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

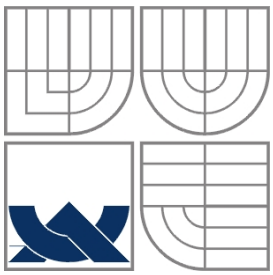
ANALÝZA ALGORITMŮ BOOLEOVSKÝCH OPERACÍ NAD OBECNÝMI POLYGONY

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

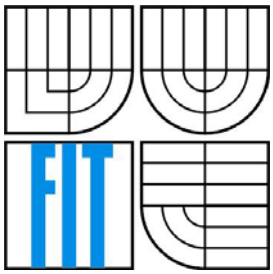
AUTOR PRÁCE
AUTHOR

Bc. TOMÁŠ DANĚK

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

ANALÝZA ALGORITMŮ BOOLEOVSKÝCH OPERACÍ NAD OBECNÝMI POLYGONY

ANALYSIS OF GENERAL POLYGON BOOLEAN OPERATION ALGORITHMS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. TOMÁŠ DANĚK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VÍTĚZSLAV BERAN

BRNO 2008

Abstrakt

Tato diplomová práce se zabývá algoritmy pro booleovské operace nad obecnými polygony. Mezi booleovské operace se řadí např. průnik, sjednocení nebo rozdíl. Obecný polygon může být např. sebeprotínající s otvorem. Pravděpodobně nejznámější booleovskou operací je oříznutí polygonu obdélníkovým oknem. Na začátku práce jsou vysvětleny základní pojmy. V další části je popsán princip vybrané množiny algoritmů, které provádějí booleovské operace nad polygony. V závěrečné části je provedeno komplexní srovnání implementací algoritmů jak z hlediska rychlosti výpočtu, tak z hlediska schopnosti zpracovat různě složité typy vstupních dat. Výstupem práce je souhrnné zhodnocení všech algoritmů a dynamická knihovna, která obsahuje implementace všech algoritmů.

Klíčová slova

ořezávání, obecný polygon, cad, 2D počítačová grafika, booleovské operace, viditelnost, odstranění skrytých ploch

Abstract

This thesis deals with general polygon boolean operation algorithms. Boolean operations are e.g. intersection, union or difference. A general polygon can be e.g. a selfinterecting polygon with inner hole. Clipping of polygons against a rectangular window is probably the most familiar boolean operation on polygons. At first, basic definitions are listed. Then the principles of a selected set of boolean operation algorithms are reviewed. Finally, a complex comparison of the algorithms is undertaken. Performance as well as the ability to handle degenerate cases are tested. The output of this thesis is an overall evaluation of algorithm properties and a dynamic library that contains the implementation of all of the tested algorithms.

Keywords

clipping, arbitrary polygon, general polygon, cad, 2D computer graphics, boolean operations, visibility, hidden surface removal

Citace

Daněk Tomáš: Analýza algoritmů booleovských operací nad obecnými polygony, diplomová práce, Brno, FIT VUT v Brně, 2008

Analýza algoritmů booleovských operací nad obecnými polygony

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. V. Berana
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Tomáš Daněk
19.5.2008

Poděkování

Děkuji Ing. V. Beranovi za vedení této práce.

Obsah

1 Úvod

Ořezávání polygonů, jakožto jedna z booleovských operací nad polygony, je jednou ze základních úloh v počítačové grafice. Jedním z nejstarších problémů, kde vznikla potřeba ořezání polygonu, bylo správné zobrazení polygonů na zobrazovacím zařízení. Zde bylo potřeba oříznout části polygonu, které ležely mimo okno výstupního zařízení. Dalším impulsem pro vývoj a zvyšování efektivity algoritmů ořezávání byla úloha zobrazení pouze viditelných objektů scény. Tato operace se řadí mezi vektorové algoritmy řešení viditelnosti a nazývá se Hidden Surface Removal (též HS Elimination nebo HS Determination). V paralelním raytracingu je ořezávání využito při rozdělování a distribuci objektů scény několika procesorům. Další oblastí použití jsou technické aplikace typu GIS, CAD, CAM. V těchto aplikacích jsou algoritmy ořezávání uplatněny např. u 2D B-Rep booleovských operací, tzn. návrh složitých 2D útvarů kombinací a transformací několika jednoduchých. Booleovské operace nad polygony jsou také často využívány například v software na návrh obvodů s velkou integrací (VLSI).

Každá aplikace má na algoritmus ořezávání specifické požadavky. U aplikací, kde je nutná efektivnost, se používají jednodušší algoritmy, které umí ořezávat pouze konvexním polygonem (v nejjednodušším případě obdélníkovým oknem). U aplikací, kde je nutná přesnost a korektnost (např. CAD), je nutné použít komplexní algoritmy, které umí pracovat s konkávními polygony či polygony, které protínají sebe sama.

V uvedených aplikacích není možné jednoduše použít algoritmus pro ořezání úsečky na jednotlivé hraniční úsečky polygonu, protože by mohly vzniknout samostatné nenavazující úsečky a nebyla by zachována uzavřenost hranice, která je nutná např. pro vyplnění polygonu barvou. Proto vznikla potřeba speciálních algoritmů, které zajistí uzavřenost hranice polygonu.

Cílem této práce je proniknout do problematiky ořezávání obecných polygonů a nastudovat a popsat nejpoužívanější algoritmy. Dále pak nalézt dostupné hotové implementace algoritmů, případně vybrané algoritmy naimplementovat, tak aby je bylo možné komplexně porovnat. Testování a porovnání algoritmů bude hodnotit hlediska omezení vstupních dat, kvality výstupních dat, ale i hlediska rychlosti výpočtu v různých typových případech a s různě komplexními vstupními daty. Bude též provedena názorná vizualizace výsledků. Z výstupu práce by mělo být patrné, jaké jsou výhody a nevýhody jednotlivých algoritmů, případně pro jaké typy aplikací jsou jednotlivé algoritmy vhodné. Následuje stručný popis jednotlivých částí práce.

V první části této práce jsou uvedeny definice obecného polygonu a booleovských operací nad obecnými polygony, které jsou nutné pro porozumění dalšímu textu.

Po vymezení základních pojmů je popsán princip vybrané množiny nejdůležitějších algoritmů pro úlohu řešení viditelnosti polygonálních ploch. Jsou též popsány problematické části těchto

algoritmů a možnosti jejich řešení. Dále je zde uveden stručný popis dostupných implementací těchto algoritmů.

V závěrečné části je popsáno vytvořené testovací prostředí a je zdokumentováno komplexní srovnání algoritmů podle několika různých hledisek. V závěru této části je provedeno shrnující zhodnocení jednotlivých algoritmů.

Tato práce navazuje na Semestrální projekt se stejným názvem. V rámci Semestrálního projektu byl proveden sběr studijních materiálů a dalších podkladů. Dále byly podrobně nastudovány a implementovány základní verze algoritmů Weiler-Atherton a Greiner-Hormann. Do diplomové práce byla ze Semestrálního projektu převzata kapitola 2 a část kapitoly 3.

2 Obecný polygon a booleovské operace

V této kapitole je definován obecný polygon a jsou zde popsány booleovské operace nad polygony. Jsou vysvětleny základní pojmy ze zpracovávané tématiky, jejichž znalost je nezbytná pro pochopení dalšího textu.

2.1 Definice obecného polygonu

„Polygon“ je anglický výraz pro mnohoúhelník. V české počítačové terminologii se používá anglický termín a proto bude používán i v této publikaci, vyjma matematické definice. U důležitých pojmů v definicích i v dalším textu práce, kde význam českého výrazu nemusí být zcela zřejmý, je v závorce uváděn i ekvivalent v anglickém jazyce.

Dle [1] je **mnohoúhelník** (též n-úhelník) definován jako část roviny vymezená úsečkami, které spojují určitý počet bodů (nejméně tři), z nichž žádné tři sousední neleží na jedné přímce, nebo výstižněji jako část roviny ohraničená uzavřenou lomenou čárou.

Níže jsou vymezeny některé základní pojmy týkající se polygonů, částečně dle [2]:

- úseky ohraničující lomené čáry se nazývají **hrany** (edges)
- průsečíky dvou hran se nazývají **vrcholy** (vertices)
- **konvexní** (convex) je takový polygon, jehož všechny vnitřní úhly jsou menší než 180°. Každá přímka nakreslená přes polygon (vyjma tečny hrany nebo vrcholu) protne hranici polygonu přesně dvakrát.
- **nekonvexní** (non-convex) je takový polygon, jehož alespoň jeden vnitřní úhel je větší než 180°. Lze nalézt přímku, která protíná jeho hranici více než dvakrát
- **jednoduchý** (simple) je takový polygon, jehož hranice se navzájem neprotínají. Všechny konvexní polygony jsou jednoduché. Nazývají se též Jordanovy polygony, podle teorému, kterým lze dokázat, že takový polygon rozděluje rovinu na dvě části – vnitřek a vnějšek.
- **konkávní** (concave, re-entrant) je nekonvexní a jednoduchý polygon. Polygon je konkávní, pokud existují dva body uvnitř polygonu, které nelze spojit úsečkou, která by ležela uvnitř polygonu
- **sebeprotínající** (self-intersecting) je takový polygon, jehož hranice protíná sebe sama

V rámci počítačové grafiky je pojem polygon chápán jako 2D útvar, který je definován souřadnicemi svých vrcholů a je možné ho obarvit, vystínovat nebo pokrýt texturou. K výše uvedeným pojmům se

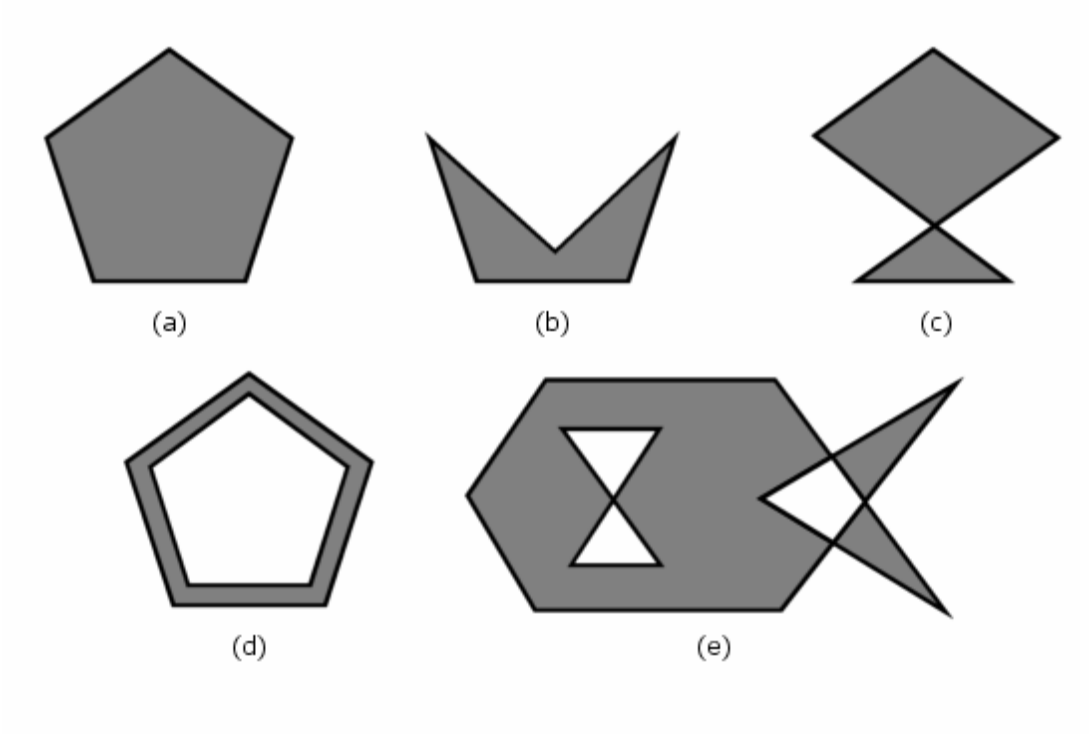
přidává pojem **komplexní** (complex) polygon. Komplexní polygon není ani konvexní ani konkávní a vzniká ve dvou případech:

- polygon je sebeprotínající
- hranice polygonu se skládá z několika oddělených částí, jako např. polygon s otvorem uvnitř

Význam komplexního polygonu v rámci počítačové grafiky je nutné odlišit od významu matematického, kde se jedná o útvar v prostoru komplexních čísel.

Uzavřený polygon lze také definovat jako seřazený seznam vrcholů, kdy první a poslední vrcholy jsou shodné. Je nutné uvést, že za vrcholy polygonu jsou považovány pouze konce hran, nikoliv průsečíky hran.

Pojem **Obecný polygon** (General, Arbitrary polygon) v počítačové grafice budeme tedy chápat jako libovolný jednoduchý nebo komplexní polygon, nebo množinu takových polygonů. Pod pojmem General polygon může být též míněn polygon, jehož hrany nejsou úsečky, ale křivky. Těmito polygony se tato publikace nebude zabývat.



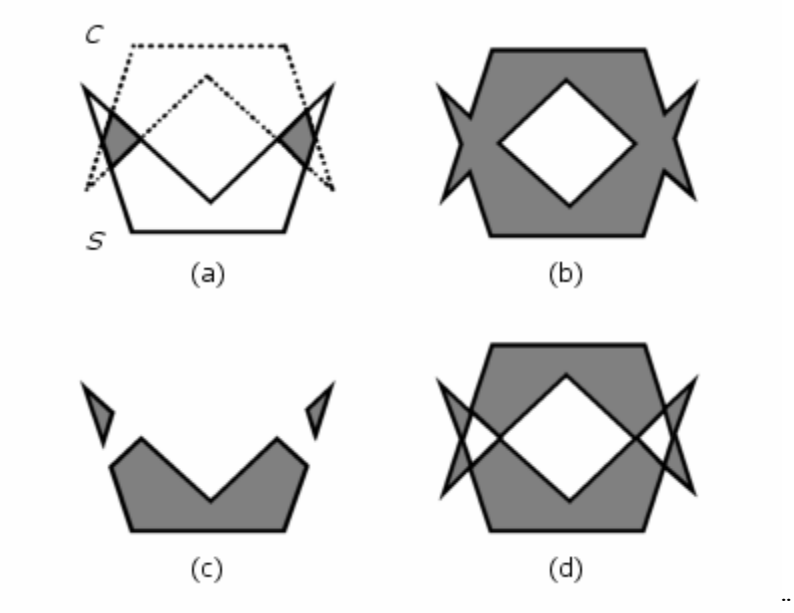
Obrázek 1: Ukázky polygonů – (a) konvexní polygon, (b) konkávní polygon, (c) sebeprotínající polygon, (d) polygon s otvorem, (e) obecný polygon

2.2 Booleovské operace nad obecnými polygony

Booleovské množinové operace (Boolean set operations) umožňují z více geometricky jednoduchých útvarů vytvořit útvar složitější. Při 3D návrhu v CAD systémech je často používána technika Constructive Solid Geometry (CSG). Při této technice je vhodnou posloupností booleovských operací a transformací aplikovaných na primitiva vytvořeno požadované, komplikované, těleso. Ve dvourozměrném případě nejsou operandy booleovských operací tělesa, ale polygony. Aplikací booleovských operací na polygony je vytvořen útvar s komplikovaným tvarem, který by bylo nepraktické definovat přímo pomocí souřadnic vrcholů.

