského rozhraní v aktuální podobě. Schéma hierarchie tříd se nachází na obrázku v příloze 1. V této kapitole bude přiblížen význam těchto tříd pro aplikaci.

5.1 Třída Mainform

Třída Mainform je třída hlavního formuláře aplikace. Definuje vzhled tohoto formuláře (uživatelského rozhraní) a obsahuje funkce pro obsluhu událostí vyvolaných uživatelem, jako například přepnutí aktuálně demonstrovaného algoritmu. Události této třídy volají a používají ke své práci funkčnost poskytovanou ostatními třídami. Události tlačítek „Krok“ a „Vyplň“ volají příslušným způsobem funkce třídy algorithm, kde jsou implementovány algoritmy pro vyplňování. Následně využívá funkcí třídy Imaging pro zobrazení výsledků práce těchto algoritmů. Kreslící plátno není nutné nijak aktualizovat díky přetížení metody Paint pro tuto komponentu. Stačí tedy pouze změnit stav bodů v paměti a plátno prohlásit za neaktuální pomocí funkce Invalidate. Systém si pak vynutí překreslení plátna.

5.2 Třída Data

Třída Data poskytuje datovou vrstvu aplikace. Instance této třídy v sobě uchovává seznamy bodů. Seznam bodů výplně je oddělen od seznamu pro hranici. Toto oddělení umožňuje jednoduše smazat výplň oblasti, pokud se uživatel rozhodne demonstraci přerušit. Hranice je v této třídě uložena dvakrát. Jednou v rastrové reprezentaci pro rastrové algoritmy a zobrazení hranice na kreslící ploše. Druhé uložení hranice ji reprezentuje vektorově pro použití ve vektorových algoritmech. Instance této třídy obsahuje rovněž atributy reprezentující vnitřní stav právě demonstrovaného algoritmu. Tyto atributy jsou implementovány jako public (veřejné) a jsou tudíž přístupné i ostatním třídám. Je tedy možné do přerušené metody pro vyplňování znovu vstoupit a pokračovat od místa, kde byla tato metoda přerušena. Toto řešení také umožňuje zobrazovat aktuální stav algoritmů a vykreslit aktuální vyplňovanou oblast a její výplň.

5.3 Třída Algorithm

Tato třída obsahuje implementace algoritmů, které jsou napsané jako static funkce a používají referenci na instanci datové vrstvy. Reference je využita pro uložení vnitřních proměnných, za účelem jejich obnovení. Je zde také implementována funkce která zjišťuje, jestli se bod nenachází za hranicí kreslícího plátna. V této třídě se rovněž nachází podpůrné funkce pro metody řádkového a inverzního řádkového vyplňování.

5.4 Třída Imaging

Třída Imaging obsahuje funkci LineGrid. Metoda Paint kreslícího plátna využívá této funkce, pro vykreslení mřížky na plátno a to podle zvolené velikosti přiblížení. Dále je zde funkce DrawPoints, která slouží pro vykreslení bodů do plátna. Tato je používána stejnou metodou, pro zobrazení bodů hranice a výplně. Funkce IsBorderClosed zjišťuje jestli je hranice oblasti již uzavřená, tuto funkci aplikace využívá pokud uživatel zadává vlastní hranici přímo do obrazového rastru. RasterizeLine je poslední funkce této třídy, tato provádí rasterizaci přímky na základě Bresenhamova algoritmu. Návratovou hodnotou funkce je seznam bodů přímky.