Základní booleovské operace jsou:

- **sjednocení** (union, merge, OR) $P \cup Q$
- **průnik** (intersection, AND) $P \cap Q$, nejpoužívanější operace – ořezání polygonu
- **rozdíl** (set-theoretic difference, substraction, SUB) $P \setminus Q$
- **exkluzivní OR** (symetric difference, XOR) $P \oplus Q$



Obrázek 2: Booleovské operace nad polygony S a C – (a) průnik $S \cap C$, (b) sjednocení $S \cup C$, (c) rozdíl S \ C, (d) exkluzivní OR $S \oplus C$

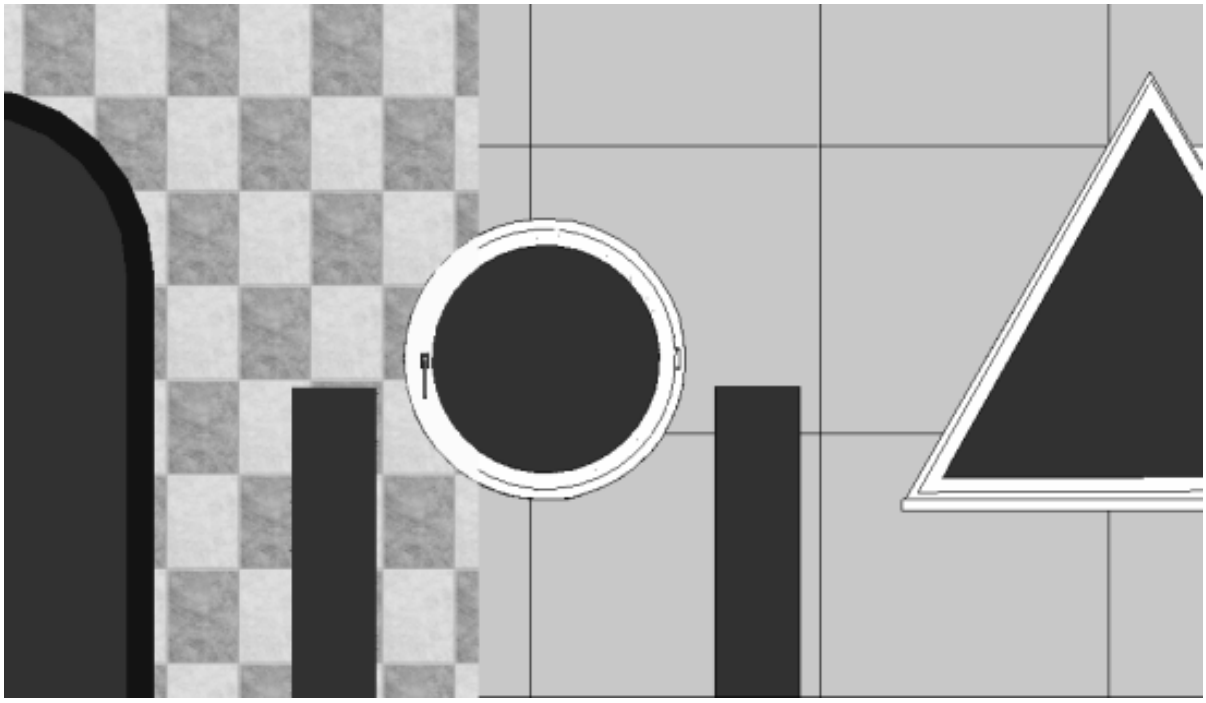
Standardní booleovské množinové operace nejsou pro účely modelování postačující, protože jejich aplikací nemusí vždy vzniknout smysluplný výsledek. V případě CSG může vzniknout výsledek, který není tělesem. V případě 2D modelování mohou vzniknout např. izolované prvky nebo

neuzavřené polygony. Proto jsou zavedeny tzv. regularizované booleovské operace. Značí se hvězdičkou a jsou definovány: $P\ op^{*}\ Q = closure(interior\ (P\ op\ Q))$, kde closure je uzávěr a interior je vnitřní oblast vzniklá booleovskou operací. Výsledkem regularizované operace je vždy uzavřený polygon, nebo množina polygonů. Případy, kdy při booleovských operacích nad polygony vznikají nežádoucí útvary se nazývají degenerativní (degenerate cases) a jejich odstraňování je často řešeno rozšířením (extension) ořezávacího algoritmu.

Při návrhu algoritmu pro booleovské operace je potřeba vzít v úvahu, že ani jednoduché polygony nejsou uzavřeny nad booleovskými operacemi. Tzn. že výsledkem operace nemusí být jednoduchý polygon, může to být například množina polygonů a děr.

Vstupem booleovských operací jsou dva operandy. Obzvláště v případě ořezávání, tedy booleovské operace průnik, je v literatuře první z operandů (ořezávaný polygon) nazýván **subject polygon** (též clippee) a druhý (ořezávací polygon) **clip polygon** (též clipper). Dále budou v této aplikaci používány anglické termíny.

Ukázka možného použití booleovských operací nad obecnými polygony v praxi je na obrázku 3. Ukázka je z aplikace na návrh pokládky obkladů a dlažeb. Plochu pokládky, což je polygon s otvory (okna, dveře), je pro účely vizualizace nutné rozřezat na čtvercové dlaždice, tzn. provést operaci ořezání čtvercovým oknem.



Obrázek 3: Praktická ukázka použití booleovské operace

3 Algoritmy pro úlohu řešení viditelnosti polygonálních ploch

V následující kapitole je popsán princip vybrané množiny algoritmů pro booleovské operace nad obecnými polygony. Zvláštní pozornost je věnována algoritmům Weiler-Atherton a Greiner-Hormann, které byly implementovány autorem práce. Výčet algoritmů zdaleka není úplný, byl však vybrán vzorek reprezentující všechny důležité principy a přístupy.

3.1 Algoritmus Sutherland-Hodgman

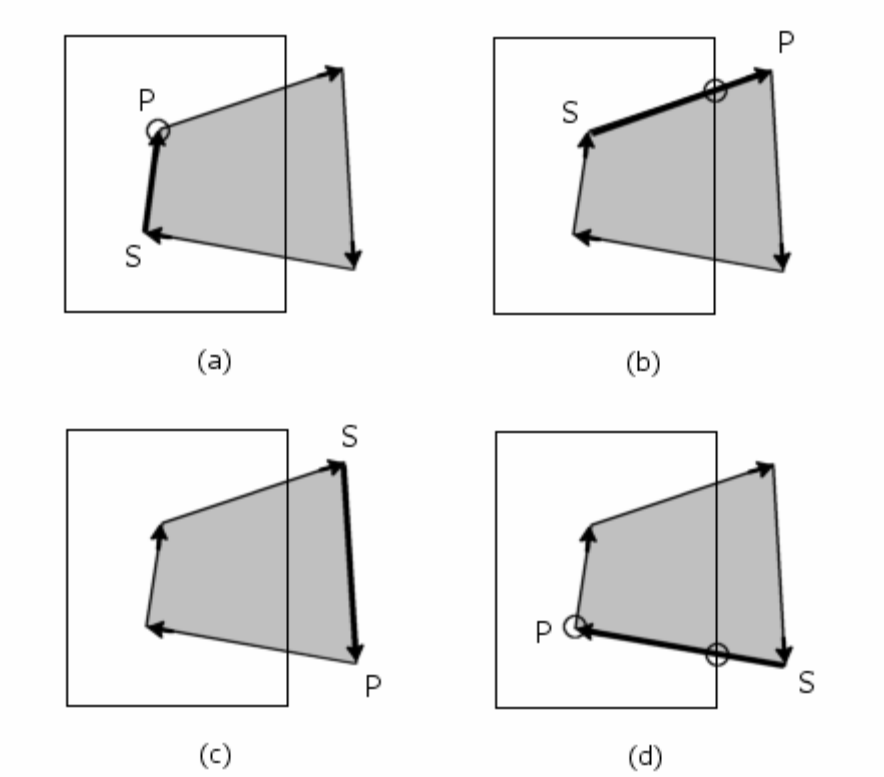
Tento algoritmus publikovali v roce 1974 pánové Sutherland a Hodgman [4]. Algoritmus intuitivně ořezává polygon postupně podle každé ořezávací roviny nebo hranice okna. Je to přirozený způsob, jaký by byl nejspíše použit při ručním ořezávání papírového polygonu nůžkami. Ořezávacím oknem může být konvexní polygon, velmi často to však bývá obdélník. Algoritmus umožňuje rekurzivní, proudové zpracování. Tzn. že jakmile je jeden vrchol zpracován vzhledem k první ořezávací rovině (clipping plane), může být odeslán ke zpracování vzhledem k další rovině. Při implementaci lze pro N ořezávacích rovin použít N-krát stejný modul. Modul si vždy pamatuje pouze dva vrcholy – předchozí a současný. U obdélníkového okna lze výhodně polygon otáčet o 90 stupňů a ořezávat stále stejnou rovinou, což zjednodušuje výpočet. Otáčení se provádí záměnou x a y-ových souřadnic současně se změnou znaménka.

Algoritmus je velmi efektivní pro dva speciální případy – když je polygon zcela uvnitř ořezávacího okna a když je polygon zcela mimo ořezávací okno.

Algoritmus v cyklu prochází všechny vrcholy polygonu, pro každý vstupní vrchol P může algoritmus vygenerovat žádný, jeden nebo dva výstupní vrcholy v závislosti na vztahu mezi vrcholem P, předchozím vrcholem S a ořezávací rovinou.

Existují 4 různé varianty polohy vrcholů S a P (hrany) vzhledem k ořezávací rovině, viz obrázek 4. Zpracovávaná hrana je vyznačena tučně, výstupy kolečkem.

a) oba vrcholy leží uvnitř ořezávacího okna (ve viditelné oblasti), výstupem je vrchol P
b) hrana vystupuje z ořezávacího okna, výstupem bude nový vrchol – průsečík
c) oba vrcholy leží mimo ořezávací okno, žádný výstup
d) hrana vstupuje do ořezávacího okna, dva výstupy – průsečík a vrchol P



Obrázek 4: Sutherland-Hodgman - Čtyři případy polohy hrany polygonu vzhledem k ořezávací rovině

Omezení subject polygonu: jednoduchý

Omezení clip polygonu: konvexní

Degenerativní případy: konkávní, sebeprotínající polygony – vznik nadbytečných, překrývajících se hran

3.2 Algoritmus Weiler-Atherton

3.2.1 Popis algoritmu

Algoritmus popsali pánové Weiler a Atherton v roce 1977 ve svém článku o řešení viditelnosti [5]. Protože v rámci řešení viditelnosti bylo nutné výstup ořezávacího algoritmu odeslat na vstup další fáze ořezávání, musel být algoritmus obecně schopen zpracovat své vlastní výstupy, tzn. ořezávat nekonvexní polygony s vnitřními otvory nekonvexním oknem.

V tomto algoritmu je polygon reprezentován jako kruhový seznam vrcholů. Jeden seznam je potřeba pro hlavní obrys a další seznamy pro obrysy otvorů. Vrcholy hlavního obrysu jsou uloženy po směru hodinových ručiček, obrysy otvorů proti směru. Výhodou tohoto pravidla je, že při sledování obrysu je vždy vnitřek polygonu na levé straně a vnějšek na pravé straně.

Při provádění algoritmu nevznikají žádné nové hrany, všechny výstupní hrany jsou pouze části hran ořezávacího polygonu. Tím je minimalizován počet vzniklých výstupních polygonů.

Výstupem jsou dva seznamy polygonů, těch, které jsou uvnitř ořezávacího polygonu a těch, které jsou vně.

Kroky algoritmu:

1) Vytvoření seznamů vrcholů pro subject i clip polygon a pro všechny otvory
2) Výpočet průsečíků a vložení falešných vrcholů do obou seznamů, subject i clip, propojení obou falešných vrcholů (umožňuje přeskočení z jednoho seznamu do druhého)
3) Zpracování obrysů bez průsečíků
4) Vytvoření dvou seznamů průsečíků, v prvním jsou průsečíky, ve kterých clip polygon vystupuje ze subject polygonu, ve druhém jsou průsečíky, ve kterých clip polygon vstupuje do subject polygonu. Tyto dva typy průsečíků se na daném obrysu pravidelně střídají, takže je postačující určit pouze typ prvního průsečíku v řadě
5) Vlastní ořezávání – procházení obrysu a provedení otáčky vpravo na každém průsečíku
   a) Z prvního seznamu průsečíků jsou vybírány vrcholy, dokud není seznam prázdný
   b) Od průsečíku, vybraného v předchozím bodě, je sledován obrys subject polygonu, dokud není dosažen další průsečík
   c) Při dosažení průsečíků je pomocí odkazu proveden skok na obrys clip polygonu
   d) Sledování obrysu clip polygonu, dokud není dosažen průsečík
   e) Přeskočení zpět na subject polygon
   f) Opakovat body b – e, dokud není dosažen počáteční vrchol

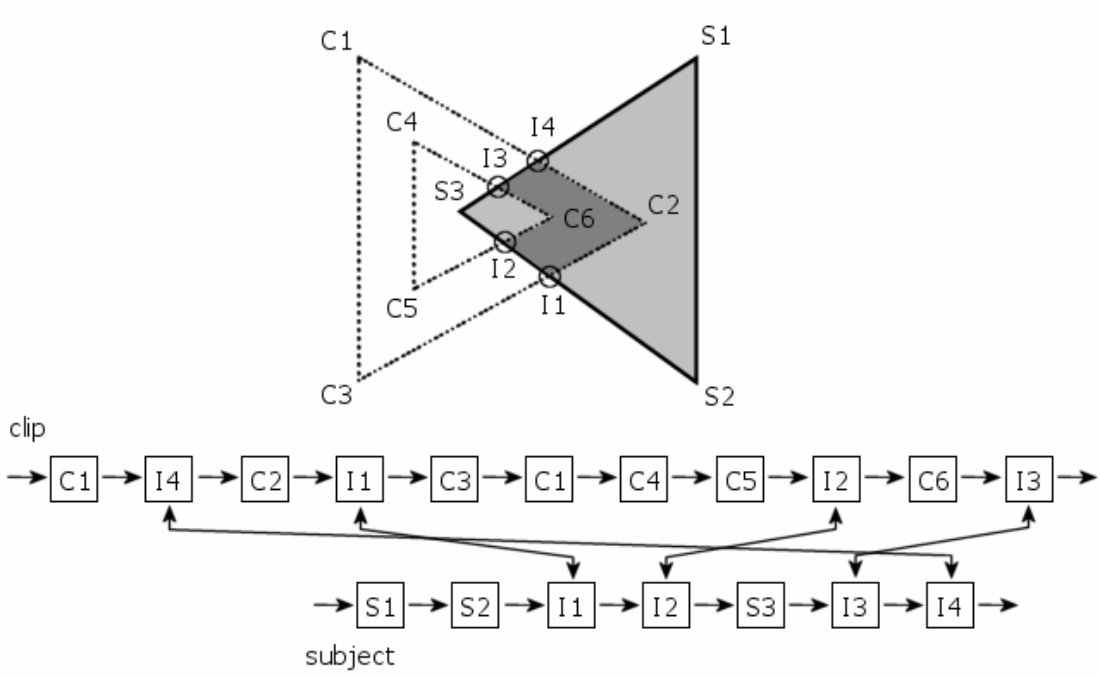
   Princip je názorně předveden na obrázku 5: Postup začíná v průsečíku I1 v seznamu vrcholů subject polygonu (spodní řada). Následuje posun doprava na průsečík I2 a poté skok na souseda (neighbour) do seznamu vrcholů clip polygonu (horní řada). Následuje posun na vrchol C6 a dále na průsečík I3. Následuje skok do spodní řady a posun vpravo na průsečík I4. Dále skok do horní řady a posun přes vrchol C2 na počáteční průsečík I1. Vzniklá posloupnost vrcholů výsledného polygonu je tedy: I1-I2-C6-I3-I4-C2.
6) Přiřazení obrysů otvorů k hlavním obrysům, obrysy otvorů lze detekovat např. pomocí určení orientace

Polygony vně ořezávacího okna je možné získat druhým průchodem. V tomto případě musí být seznam vrcholů obousměrný. Získání těchto polygonů je nutné pro provedení booleovských operací sjednocení a rozdíl.

Nevýhodou je složitost algoritmu. Je nutné vytvořit několik pomocných seznamů. Výhodou je shodná forma vstupních a výstupních dat.

Omezení subject polygonu: konkávní s otvory

Omezení clip polygonu: konkávní s otvory



Obrázek 5: Ukázkový příklad ořezávání algoritmem Weiler-Atherton

3.2.2 Speciální případy a jejich řešení

Při vytváření seznamu vrcholů a průsečíků je potřeba ošetřit následující speciální případy (v literatuře nazývané degenerate cases), které mohou vést k nekorektním výsledkům (vztah subject – clip polygon je zaměnitelný):

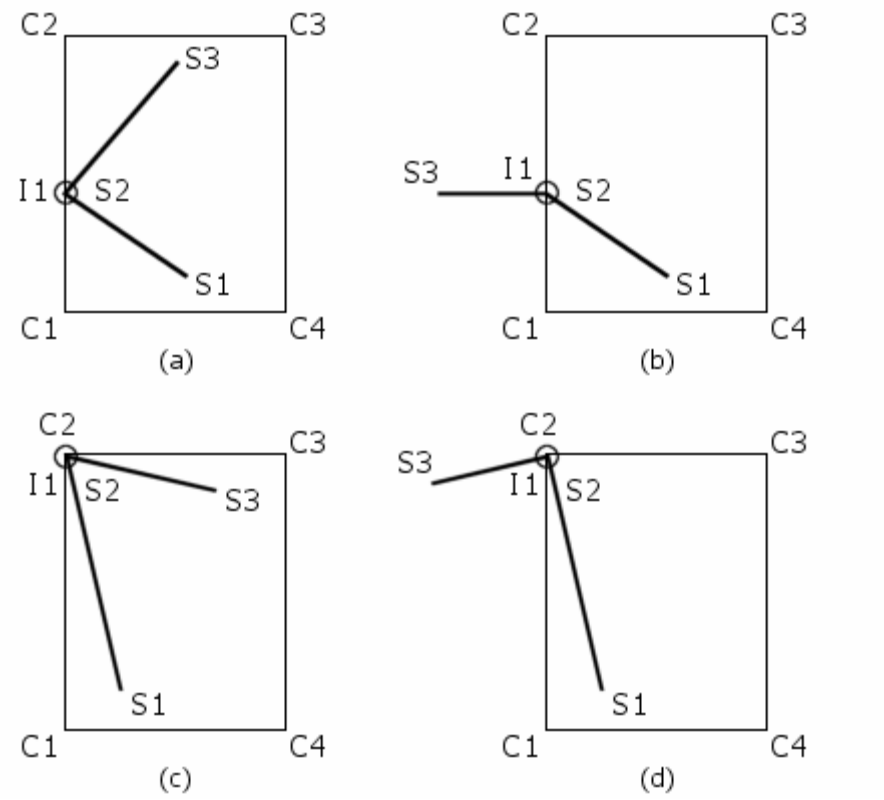
- Jeden z vrcholů subject polygonu leží na jedné z hran clip polygonu (Obrázek 6 - a, b)

  V případě (a) je při zpracování průsečík I1 ignorován a je vložen pouze vrchol S2, algoritmus se chová, jako kdyby se polygony v tomto místě neprotínaly

  V případě (b) je naopak vložen průsečík I1 a je odstraněn vrchol S2
- Jeden z vrcholů subject polygonu je totožný s některým z vrcholů clip polygonu (Obrázek 6 - c, d)

  Případ (c) je podobný případu (a), I1 je v obou seznamech vrcholů ignorován a jsou vloženy pouze vrcholy S2 a C2

  Případ (d) je podobný případu (b), do obou seznamů je vložen průsečík I1 a jsou odstraněny oba vrcholy, S2 i C2

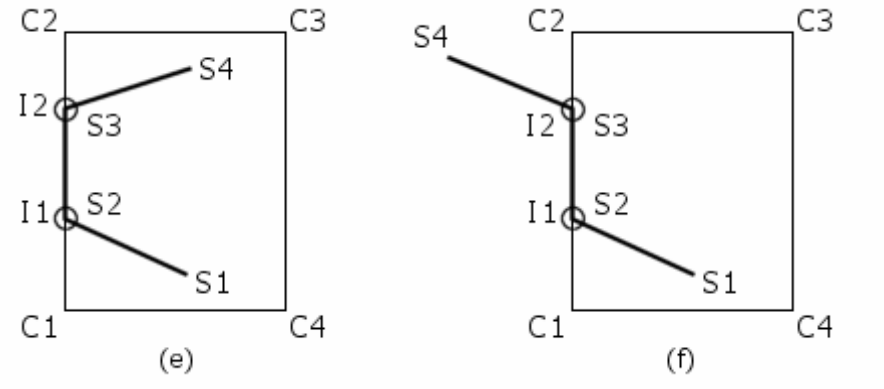


Obrázek 6: Weiler-Atherton - speciální případy

- Některá z hran subject polygonu se překrývá s některou z hran clip polygonu (Obrázek 7)

  V případě (e) jsou průsečíky I1 a I2 ignorovány

  V případě (f) je do seznamů vložen jeden z průsečíků I1, I2 a jeden z vrcholů S2, S3 je odstraněn



Obrázek 7: Weiler-Atherton - speciální případy

3.3 Algoritmus Greiner-Hormann

3.3.1 Popis algoritmu

Tento algoritmus, publikovaný v roce 1992 [7] se přístupem příliš neliší od algoritmu Weiler-Atherton. Je však jednodušší, elegantnější a obecnější.

Vstupní polygony jsou reprezentovány obousměrnými seznamy vrcholů. Datová struktura pro uložení vrcholů obsahuje kromě souřadnic a ukazatelů na sousední vrcholy pomocné informace o typu vrcholu. Velkým zjednodušením oproti algoritmu Weiler-Atherton je absence nutnosti vytváření pomocných seznamů. Výstupní polygony jsou získány průchodem přes upravené vstupní polygony.

Autoři problém ořezávání redukují na problém nalezení těch částí hranic polygonu, které leží uvnitř druhého polygonu. Tyto části mohou být spojeny ve výsledný polygon.

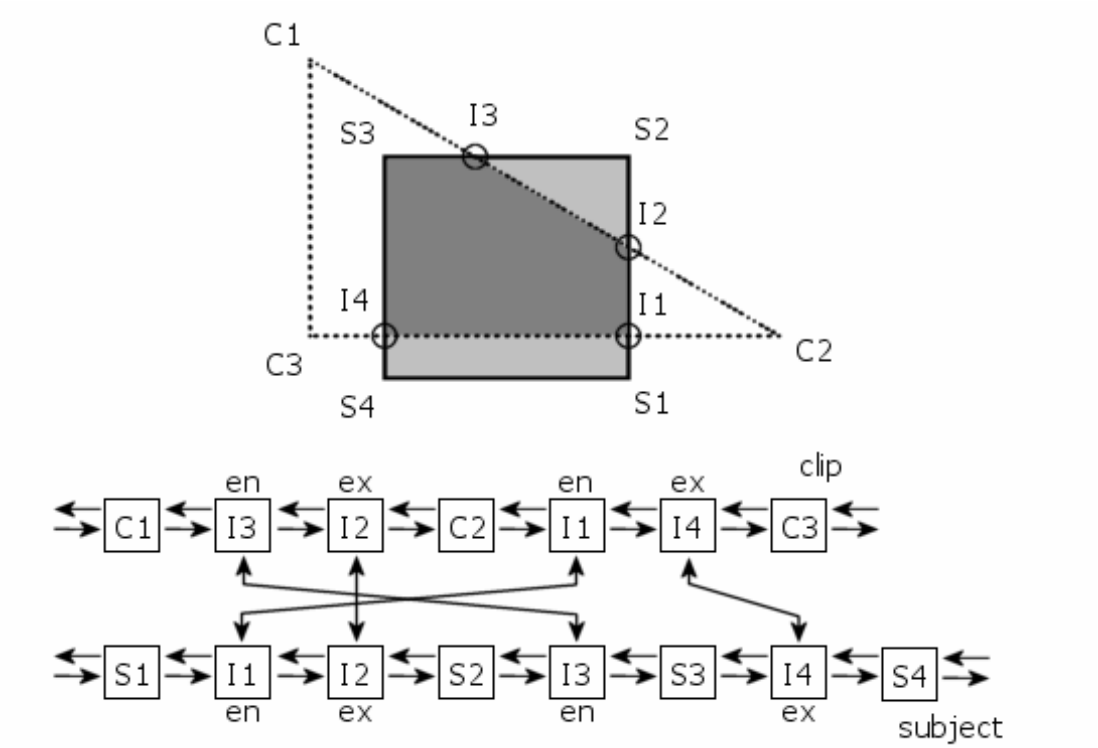
Nalezení částí subject polygonu, které leží v clip polygonu autoři názorně přirovnávají k jízdě s vozíkem plným křídy. Jízda začíná na některém z vrcholů polygonu, pokud ten leží uvnitř clip polygonu, otevřou se dvířka a sype se křída. Vozík je tlačen po obrysu subject polygonu a kdykoliv narazí na průsečík, přepne stav dvířek (otevřená/zavřená). Když vozík dojede zpět k počátečnímu vrcholu, všechny úseky obrysu, které leží uvnitř clip polygonu jsou označeny křídou. Stejný postup je proveden i s clip polygonem. Následně jsou tyto části spojeny a výsledkem je oříznutý polygon.

Rozhodování, zda se vrchol subject polygonu nachází uvnitř clip polygonu a naopak (inside test), je základním kamenem tohoto algoritmu a může být provedeno několika způsoby. Nejčastější jsou algoritmy crossing number a winding number. Při použití metody crossing number je zjišťován počet křížení polopřímky vedené z daného bodu s hranami polygonu. Pokud je počet lichý, bod leží uvnitř polygonu. Hodnota winding number je počet ovinutí polygonu kolem bodu. Pokud je hodnota winding number nenulová, bod leží uvnitř polygonu. Viz článek o výpočtu winding number [8].

Algoritmus Greiner-Hormann má tři hlavní fáze:

1) Vyhledání průsečíků hran, určení jejich relativní polohy vzhledem k počátečnímu a koncovému vrcholu hrany, vložení na správnou pozici do datové struktury subject i clip polygonu. Dva vložené průsečíky jsou označovány jako sousedi (neighbours) a jsou propojeny. Pokud nejsou nalezeny žádné průsečíky, pomocí pravidla even-odd je zjištěno, zda subject polygon leží zcela uvnitř clip polygonu, nebo naopak, nebo jestli jsou polygony odloučené.
2) Tlačení vozíku s křídou, aneb vyznačení vstupních a výstupních bodů. Každý průsečík bude označen jako vstupní nebo výstupní (entry/exit). Lze využít toho, že typy průsečíků entry/exit se pravidelně střídají. Stačí tedy určit typ prvního průsečíku v řadě.

3) Získání výsledného polygonu z propojených seznamů vrcholů. Na každém průsečíku nastává skok do druhého seznamu. Vstupní průsečík signalizuje posun v aktuálním seznamu doprava (vpřed), výstupní průsečík posun v seznamu doleva (zpět). Princip je názorně předveden na obrázku 8: Postup začíná v průsečíku I1 v seznamu vrcholů subject polygonu (spodní řada). Následuje skok na souseda (neighbour) do seznamu vrcholů clip polygonu (horní řada). Průsečík I1 je vstupní (značka en), proto nastává posun doprava na výstupní průsečík I4. Následuje skok do spodní řady a posun vlevo (značka ex) na vrchol S3. Dále posun na průsečík I3 a skok do horní řady a posun doprava na průsečík I2. Zbývá už jen skok do spodní řady. Je dosažen počáteční průsečík. Vzniklá posloupnost vrcholů výsledného polygonu je tedy: I1-I4-S3-I3-I2.