5.5 Třída Loader

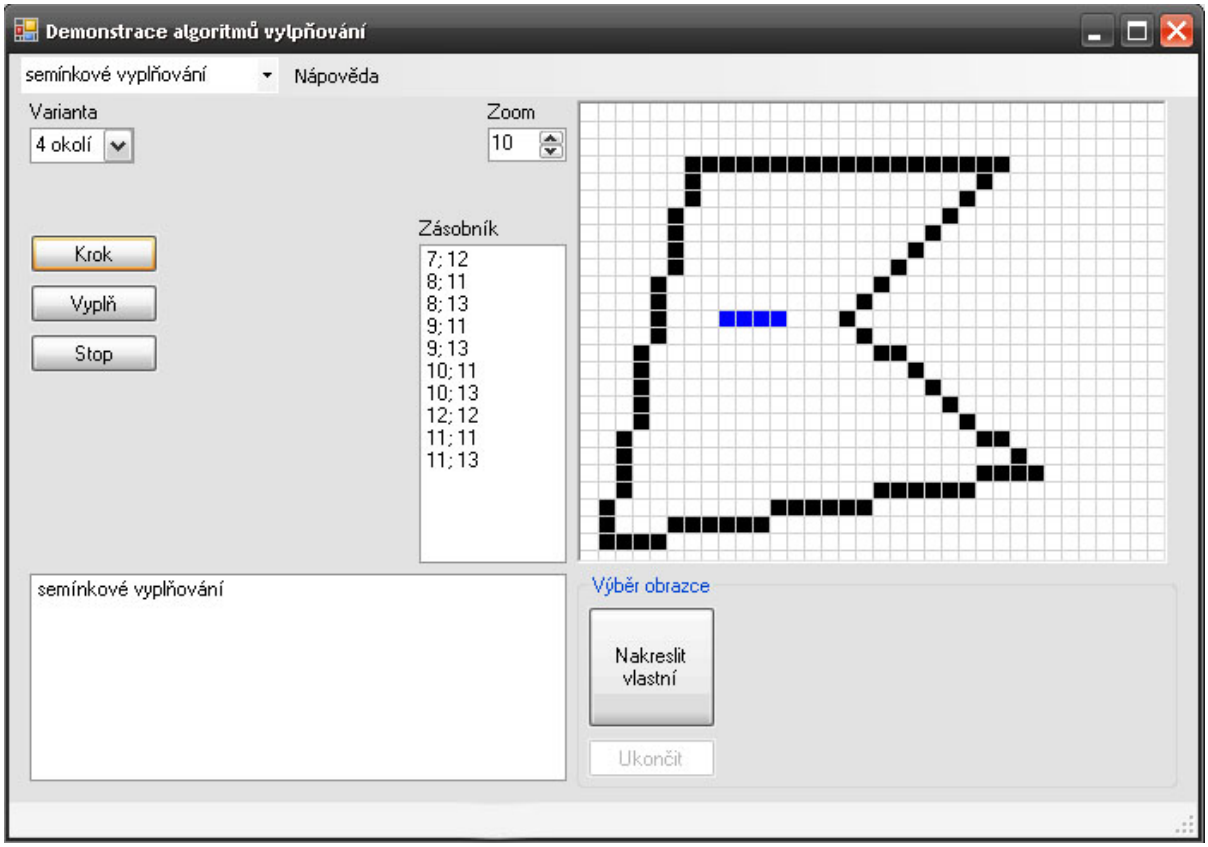
Funkce této třídy nahrávají, na základě zvoleného vyplňovacího algoritmu, který se má aktuálně demonstrovat, popis algoritmu a uložené předdefinované oblasti. LoadAlg je funkce, která je pro tento účel volána událostí uživatelského rozhraní. Popis a předdefinované vlastnosti jsou uloženy v XML souboru. Pro načtení informací z XML souboru je využívána funkčnost třídy XmlWorker.

5.6 Třída XmlWorker

Tato třída poskytuje prostředky pro získání požadovaných informací z XML souboru. Nachází se zde funkce LoadXml, která nahrává XML soubor ze zadané cesty. Metodou, kterou využívá třída Loader je, LoadDescription. Tato funkce nahrává popis a uložené hranice pro určitý algoritmus zadaný parametrem.

6 Uživatelské rozhraní

Uživatelské rozhraní aplikace je tvořeno, klasickým formulářem, který obsahuje prvky několika skupin.



Obrázek 6.1: Ukázka uživatelského rozhraní

Hlavním prvkem uživatelského rozhraní (dále GUI) je kreslící plátno, na kterém je demonstrováno vlastní vyplňování. Lze u něj zvolit zvětšení, se kterým se budou zobrazovat body v rastru. Pod kreslícím plátnem se nachází oblast pro volbu obrazce, na kterém se bude vyplňování demonstrovat. Je zde i možnost nakreslení vlastního obrazce. Tato možnost je k dispozici pod tlačítkem „Nakreslit vlastní“, které aktivuje uživatelské zadání oblasti, případně je resetuje při opětovném stisku. Zadávání oblasti se provádí pomocí myši přímo do kreslícího plátna. Druhé tlačítko „Ukončit“ ukončuje zadávání uživatelské oblasti. Ovládání vyplňování se provádí pomocí tlačítek „Vyplň“, „Krok“ a „Stop“ v levé části GUI. Stiskem tlačítka „Vyplň“ uživatel spustí zvolený algoritmus v klasickém režimu. Proces vyplnění se v tomto případě provede celý a uživateli je zobrazen výsledek vyplnění. Stiskem tlačítka „Krok“ se zvolený algoritmus spustí v režimu krokování. Po ukončení kroku je uživateli zobrazena aktuální podoba oblasti a pro každý algoritmus