Obrázek 8: Ukázkový příklad ořezávání algoritmem Greiner-Hormann

Omezení subject polygonu: bez otvorů

Omezení clip polygonu: bez otvorů

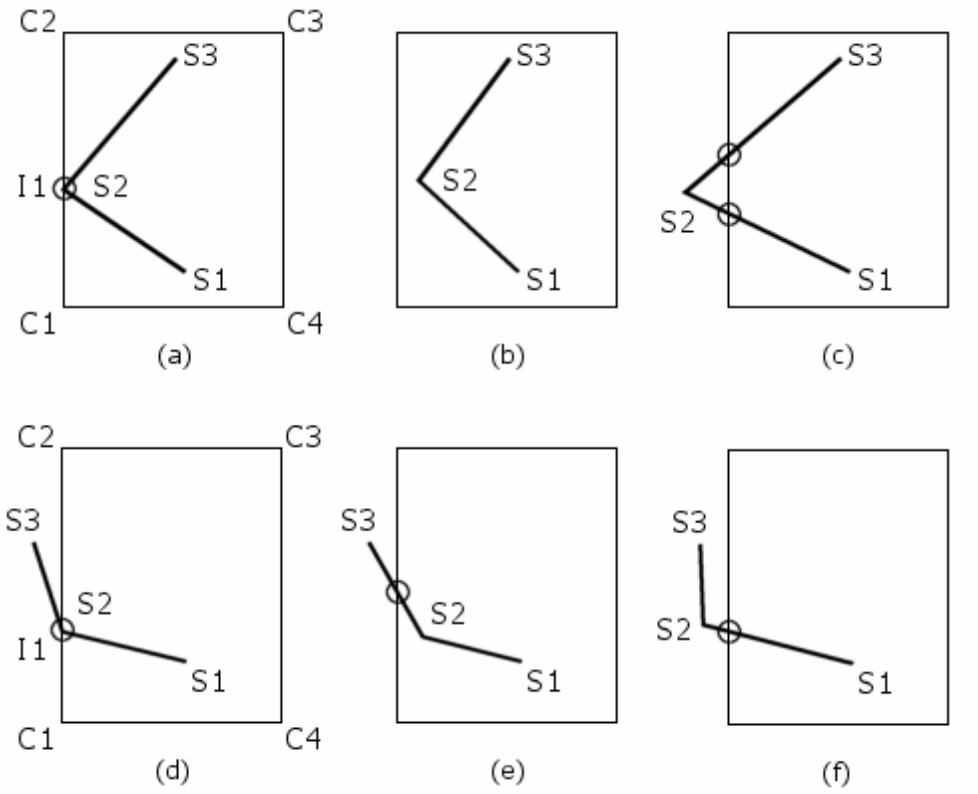
Malé modifikace algoritmu umožní provádět booleovské operace sjednocení a rozdíl.

3.3.2 Speciální případy a jejich řešení

U algoritmu Greiner-Hormann je potřeba ošetřit stejné speciální případy jako u algoritmu Weiler-Atherton. Tzn. když vrchol jednoho polygonu leží na hraně druhého polygonu, nebo se hrany překrývají. Dva možné přístupy řešení jsou uvedeny níže.

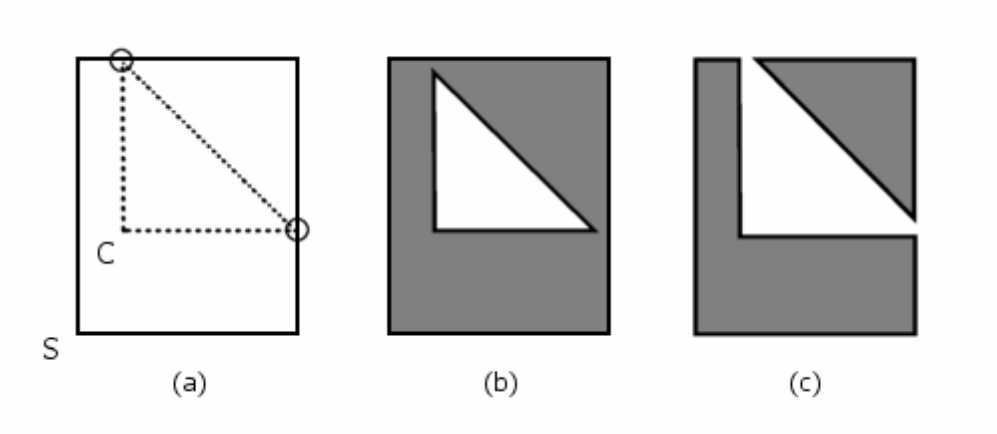
3.3.2.1 Vychýlení vrcholu (perturbation)

Toto řešení bylo navrženo samotnými autory algoritmu [7]. Autoři navrhují vrchol při výpočtu vychýlit na jednu nebo druhou stranu od hrany, na které leží. Pokud je odchylka na zobrazovacím zařízení menší než 1 pixel, výsledek je korektní. Nevýhodou je nutnost provést po vychýlení znovu výpočet nově vzniklých průsečíků. Existuje riziko, že po novém výpočtu se objeví nové speciální případy, což činí algoritmus nedeterministickým.



Obrázek 9: Speciální případy algoritmu Greiner-Hormann, řešení metodou vychýlení vrcholu.První případ (a) a jeho dvě možná řešení (b) a (c). Druhý případ (d) a jeho dvě možná řešení (e) a (f)

Metoda vychýlení vrcholu bohužel může při vychýlení na jednu nebo na druhou stranu vytvořit zcela rozdílné výsledky, viz Obrázek 10.



Obrázek 10: Operace rozdíl S \ C - různé výsledky při použití metody vychýlení vrcholu

3.3.2.2 Rozšíření algoritmu o nové typy průsečíků

Komplexní rozšíření algoritmu Greiner-Hormann, které zaručuje deterministické řešení speciálních případů, navrhli Kim, D.H. a Kim, M-J. ve článku [9]. K dosavadním dvěma typům průsečíků entry a exit jsou přidány další dva složené typy entry/exit a exit/entry. To znamená, že události výstupu i vstupu z oblasti se mohou vyskytnout v jednom vrcholu současně. Typ průsečíku je stanoven z tabulky pravidel na základě kombinace typů hran, které z průsečíku vycházejí. Typy hran jsou následující:

(on) hrana leží na obrysu druhého polygonu

(in) hrana leží uvnitř druhého polygonu

(out) hrana leží vně druhého polygonu.

K určení typu hrany je nutné provést tzv. inside test, což zvyšuje složitost výpočtu.

I s tímto rozšířením však algoritmus není schopen správně zpracovat situaci, kdy hrana jednoho polygonu prochází průsečíkem sebe sama (self intersection) druhého polygonu.

3.4 Algoritmus Vatti

Pan Vatti ve svém algoritmu [10] zvolil odlišný přístup, než autoři v předchozích případech.

V tomto algoritmu je polygon chápán tak, že je složený z levých a pravých hranic vedoucích z lokálních minim do lokálních maxim. Polygon je skenován zdola nahoru horizontálním pruhem (scanbeam), který je zdola a shora ohraničen událostmi typu vrchol nebo průsečík. Pokud je vstupní polygon sebeprotínající jsou jeho vlastní průsečíky vloženy do grafu. Je udržována tabulka aktivních hran, ve které jsou hrany protínající scanbeam, seřazené podle x-ové souřadnice. Každý vrchol i průsečík je klasifikován podle tabulky pravidel, která se liší podle prováděné booleovské operace. Při skenování je prováděn výpočet a klasifikace průsečíků a vrcholů a jsou vytvářeny výsledné polygony.

Operace sjednocení a rozdíl lze získat změnou klasifikačních pravidel. Horizontální hrany musí být zpracovány samostatně.

Vattiho algoritmus je obecný, ale velmi komplexní, vyžaduje vytvoření několika pomocných tabulek.

3.5 Algoritmus Schutte

Algoritmus, který publikoval pan Schutte [11] rozděluje hrany vstupních polygonů na vnitřní, vnější a sdílené. Poté hledá minimální polygony, které vznikají z průsečíků polygonů. Je zaveden pojem rodičovství (parenthood), který určuje vztah hrany ke vstupnímu polygonu a vztah minimálního polygonu ke vstupnímu polygonu. Rodičovství nabývá hodnot ano, ne, neznámé a smíšené. V závislosti na zjištěném rodičovství a typu prováděné booleovské operace jsou minimální polygony přidávány do výstupu.

3.6 Algoritmus použitý v knihovně Boolean

Autorem algoritmu je Klaas Holwerda [12]. Princip tohoto velmi komplexního algoritmu je založen na scanbeams a zpracování grafu. Algoritmus má 4 hlavní fáze:

- Příprava grafu – Vstupní polygony jsou převedeny na grafy, je provedeno zarovnání na celočíselnou mřížku, zjednodušení grafů v závislosti na velikosti konstanty MARGE (tato konstanta říká, jak velká vzdálenost je vzhledem k rozměrům polygonů považována za nepodstatnou) a posléze sloučení všech dílčích grafů do jednoho velkého.
- Výpočet průsečíků – je provedeno sloučení uzlů, přisednutí uzlů k hranám v závislosti na velikosti konstanty MARGE, jsou vypočteny průsečíky hran metodou Scanbeams (vertikální paprsky) a jsou vloženy do grafu.
- Příprava na booleovské operace – Graf nyní vytváří uzavřené oblasti a je nutné určit, do kterého ze vstupních polygonů každá z oblastí patří. Poté jsou všem hranám přiřazeny příznaky, které určují vztah hrany ke vstupnímu polygonu. Dle těchto příznaků je určeno, zda bude hrana zahrnuta do výsledku prováděné booleovské operace. V závěru této fáze jsou odstraněny nadbytečné hrany.
- Provedení booleovské operace – Dle příznaků určených v předešlém kroku jsou postupně vybírány takové hrany, který patří do výsledku prováděné booleovské operace. Postupně jsou získány všechny vnější i vnitřní obrysy (díry). Poté jsou díry spojeny s vnějšími obrysy a graf je převeden na výsledné polygony.

3.7 Algoritmus použitý v knihovně PolyBoolean

V první fázi tento algoritmus [13] počítá průsečíky mezi vstupními polygony. Na rozdíl od jiných algoritmů však zavádí tzv. vrcholy vyššího stupně (high degree vertices). V každém bodě se totiž může vyskytovat libovolný počet vrcholů různých polygonů. Pro každý bod průsečíku je vytvořen tzv. connectivity list, který obsahuje deskriptory všech průsečíků, které se nacházejí v tomto bodě, seřazené podle úhlů hran, které z těchto průsečíků vycházejí.

V další fázi tento algoritmus, podobně jako Schutte, značkuje hrany a obrysy (labeling). Zavádí celkem 4 typy hran (vnitřní, vnější, sdílené a sdílené s opačnou orientací) a 3 typy obrysů (vnitřní, vnější a protínající). Typy hran a obrysů jsou určeny podle souboru pravidel.

Podle značek a podle typu prováděné booleovské operace jsou poté hrany a obrysy vkládány do výsledné množiny polygonů. Při této finální fázi už není vůbec nutné pracovat s geometrickou informací. Přechod z hrany na hranu v průsečíku je proveden na základě souboru pravidel (jump rules).

Za hlavní přínos algoritmu autoři považují fakt, že booleovské operace jsou uzavřené, tzn. že jakákoliv výstupní data mohou být i vstupními daty operace.

3.8 Další algoritmy

V této kapitole jsou uvedeny některé další zajímavé algoritmy, které však nebyly zahrnuty do srovnávací části této práce.

3.8.1 Algoritmus Liang-Barsky

Algoritmus publikovaný ve článku [14] vychází ze známého algoritmu pro ořezávání úseček. Clip polygonem může být pouze obdélník, jehož hrany jsou rovnoběžné s osami x a y. Hrany subject polygonu jsou reprezentovány v parametrické podobě jako přímky se směrovým vektorem. Hrany clip polygonu, tedy strany obdélníkového okna, jsou reprezentovány stejným způsobem a jsou nazývány „hraniční přímky“. Průsečíky hran a hraničních přímek jsou označeny jako vstupní, resp. výstupní a lze z nich určit viditelnost hrany. Vodorovné a svislé hrany jsou řešeny jako speciální případy. Komplexně je vyřešen případ, kdy je do výsledného polygonu nutné vložit roh clip polygonu (zde nazývaný „turning point“). Autoři uvádí, že rychlost výpočtu je až dvakrát rychlejší než u algoritmu Sutherland-Hodgman. Tento algoritmus nebyl zařazen do testování, protože kvůli svým silným vstupním omezením není pro booleovské operace nad obecnými polygony vhodný.

3.8.2 Algoritmus Weiler

Navržený algoritmus [15] využívá k získání výsledků booleovských operací grafovou reprezentaci. Některé fáze algoritmu jsou podobné jako u algoritmu, použitém v knihovně Boolean (viz kapitola 3.6). Weiler navrhuje uložení topologických vztahů do kombinace datových struktur binární strom a okřídlená hrana. Nad hranami, reprezentovanými datovou strukturou okřídlená hrana, lze výhodně ve slučovací fázi provádět manipulační operace zvané „eulerovy operátory“, které zachovávají konzistenci grafu. Všechna vstupní data jsou sloučena do jednoho grafu, ze kterého lze, po provedení analýzy, průchodem získat výsledky všech booleovských operací.

3.8.3 Algoritmy založené na teorii simplexů

Ve článku [16] autoři Rivero a Feito popisují formální matematický model pro geometrické modelování založený na konceptu řetězce simplexů (simplicial chain). Simplex je definován jako n-dimenzionální zobecnění trojúhelníku. Tento formalizmus umožňuje pomocí algebraických operací nad řetězci simplexů popsat obecně jakýkoli mnohostěn v jakékoliv dimenzi. V navazující práci [17] autoři aplikují tento formalizmus na problém booleovských operací nad obecnými polygony. V případě 2D prostoru je simplexem trojúhelník. Pokud popisovaný polygon posuneme do prvního kvadrantu 2D souřadnicového systému, můžeme si simplex představit jako trojúhelník, jehož jednu hranu tvoří hrana polygonu a jehož zbývající vrchol je umístěn v počátku souřadnicového systému. Nad řetězci simplexů lze pak snadno provádět booleovské operace. Tato reprezentace omezuje množství speciálních případů a proto se autorům podařilo navrhnout robustní a efektivní algoritmus. Kolektivu autorů z Tsihghua University [18] se podařilo na stejném formalismu vystavět algoritmus, který je jednodušší, robustnější a rychlejší než algoritmus Rivero-Feito.

3.9 Dostupné implementace algoritmů

V této kapitole jsou stručně popsány dostupné knihovny pro booleovské operace nad obecnými polygony zahrnuté do srovnávací části této práce. Podrobné informace o licenčních podmínkách, verzích knihoven včetně odkazů na soubory ke stažení jsou uvedeny v příloze 1.

3.9.1 General Polygon Clipper library

GPC - General Polygon Clipper library je knihovna vytvořená Alanem Murtou na University of Manchester v roce 1999. Knihovna je založena na modifikované verzi Vattiho algoritmu (viz. kapitola 3.4). Vattiho algoritmus je rozšířen o podporu horizontálních hran a o robustní zpracování překrývajících se hran.

3.9.2 Boolean, a C++ library

Knihovnu Boolean vytváří od roku 1998 pan Klaas Holwerda se svými studenty. Popis algoritmu je uveden v kapitole 3.6. Knihovna pro booleovské operace je součástí prohlížeče a editoru GDSII souborů.

3.9.3 ClipPoly

ClipPoly je knihovna vytvořená panem Schutte v roce 1995 na University of Twente. Princip použitého algoritmu je uveden v kapitole 3.5.

3.9.4 PolyBoolean

Knihovnu PolyBoolean vytvořili pánové Michael Leonov a Alexey Nikitin z ruské akademie věd. Algoritmus je publikován v článku [13]. Tvůrci uvádějí, že algoritmus co se týče výkonnosti překonává všechny ostatní algoritmy a je numericky robustní. Knihovna umožňuje provádět i triangulaci polygonů.

3.9.5 CGAL

CGAL - Computational Geometry Algorithms Library je rozsáhlá knihovna geometrických algoritmů pro počítačovou grafiku. Na vývoji se podílí několik univerzit. Knihovna se skládá z řady modulů, jeden z nich se zabývá prováděním booleovských operací nad polygony.

3.9.6 Wykobi

Wykobi – Computational Geometry Library je středně rozsáhlá knihovna geometrických funkcí a algoritmů. Je vhodná pro menší projekty, které nepotřebují tak rozsáhlou knihovnu jako je např. CGAL.

3.10 Chronologický přehled algoritmů

V tabulce 1 je uveden chronologický přehled algoritmů popsaných v kapitole 3.

Název	Rok	Autoři	Princip	Implementace
Sutherland-Hodgman [4]	1974	Ivan E. Sutherland, Gary W. Hodgman	Postupné ořezávání podle hranic okna	Wykobi (Arash Partow)
Weiler-Atherton [5]	1977	Kevin Weiler, Peter Atherton	Průchod s „otočkou“ na každém průsečíku	Vlastní
Weiler [15]	1980	Kevin Weiler	Okřídlená hrana + graf	Není
Liang-Barsky [14]	1983	You-Dong Liang, Brian A. Barsky	Rozšířený algoritmus na ořezávání úseček	Není
Vatti [10]	1992	Bala R. Vatti	Horizontální scanbeam, pomocné tabulky	Generic Polygon Clipper (Alan Murta)
Schutte [11]	1995	Klamer Schutte	Značkování hran	ClipPoly (K. Schutte)
PolyBoolean [13]	1997	Michael V. Leonov, Alexey G. Nikitin	Značkování hran, vrcholy vyššího stupně	PolyBoolean (Michael V. Leonov)
Boolean [12]	1998	Klaas Holwerda	Scanlines + graf	Boolean (Klaas Holwerda)
Greiner-Hormann [7]	1998	Günther Greiner, Kai Hormann	Označení entry/exit průsečíků	Vlastní
Rivero-Feito [17]	2000	F.R. Feito, M. Rivero	Teorie simplexů	Není
Algoritmus kolektivu autorů z Tsinghua University [18]	2005	Y. Peng, J.-H. Yong, W.-M. Dong, H. Zhang, J.-G. Sun	Teorie simplexů	Není
Rozšíření Greiner-Hormann [9]	2006	Dae Hyun Kim, Myoung-Jun Kim	Nový typ vrcholu entry/exit	Není

Tabulka 1: Chronologický přehled algoritmů

4 Srovnání implementací

V této kapitole je popsáno testovací prostředí a posléze je zdokumentováno komplexní srovnání implementací algoritmů pro booleovské operace nad obecnými polygony. V závěru kapitoly je provedeno souhrnné vyhodnocení. Jednotlivé implementace budou pro jednoduchost v následujícím textu uváděny jako „algoritmy“ a tedy v mužském rodě, přičemž slovo „algoritmus“ bude často vynecháno např. „Algoritmus GPC byl nejrychlejší“ a „GPC byl nejrychlejší“. Termín „algoritmus“ bude použit, i když se v některých případech nejedná o algoritmus, ale o knihovnu.

4.1 Implementace algoritmů

Do testování jsem zařadil implementace algoritmů popsané v kapitole 3.9. Pro algoritmy Weiler-Atherton a Greiner-Hormann jsem nenalezl žádné vhodné implementace, proto jsem je naimplementoval sám. Základní verzi algoritmu Weiler-Atherton jsem rozšířil o řešení některých speciálních případů (viz kapitola 3.2.2). Původní algoritmus Greiner-Hormann jsem rozšířil o vychylování vrcholů (viz kapitola 3.3.2). Algoritmy jsem naimplementoval v Jazyce C++ s využitím knihovny STL.

Oba algoritmy využívají stejnou datovou strukturu pro uložení informací o vrcholu polygonu – `GHVertex`. Souřadnice vrcholů jsou uloženy v datovém typu `double`. Pro manipulaci se seznamy je použita STL třída `CList`. Pro rozhodování, zda je bod uvnitř polygonu je použita metoda crossing number. Funkce jsou označeny prefixem WA pro algoritmus Weiler-Atherton a GH pro Greiner Hormann. Prvotním krokem je naplnění datových struktur vstupními daty v metodách `WAPreparePoly` a `GHPreparePoly`. Poté probíhají jednotlivé kroky algoritmů v metodách `WAAlgRun` a `GHAlgRun`. Tyto metody zpracovávají vstup pomocí několik pomocných procedur. Pokud zpracování proběhlo v pořádku, výsledné polygony jsou získány z datových struktur v metodách `WAExtractPoly` a `GHExtractPoly`.

4.2 Testovací prostředí

Pro efektivní průběh testování jsem vytvořil pohodlné testovací prostředí. Pro editaci vstupních polygonů jsem využil upravenou verzi programu DXF Editor, který je výsledkem mé bakalářské práce [21]. DXF Editor umožňuje ruční i numerickou editaci vstupních polygonů, které je možné uložit do různých vrstev. Vytvořenou geometrii lze uložit a posléze znovu načíst ze souboru ve formátu XML. Geometrii polygonů lze také importovat ze souboru DXF. Aplikace DXF Editor je vhodná i pro názorné zobrazení a zkoumání výsledků booleovských operací.

Všechny implementace byly v jazyce C nebo C++ a proto nebyl problém je všechny umístit do jedné dynamické knihovny PolygonClipper.dll. Jako komunikační rozhraní mezi knihovnou a DXF Editorem jsem zvolil soubor ve formátu XML, se kterým DXF Editor nativně pracuje. Výhoda tohoto přístupu spočívá ve snadném vytvoření a zpracování souboru XML v mnoha vývojových platformách, snadná čitelnost a orientace v uložených datech a v neposlední řadě snadná možnost rozšíření struktury dat o přídavné informace. Protože každý algoritmus pracuje se svou vlastní datovou strukturou, bylo nutné vytvořit převodní procedury z a do univerzální datové struktury. Poté jsem vytvořil procedury pro načtení a uložení této univerzální datové struktury do souboru XML. Pro manipulaci se soubory ve formátu XML v jazyce C++ jsem použil knihovnu MS XML 2. Dynamická knihovna PolygonClipper exportuje metody pro provedení algoritmů s následujícími parametry: Název vstupního souboru, Název výstupního souboru a Typ booleovské operace.

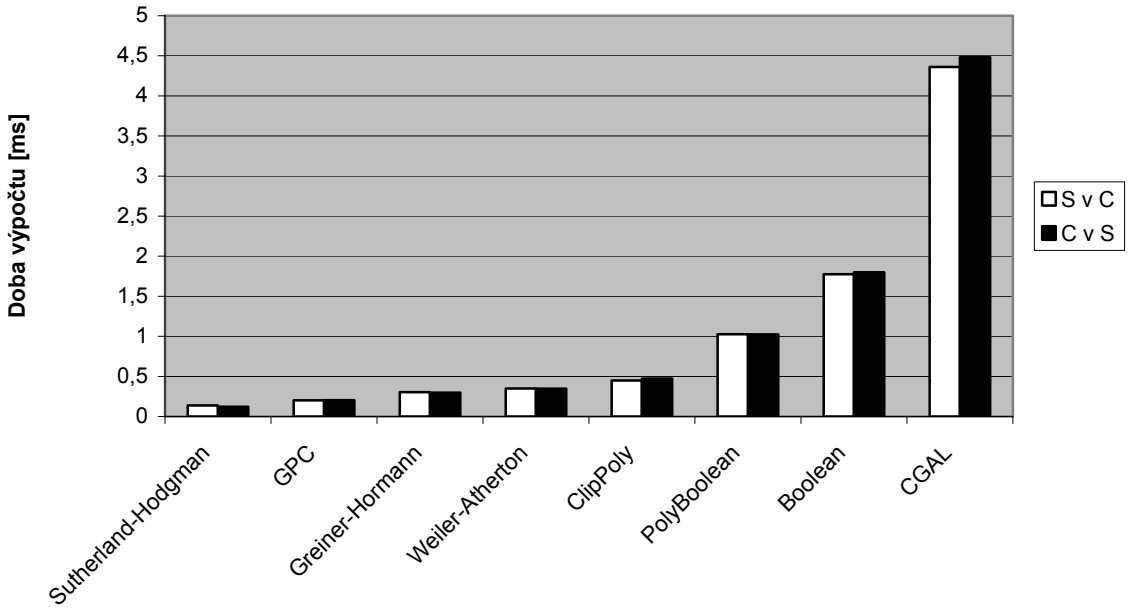
Popis sestavení knihovny i aplikace DXF Editor ze zdrojových kódu je uveden v příloze 2. Použití testovacího prostředí je stručně popsáno v příloze 3.

4.3 Testování rychlosti výpočtu

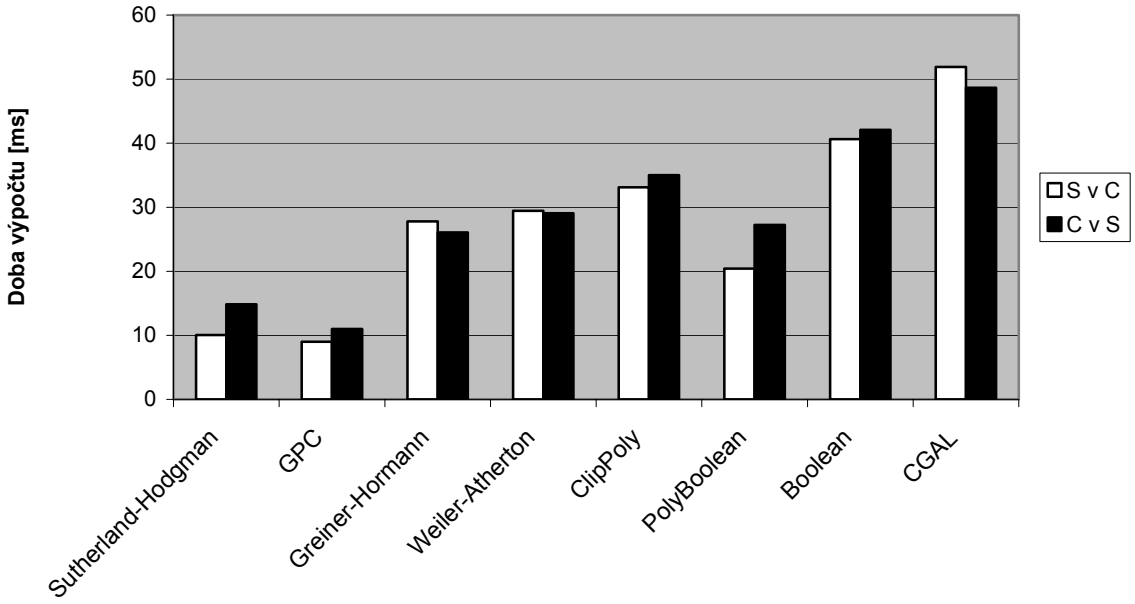
Srovnávací testy byly prováděny na notebooku Toshiba Satellite A60, s procesorem Pentium M 1.60 GHZ a 1 GB RAM. K měření času byla využita WinAPI funkce GetTickCount(), která vrací počet milisekund od startu systému. Aby byl získán změřitelný časový údaj (tzn. větší než jedna milisekunda), byly u testů s nízkým počtem vrcholů ve vstupních polygonech prováděny booleovské operace v cyklu s vysokým počtem opakování (1000, 10000). Každé volání funkce knihovny PolygonClipper bylo provedeno 5x, aby bylo zamezeno vlivu náhodných výkyvů výkonu počítače.

4.3.1 Jeden polygon je obsažen ve druhém

Tento test srovnává rychlost algoritmů v situaci, kdy jeden polygon zcela obsahuje druhý, tzn. že se polygony neprotínají, ani nemají sdílené hrany. Situace, kdy obdélníkové ořezávající okno zcela obsahuje třeba až tisíce polygonů, může nastat např. při vykreslování dost často, proto je důležité, aby algoritmus dokázal takovou situaci vyhodnotit co nejrychleji. Následující dva grafy ukazují výsledky v situaci, kdy subject i clip polygon mají každý 50, resp. 500 vrcholů. V grafech jsou zaneseny oba případy, tzn. když subject polygon zcela obsahuje clip polygon (C v S) a když clip polygon zcela obsahuje subject polygon (S v C).



Obrázek 11: Subject polygon uvnitř Clip polygonu a naopak. Subject i clip polygon mají každý 50 vrcholů.

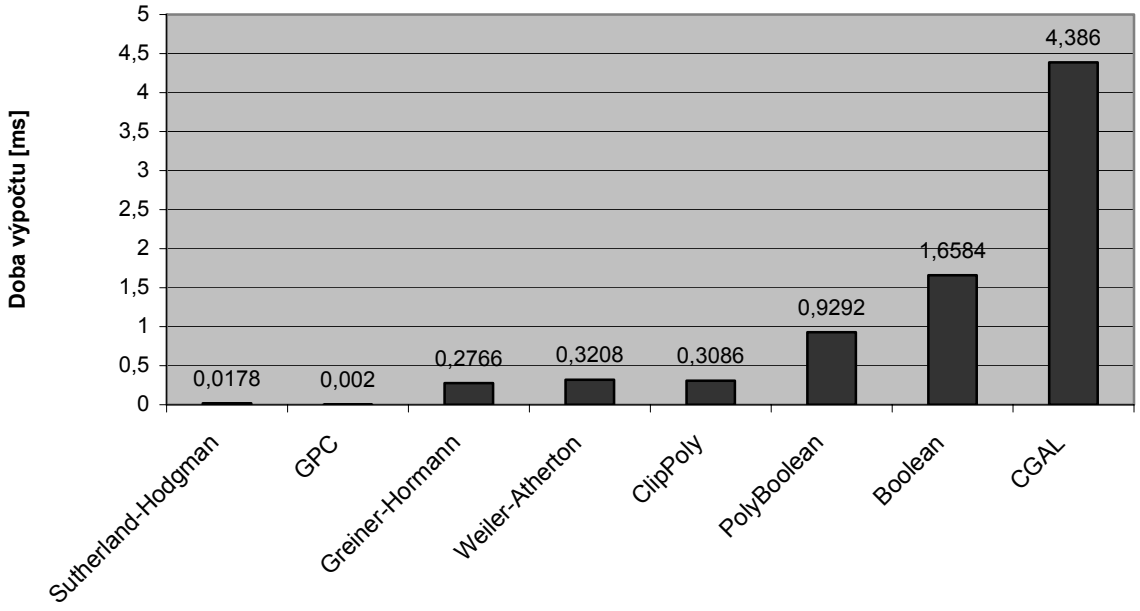


Obrázek 12: Subject polygon uvnitř Clip polygonu a naopak. Subject i clip polygon mají každý 500 vrcholů.