jeho specifické atributy. Opětovným stiskem tlačítka „Krok“ uživatel provede další krok algoritmu. Stisk tlačítka „Stop“ má za následek vymazání výplně oblasti a také uvedení všech hodnot atributů algoritmů do původního stavu. Po stisku tohoto tlačítka je možné znovu začít demonstraci vyplňování od jejího počátku. Pod ovládacími tlačítky se nachází informační oblast, do které je vypsán stručný popis zvoleného algoritmu. Uživatel má možnost shlédnout podrobnější popis algoritmu přístupný přes Menu v horní části GUI. V dolní části GUI se nachází Status bar, který funguje jako drobná nápověda uživateli, pro akce které může vykonávat. V horní části GUI se nachází Menu bar, který obsahuje komponentu pro volbu algoritmu, který se má demonstrovat. Další položkou v menu je nápověda k ovládání GUI. Poslední položku tvoří informace o algoritmech použitých v této aplikaci.

7 Závěr

Cílem této práce bylo přiblížit čtenáři funkci algoritmů vyplňování uzavřených oblastí ve 2D. A vybrané algoritmy demonstrovat v implementované aplikaci. Hlavní zástupci vektorových vyplňovacích algoritmů jsou popsáni v kapitole 2. Principy rastrových algoritmů jsou pak popsány v kapitole 3. Pro popis byly vybrány pouze algoritmy, kterými se zabývám dále v práci, a které obsahuje implementace demonstrační aplikace. Pro implementaci algoritmů vyplňování a celé demonstrační aplikace byl zvolen jazyk C#. Implementace této aplikace přinesla několik problémů. Největším z nich byla volba způsobu krokování algoritmů. Tento problém je popsán v úvodu 4. kapitoly.

Textová část práce se snaží srozumitelně seznámit čtenáře s principem funkce algoritmů pro vyplňování uzavřených oblastí ve 2D. Dále pak se způsobem jakým byly algoritmy implementovány v aplikaci. Uvádí také, které třídy aplikace pro svoji funkci používá a přibližuje jakou funkčnost tyto třídy implementují.

Implementovaná aplikace se pak snaží uživateli názorně demonstrovat postup vyplňování těchto oblastí, při používání zvoleného algoritmu a oblasti pro vyplnění. Uživateli zobrazuje aktuální stav vyplňované oblasti, se kterou algoritmy pracují. V informativní části uživatelského rozhraní může pozorovat změny stavu vnitřních proměnných algoritmu, ke kterým dochází v průběhu vyplňování. Rovněž má k dispozici stručný popis funkce algoritmu. Může také využít nápovědu, která obsahuje detailnější popis algoritmů a popis ovládání uživatelského rozhraní.

Do implementované aplikace by mělo být bez větších úprav možné doimplementovat další algoritmy pro vyplňování uzavřených oblastí ve 2D. Případně i další varianty algoritmů, které již jsou v aplikaci implementovány.

Literatura

[1] Žára, J., Beneš, B., Sochor, J., Felkel, P.: Moderní počítačová grafika. 2. vydání, Brno, Computer Press, 2004.

[2] Míchal, V.: Výukový program pro demonstraci ořezávání 2D objektů a vyplňování 2D uzavřených oblastí. [bakalářská práce], Brno, FIT VUT v Brně, 2006.

[3] Eller, F.: C# - Začínáme programovat. Praha, Grada Publishing, 2002.

[4] Kent, J.: Visual C# 2005 bez předchozích znalostí. Brno, Computer Press, 2007.

[5] Kršek, P.: Základy počítačové grafiky. [studijní opora], Brno, FIT VUT v Brně, 2006.

[6] Kršek, P., Španěl, M.: Základy počítačové grafiky, Vyplňování 2D oblastí. [přednáška k předmětu Základy počítačové grafiky], Brno, FIT VUT v Brně, 2007.

Seznam příloh

Příloha 1: Hierarchie tříd