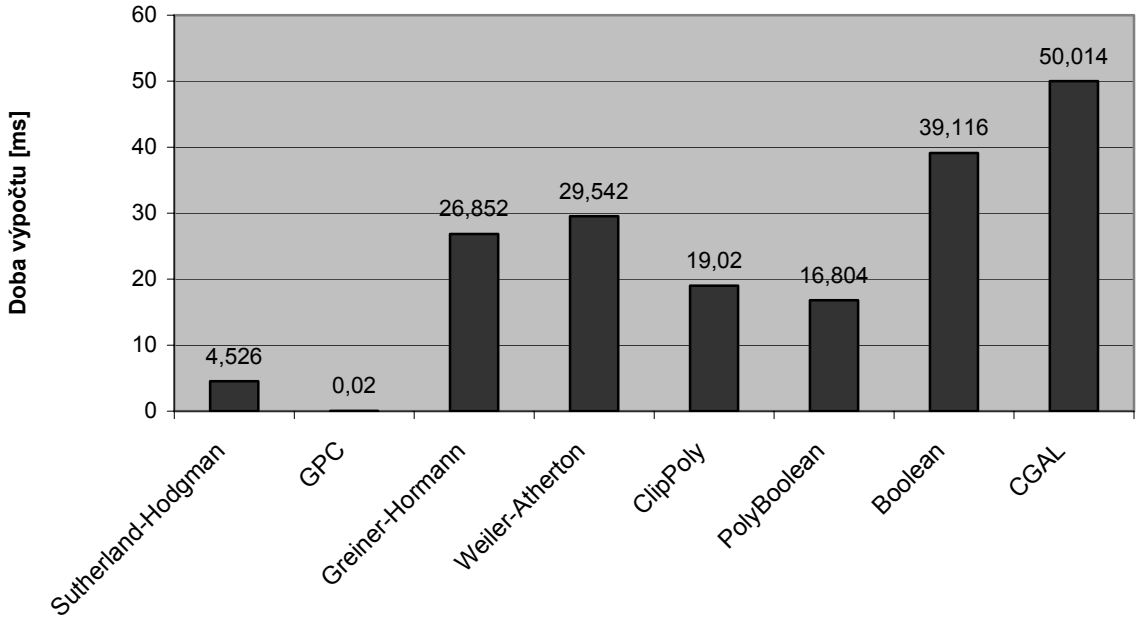
Z výsledků je zřejmé, že pro menší počet vrcholů jsou doby výpočtů celkem vyrovnané až na PolyBoolean, Boolean a CGAL. U vyššího počtu vrcholů již rozdíly nejsou tak velké. Nejlépe si vedl GPC, dále v pořadí následovaly Sutherland-Hodgman a PolyBoolean.

4.3.2 Polygony se neprotínají

Následující test zjišťuje rychlost algoritmů v případě, že se subject polygon s clip polygonem vůbec neprotínají, tzn. že jsou odloučené. V takovém případě nemá žádné další zpracování polygonů smysl.



Obrázek 13: Doba výpočtu v případě, že Subject polygon a Clip polygon se neprotínají (celkem 100 vrcholů)



Obrázek 14: Doba výpočtu v případě, že Subject polygon a Clip polygon se neprotínají (celkem 1000 vrcholů)

Z výsledků tohoto testu lze vyvodit, že jedině GPC s tímto speciálním případem počítá a prázdný výsledek vrátí okamžitě i pro vyšší počet vrcholů. Rychlost algoritmu Sutherland-Hodgman je dána jeho jednoduchostí. Pravdou je, že u ostatních algoritmů by šel tento speciální případ vyřešit případným předzpracováním. Například vzájemným porovnáním obdélníkových obálek polygonů by mohly být odhaleny případy, kdy jsou polygony zcela odloučené a již by další zpracování nemuselo probíhat.

4.3.3 Oříznutí obdélníkem

Tento test zjišťuje rychlost algoritmů v případě, že clip polygon je obdélník, což je velmi častá operace, u které je rychlost výpočtu obzvláště důležitá.

	Počet vrcholů subject polygonu			
	10	100	1000	10000
Sutherland-Hodgman	0,01142	0,0602	0,31	3,20
GPC	0,02682	0,5428	33,264	2748,90
Greiner-Hormann	0,02104	0,2064	4,586	251,70
Weiler-Atherton	0,02202	0,2024	4,626	229,50
ClipPoly	0,2404	3,615	198,492	18422,50
PolyBoolean	0,1903	2,12	54,196	2050,90
Boolean	0,3495	3,7934	145,27	7286,50
CGAL	1,687	6,5534	61,674	2615,54

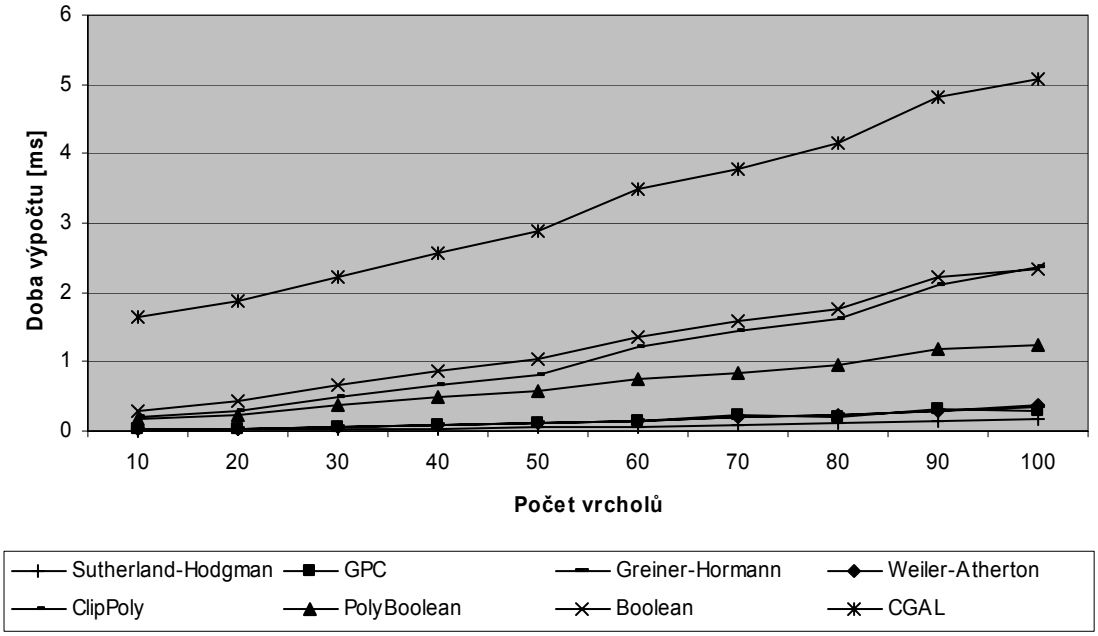
Tabulka 2: Doba výpočtu ořezání obdélníkovým oknem, v milisekundách

Z výsledků testu vyplývá, že Sutherland-Hodgman je při vysokém počtu vrcholů až stonásobně rychlejší než další konkurenti v pořadí. To je dáno tím, že ořezávání obdélníkem je jedna z hlavních úloh, pro které byl tento algoritmus vytvořen. Dobrých výsledků dosahují i algoritmy Weiler-Atherton a Greiner-Hormann.

4.3.4 Závislost doby výpočtu na počtu vrcholů

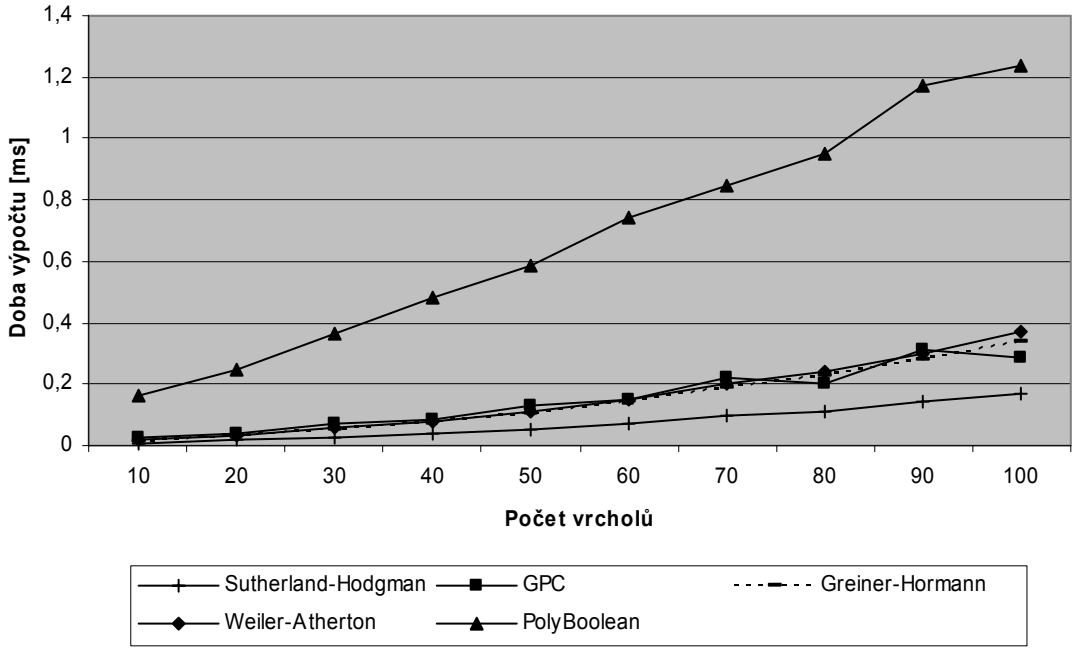
V tomto testu byly algoritmy postupně aplikovány na vstupní polygony s určitým počtem vrcholů (počet vrcholů subject polygonu = počet vrcholů clip polygonu). Testování bylo provedeno nejdříve pro relativně nízké počty vrcholů – násobky deseti v intervalu <10,100> a poté pro vysoké počty vrcholů – násobky tisíce v intervalu <1000, 10000>. Jako testovací subject polygon byla použita tzv. roztřesená kružnice (kružnice, která byla navzorkována na daný počet vrcholů, přičemž byl poté každý vrchol vychýlen ze své polohy o náhodnou vzdálenost směrem ke středu kružnice). Jako clip polygon byl použit pravidelný N-úhelník, který je konvexní a tudíž splňuje vstupní omezení všech

algoritmů. V grafu na obrázku 15 jsou vyneseny výsledky všech testovaných algoritmů pro nízké počty vrcholů.



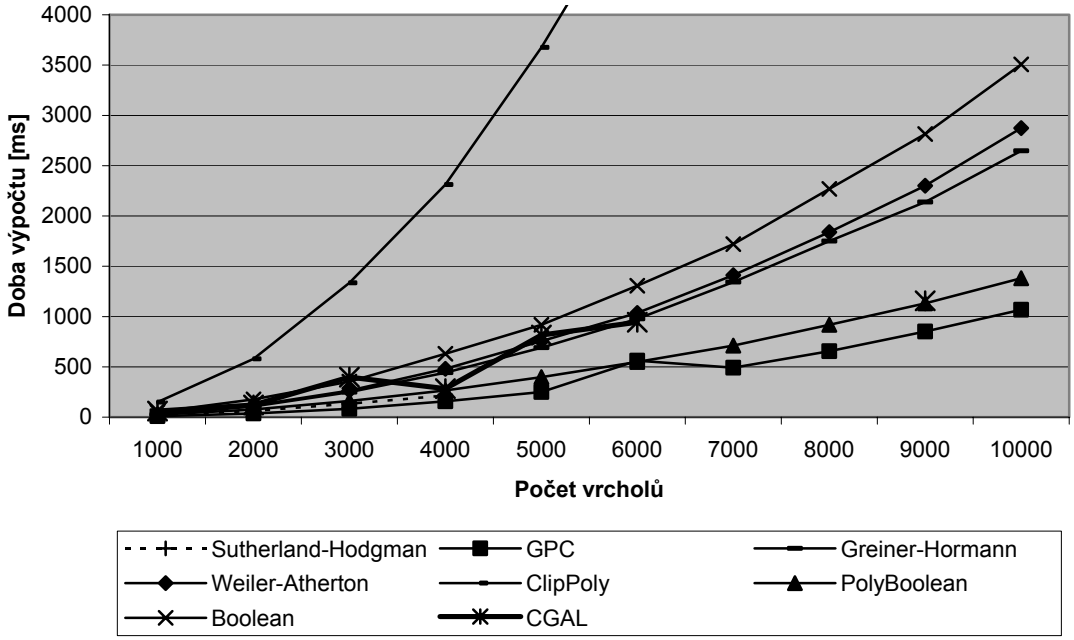
Obrázek 15: Graf závislosti doby výpočtu na počtu vrcholů pro počet vrcholů 10 – 100.

Z grafu je zřejmé, že algoritmus CGAL výrazně zaostává. Vzhledem k velkému rozptylu hodnot jsou pro lepší rozlišení v grafu na obrázku 16 zobrazeny pouze výsledky pěti nejrychlejších algoritmů.



Obrázek 16: Graf závislosti doby výpočtu na počtu vrcholů pro počet vrcholů 10 – 100. Detail 5-ti nejrychlejších algoritmů.

Z výsledků testů pro nízký počet vrcholů vychází jako vítěz algoritmus Sutherland-Hodgman, což je logické, protože je to algoritmus jednoduchý a přímočarý, který má přísná vstupní omezení a neřeší žádné speciální případy.



Obrázek 17: Graf závislosti doby výpočtu na počtu vrcholů pro počet vrcholů 1000 – 10000.

U testů pro vysoký počet vrcholů bohužel nastaly komplikace. Algoritmus Sutherland-Hodgman od počtu vrcholů 4000 vracel pouze prázdný výsledek. Weilter-Atherton od hodnoty 3000 sice vypočítal průnik polygonů správně, ale jako výsledek vrátil výsledek operace rozdíl. Algoritmus CGAL pro 7000, 8000 a 10000 vrcholů skončil s chybou. Tyto komplikace jsou pravděpodobně způsobeny vznikem speciálních případů, tzn. vícenásobných vrcholů či sdílených hran. Z grafu na obrázku 17 je patrné, že výsledky jsou vyrovnanější než u nízkého počtu vrcholů. Do popředí se dostávají algoritmy GPC a PolyBoolean. CGAL, který u nízkého počtu vrcholů výrazně zaostával, zde vyjma chybných výpočtů dosáhl velmi dobrých časů a např. pro 9000 vrcholů vypočítal správný výsledek téměř stejně rychle jako PolyBoolean.

Z testů rychlosti lze vypozorovat, že pro nižší počet vrcholů jednoznačně vedou jednodušší algoritmy. Pro vyšší počet vrcholů se výsledky srovnávají. To je pravděpodobně dáno tím, že komplexní algoritmy stráví dost času inicializací datových struktur, přičemž samotný výpočet už poté probíhá rychle.

4.4 Zpracování speciálních případů

V následující sérii testů bylo ověřeno, jak si jednotlivé algoritmy poradí se speciálními případy, které jsou způsobeny tvarem vstupních polygonů, nebo jejich vzájemnou polohou.

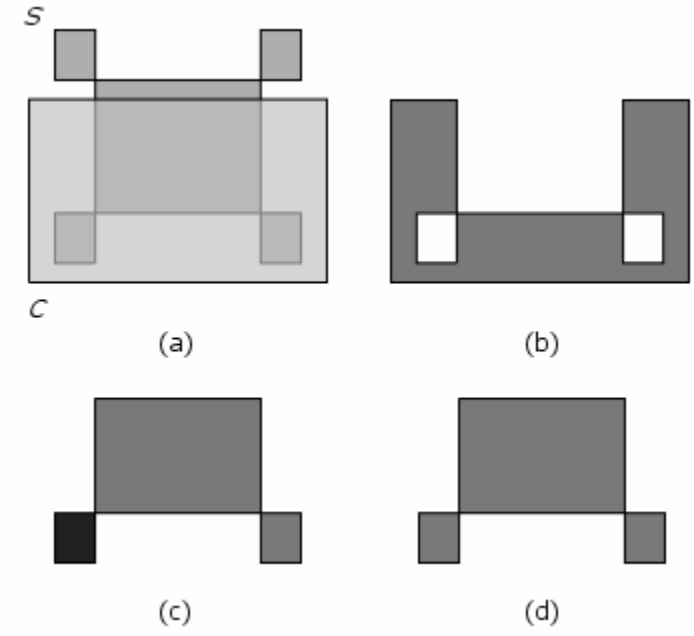
Vstupní i výstupní data byla vizualizována v programu DXF Editor. Znak S v obrázcích značí subject polygon, Znak C značí clip polygon. Výplně polygonů jsou vykresleny s částečnou průhledností, aby byly viditelné obrysy obou polygonů. Pokud je výsledkem operace více polygonů, jsou tyto odlišeny barvou výplně. Protože je poloha výsledných polygonů vzhledem ke vstupním polygonům zřejmá, jsou výsledné polygony pro větší názornost zobrazeny samostatně.

Ve všech testech byla nad vstupními polygony prováděna operace průnik (intersection). Zpracování uvedených speciálních případů není typem booleovské operace ovlivněno.

Ve většině testů byl jako clip polygon použit obdélník, aby mohl být do testů zařazen i algoritmus Sutherland-Hodgman.

4.4.1 Sebeprotínající polygony

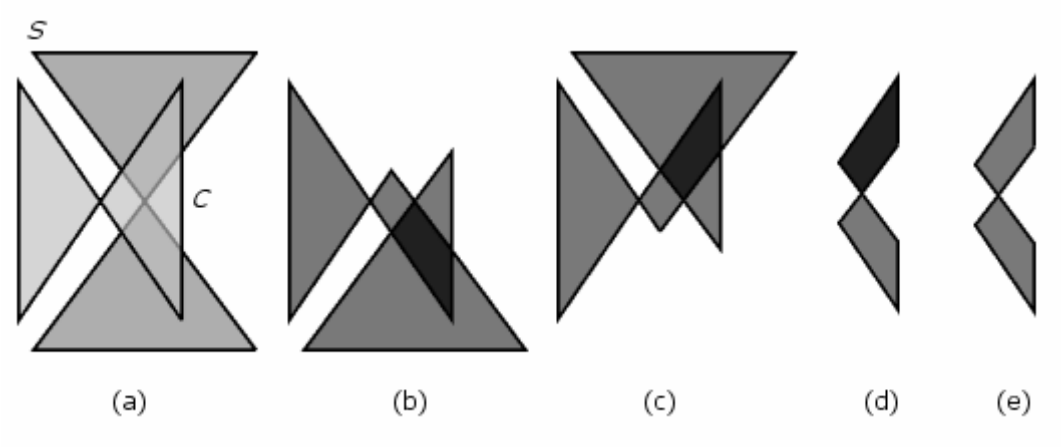
Sebeprotínající polygony mohou vzniknout například projekcí zkrouceného čtyřúhelníka do roviny. Na ověření schopnosti testovaných algoritmů takový polygon zpracovat, byly provedeny dva testy. V prvním testu byl za clip polygon zvolen obdélník. Ve druhém testu byl za clip polygon zvolen sebeprotínající polygon. Výsledky testů jsou zobrazeny na obrázcích 18 a 19. Různé typy výsledných polygonů jsou popsány v seznamech pod obrázky.



Obrázek 18: Test zpracování sebeprotínajících polygonů

a) Vstupní polygony
b) Výsledek provedení algoritmu Weiler-Atherton. Při průchodu seznamem vrcholu pravděpodobně došlo ke změně směru a algoritmus vrátil výsledek operace rozdíl.
c) Výsledek algoritmu GPC. Obsahuje dva polygony, větší z nich není sebeprotínající, v místě křížení jsou dva vrcholy.
d) Výsledek, který vytvořily implementace Sutherland-Hodgman, Greiner-Hormann Boolean a ClipPoly. Polygon, vytvořený algoritmem Boolean, není sebeprotínající, v bodech křížení jsou dva vrcholy.

Algoritmy PolyBoolean i CGAL vrátily prázdný výsledek, sebeprotínající polygon nedokázaly zpracovat.



Obrázek 19: Test zpracování sebeprotínajících polygonů (subject i clip)

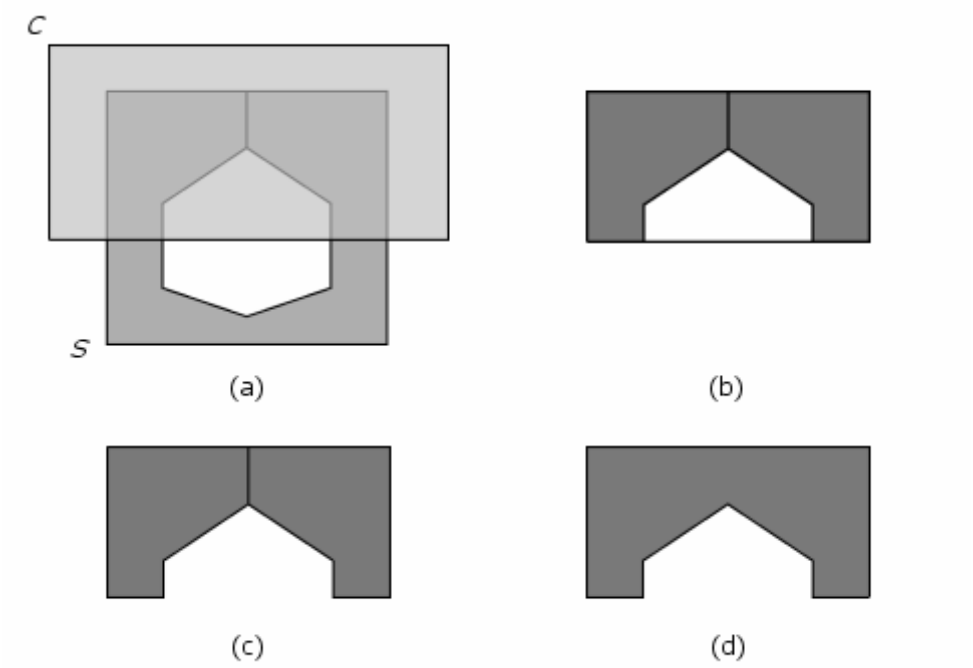
(a) Vstupní polygony
(b) Nesprávný výsledek Weiler-Atherton - dva výsledné polygony
(c) Nesprávný výsledek PolyBoolean – dva výsledné polygony
(d) Korektní výsledek GPC – dva výsledné polygony
(e) Korektní výsledek Greiner-Hormann, Boolean – jeden polygon. Výsledkem algoritmu Boolean není sebeprotínající polygon, v bodě křížení jsou dva vrcholy.

Výsledky (d) i (e) jsou korektní, ale přesto dost rozdílné. Posouzení kvality jednoho nebo druhého výsledku zřejmě závisí na dalším zpracování výstupních dat. V některých aplikacích může být například sebeprotínající polygon jako výstup nežádoucí.

Algoritmy Sutherland-Hodgman (neumí zpracovat nekonvexní clip polygon), ClipPoly i CGAL vrátily prázdný výsledek.

4.4.2 Otvory vytvořené hlavním obrysem

Některé z algoritmů neumí zpracovat otvory vytvořené pomocí vnořeného obrysu. V takovém případě je možné vytvořit otvor vytvořením uzavřené smyčky v rámci hlavního obrysu. V následujícím testu je ověřeno, zda jednotlivé algoritmy dokážou takto vytvořený obrys korektně zpracovat. Výsledky testu jsou znázorněny na obrázku 20, popis výstupních dat následuje pod obrázkem.



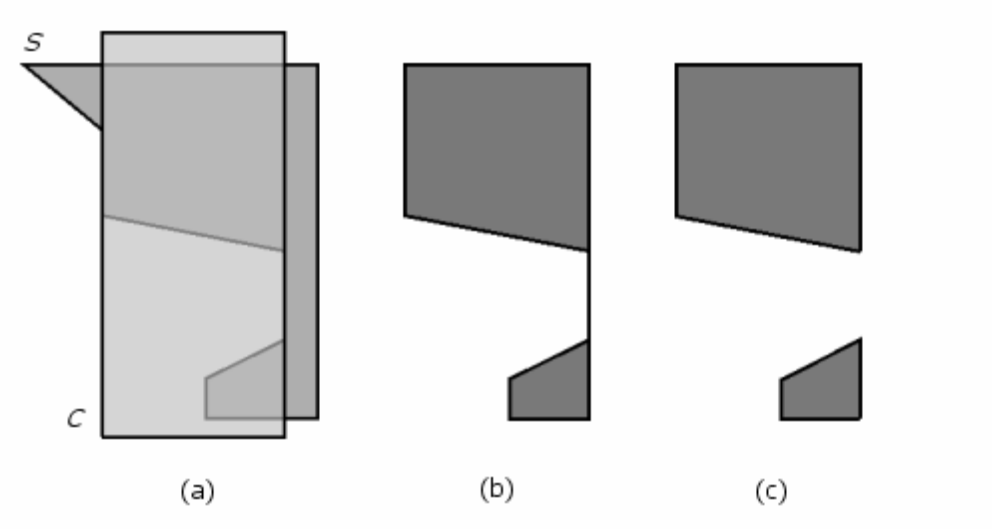
Obrázek 20: Test zpracování otvorů vytvořených hlavním obrysem

a) Vstupní polygony
b) Výsledek Sutherland-Hodgman
c) Výsledek Weiler-Atherton, Greiner-Hormann, PolyBoolean, ClipPoly
d) Výsledek GPC, Boolean, CGAL

Nejlepší výsledek (d) v tomto testu vrátily algoritmy GPC, Boolean a CGAL. Výsledek (c), složený ze dvou polygonů, je korektní, případně i použitelný pro další zpracování, avšak není považován za správný výsledek této operace.

4.4.3 Společné hrany polygonů

V dalším testu jsou jako vstupní data algoritmům předloženy polygony, které mají koincidentní hrany. To znamená, že se např. část hrany clip polygonu překrývá s některou z hran clip polygonu.

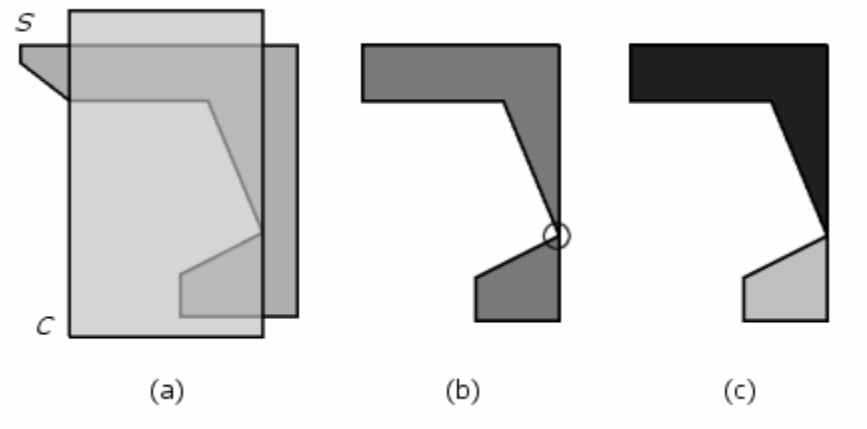


Obrázek 21: Test zpracování společných hran

Tento test zvládly bez problémů všechny algoritmy kromě Sutherland-Hodgman (c), kde je součástí výsledku zbytečná hrana a Greiner-Hormann. Ten se při výpočtu zacyklil, pravděpodobně při nedeterministickém vychylování vrcholů.

4.4.4 Vrcholy ležící na hranách polygonů

Situace, kdy vrchol jednoho z polygonů leží na hraně druhého polygonu může nastat poměrně často a proto je nutné, aby algoritmus dokázal tento speciální případ uspokojivě vyřešit.



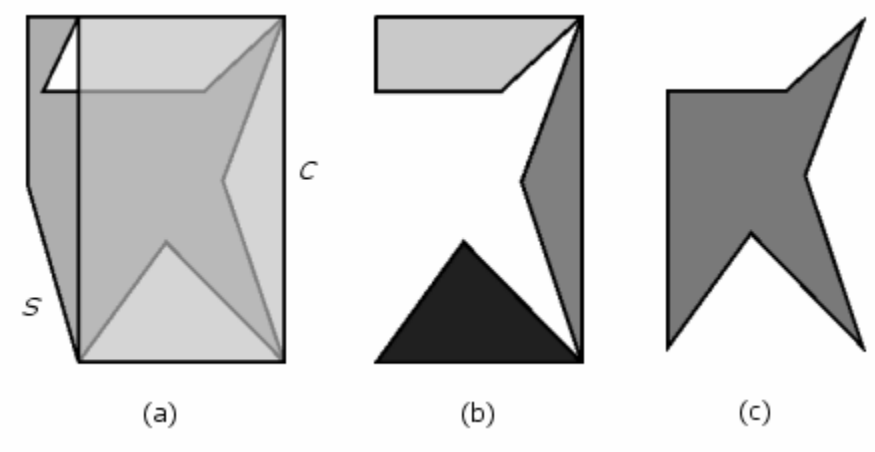
Obrázek 22: Test zpracování vrcholů ležících na hranách

a) Vstupní data
b) Výsledek Sutherland-Hodgman, Weiler-Atherton, GPC, Greiner-Hormann, Boolean
c) Výsledek PolyBoolean, ClipPoly

Všechny algoritmy vrátily rozumné výsledky. Výsledek (b) se u jednotlivých algoritmů liší v reprezentaci vyznačeného místa dotyku. Algoritmus Sutherland-Hodgman vytvořil jeden výsledný polygon, ve vyznačeném místě dotyku umístil dva vrcholy, které oba patří k levé straně polygonu. Algoritmus Greiner-Hormann vytvořil v místě dotyku dva vrcholy, obrys se v tomto bodě kříží. Algoritmy Boolean a CGAL umístily v místě dotyku dva vrcholy bez křížení obrysu. Algoritmy Weiler-Atherton a GPC vytvořily jeden polygon, v místě dotyku umístili jeden vrchol. Implementace PolyBoolean a ClipPoly vrátily jako výsledek dva polygony (c). Rozhodnutí o tom, zda jsou jako výsledek lepší dva polygony, nebo jeden polygon, zřejmě závisí na tom, jak budou výstupní data dále zpracovávána.

4.4.5 Společné vrcholy

Tento test ověřuje schopnost algoritmů zpracovat situaci, kdy je nějaký vrchol subject polygonu totožný s některým z vrcholů clip polygonu. Tento případ je vlastně speciálním případem dalšího testu v pořadí – testu na vrcholy vyššího stupně.



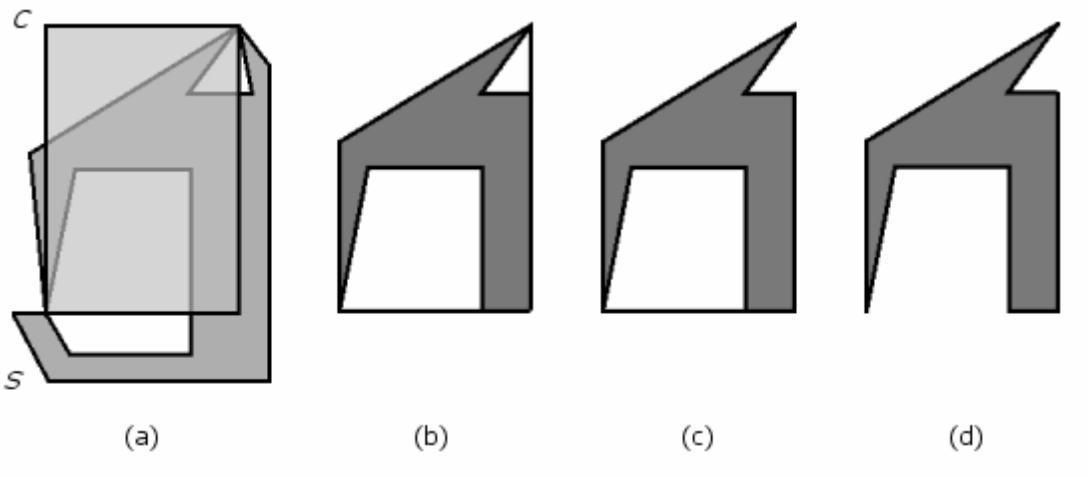
Obrázek 23: Test zpracování společných vrcholů

a) Vstupní polygony
b) Výsledek Greiner-Hormann se podobá výsledku operace C / S
c) Výsledek zbylých algoritmů

Společné vrcholy nezvládl korektně zpracovat pouze Greiner-Hormann. Výsledek Sutherland-Hodgman obsahoval zbytečné vícenásobné vrcholy.

4.4.6 Vrcholy vyššího stupně

Pojem „vrchol vyššího stupně“ je vysvětlen v kapitole 3.7. Ve vstupní datech pro tento test se vyskytují dva vrcholy třetího stupně. V těchto bodech se dotýká subject polygon sebe sama a navíc se v tom samém bodě vyskytuje vrchol clip polygonu.



Obrázek 24: Test zpracování vrcholů vyššího řádu

(a) Vstupní polygony
(b) Výsledek Greiner-Hormann
(c) Výsledek Sutherland-Hodgman
(d) Výsledek zbylých implementací vyjma ClipPoly

Tato vstupní data nezvládl zpracovat algoritmus ClipPoly, který vrátil prázdný výsledek. Greiner-Hormann a Sutherland-Hodgman vytvořily nekorektní výsledek se zbytečnými hranami. Weiler-Atherton skončil s chybou. Ostatní algoritmy vyprodukovaly očekávaná, korektní, výstupní data.

4.5 Numerická robustnost

Problém s přesností reprezentace reálných čísel se vyskytuje obecně u výpočtů prováděných v počítačích a výjimkou nejsou ani výpočty v počítačové grafice. Poloha bodu ve spojitém euklidovském prostoru je v počítači většinou reprezentována souřadnicemi uloženými jako číslo s omezenou přesností (float, double). Při uložení reálného čísla s omezenou přesností dochází k tzv. chybě zaokrouhlení. Důsledkem toho je, že dvě vypočtená čísla s pevnou přesností nelze obecně porovnávat na rovnost. Pokud máme například určit, zda se tři úsečky protínají v jednom bodě, můžeme narazit na problém. Ač se v euklidovském prostoru úsečky v jednom bodě protínají, v reprezentaci s pevnou přesností tomu tak být nemusí. Za numericky robustní jsou považovány implementace, které mají problém s přesností uložení reálných čísel uspokojivě vyřešen.

Standardním řešením je stanovit nějakou konstantu epsilon (např. 0.001) a pak rovnost dvou čísel definovat tak, že absolutní hodnota rozdílu těchto čísel musí být menší než epsilon. Epsilon tedy značí požadovanou přesnost porovnání dvou čísel, tato hodnota může také být nazývána tolerance. Tato metoda také bývá nazývána „fuzzy comparison“.

Z testovaných implementací používají aritmetiku s plovoucí čárkou a konstantu epsilon: Greiner-Hormann (GH_DELTA), Weiler-Atherton (WA_DELTA), GPC (GPC_EPSILON), ClipPoly a wykobi. Tento způsob není považován za robustní.

Knihovna CGAL používá přesnou aritmetiku, tzv. „exact computation paradigm“. Pro reprezentaci čísel v CGAL lze použít vestavěné datové typy nebo externí knihovny, např. knihovnu GMP (Multiple precision arithmetic library). Tato knihovna umožňuje práci s čísly s libovolnou přesností (arbitrary-precision). Omezení přesnosti je dáno pouze kapacitou operační paměti počítače. Vysoká přesnost je však vykoupena zvýšenými nároky na výkon počítače, [20] uvádí možnost zpomalení výpočtu až o 80%. Ve výkonnostních testech byla u knihovny CGAL použita floating-point aritmetika, aby byly výsledky srovnatelné s ostatními algoritmy.

Algoritmy PolyBoolean a Boolean oba používají podobný přístup – zaokrouhlení souřadnic vrcholů a průsečíků na celočíselnou mřížku s danou přesností. Tento přístup je považován za robustní.

4.6 Vyhodnocení

V této podkapitole je uveden přehled základních vlastností všech algoritmů a poté je pro každý z algoritmů uvedeno stručné vyhodnocení na základě provedených testů.

4.6.1 Přehled vlastností

Údaje uvedené v této tabulce jsou kombinací vlastností algoritmů uvedených samotnými autory algoritmu a vlastností zjištěných během testování.

Algoritmus	KK	SP	OT	OOT	VP	UZ	SD	VS	NR	AND	OR	SUB	XOR
Sutherland-Hodgman	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne	**Ano**	Ne	Ne	Ne
Weiler-Atherton	**Ano**	Ne	**Ano**	Ne	Ne	Ne	Ne	Ne	Ne	**Ano**	Ne	**Ano**	Ne
Greiner-Hormann	**Ano**	**Ano**	Ne	Ne	Ne	Ne	Ne	Ne	Ne	**Ano**	**Ano**	**Ano**	Ne
GPC	**Ano**	**Ano**	**Ano**	**Ano**	**Ano**	**Ano**	**Ano**	**Ano**	Ne	**Ano**	**Ano**	**Ano**	**Ano**
Boolean	**Ano**	**Ano**	Ne	**Ano**	Ne	**Ano**	**Ano**	**Ano**	**Ano**	**Ano**	**Ano**	**Ano**	**Ano**
ClipPoly	**Ano**	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne	**Ano**	Ne	**Ano**	Ne
PolyBoolean	**Ano**	Ne	**Ano**	Ne	**Ano**	**Ano**	**Ano**	**Ano**	**Ano**	**Ano**	**Ano**	**Ano**	**Ano**
CGAL	**Ano**	Ne	**Ano**	**Ano**	**Ano**	**Ano**	**Ano**	**Ano**	**Ano**	**Ano**	**Ano**	**Ano**	Ne

Tabulka 3: Přehled vlastností implementací algoritmů

KK – Vstupní polygony mohou být konkávní

SP – Vstupní polygony mohou být sebeprotínající

OT – Vstupní polygony mohou obsahovat otvory tvořené vnořeným obrysem

OOT – Vstupní polygony mohou obsahovat otvory vytvořené obrysem polygonu

VP – Vstupem může být více polygonů (Polygon Set)

UZ – Algoritmus je uzavřený nad booleovskými operacemi, tzn. že výstupní data mohou být i vstupem algoritmu

SD – Vstupní polygon se může dotýkat sebe sama

VS – Vstupní polygony mohou obsahovat vrcholy vyššího stupně

NR – Algoritmus je numericky robustní

AND, OR, SUB, XOR – Algoritmus umožňuje provést uvedené booleovské operace

4.6.2 Sutherland-Hodgman

V jednoduchosti algoritmu Sutherland-Hodgman je síla. Ve většině testů byl nejrychlejší, při ořezání polygonu obdélníkem až o několik řádů. Omezení vstupních polygonů jsou však silná, algoritmus navíc umí pouze booleovskou operaci průnik. Proto algoritmus nelze doporučit pro provádění booleovských operací. Sutherland-Hodgman je však dobře použitelný v aplikacích, kde je potřeba ořezávat konvexním polygonem (nejlépe obdélníkovým oknem) a kde je žádoucí efektivita na úkor kvality výsledků, což může být např. rendering.

4.6.3 Weiler-Atherton

Algoritmus Weiler-Atherton je relativně jednoduchý a rychlý, není však dostatečně robustní pro reálné nasazení v aplikaci, která vyžaduje provádění booleovských operací nad polygony. Weiler-Atherton je pro svou názornost hojně využíván pro studijní účely.

4.6.4 Greiner-Hormann

Algoritmus Greiner-Hormann neumí zpracovat otvory a má ve své základní verzi problémy s degenerativními případy. S rozšířením umožňujícím vychýlení vrcholů je algoritmus robustnější, avšak nedeterministický. Možnost zásadního vylepšení tohoto algoritmu spočívá v implementaci rozšíření uvedeného v části 3.3.2.2. V rychlosti výpočtu je Greiner-Hormann zhruba na stejné úrovni jako Weiler-Atherton.

4.6.5 GPC

GPC je dle mého názoru nejlepší z množiny testovaných algoritmů. Poradí si se všemi speciálními případy. Zvládá všechny typy booleovských operací. I přes svou komplexnost je velmi rychlý, v testech 4.3.1 a 4.3.2 dokonce nejrychlejší ze všech. K rychlosti jistě přispívá i implementace v jazyce C, tedy bez použití objektově orientovaného přístupu. Integrace do aplikací je snadná díky existenci wrapperů pro mnoho programovacích jazyků. Jako jediná nevýhoda se může jevit absence numericky robustní aritmetiky. GPC je vhodný pro všechny typy aplikací.

4.6.6 Boolean

Knihovna Boolean je velmi komplexní a umí korektně zpracovat všechny speciální případy. U některých výstupů se však objevují nadbytečné vrcholy. Algoritmus však neumí pracovat s otvory, zadanými vnořeným obrysem. V testech rychlosti výpočtu Boolean znatelně zaostával. Jako jeden z mála je numericky robustní.

4.6.7 ClipPoly

Knihovna ClipPoly je implementací základní verze algoritmu Klamera Schutte. Neumí zpracovat otvory ani sebeprotínající a ani sebedotýkající polygony, což jsou značná omezení vstupních dat. Rychlost výpočtu pro nízký počet vrcholů je průměrná, pro vysoký počet vrcholů (>1000) ovšem doba výpočtu strmě stoupá.

4.6.8 PolyBoolean

Ruskou knihovnu PolyBoolean řadím v závěrečném vyhodnocení na druhé místo za GPC. Neumí sice zpracovat sebeprotínající polygony, v ostatních testech si však vedla dobře. V testech rychlosti se kvalita algoritmu začala projevovat až u vyššího počtu vrcholů, kde se knihovna rychlostí blížila ke GPC. Knihovna umožňuje zpracovat otvory a má kvalitně vyřešené zpracování vrcholů vyššího řádu. Numerická robustnost je zajištěna zaokrouhlením na celočíselnou mřížku. Nevýhodou je, že nejnovější verze knihoven jsou dostupné pouze v komerční verzi.

4.6.9 CGAL

CGAL se jako jediný v testu může pochlubit aritmetikou s volitelnou přesností. Kromě sebeprotínajících polygonů si poradil se všemi speciálními případy. Doba výpočtu není předností této knihovny. Pro určité aplikace může jistě být výhodou, že CGAL umí zpracovat i polygony, jejichž hrany nejsou úsečky, ale křivky. Pro studijní účely je nevýhodné, že algoritmus použitý v této knihovně není vůbec zdokumentován.

5 Závěr

Tato diplomová práce se zabývala analýzou algoritmů pro booleovské operace nad obecnými polygony. V následujících odstavcích je uvedeno krátké shrnutí obsahu práce a dosažených výsledků.

V první části práce byly vysvětleny základní pojmy týkající se polygonů a byl definován pojem obecný polygon. Dále byly názorně popsány základní booleovské operace nad polygony a byl vysvětlen pojem regularizované booleovské operace.

Poté, co byly vymezeny základní pojmy, byl vysvětlen princip vybrané podmnožiny algoritmů pro booleovské operace nad polygony. Zvláštní pozornost byla věnována algoritmům Weiler-Atherton a Greiner-Hormann, u kterých byly též popsány možnosti řešení speciálních případů. V závěru kapitoly byly uvedeny dostupné implementace uvedených algoritmů a chronologický přehled vzniku algoritmů.

Další část diplomové práce se věnovala komplexnímu srovnání implementací. Dva z popsaných algoritmů, Weiler-Atherton a Greiner-Hormann, byly naimplementovány v Jazyce C++. Spolu s ostatními implementacemi byly umístěny do dynamické knihovny PolygonClipper. Byl vytvořen komunikační kanál v podobě XML souboru mezi knihovnou PolygonClipper a aplikací DXF Editor, která umožňuje pohodlnou editaci geometrie testovacích polygonů. Tím bylo vytvořeno vhodné testovací prostředí pro komplexní testování algoritmů a vizualizaci výsledků.

Implementace algoritmů byly porovnány z hlediska doby výpočtu a schopnosti zpracovat speciální případy. Dále byla porovnána omezení vstupních dat a množina prováděných booleovských operací. Všechny vlastnosti algoritmů jsou přehledně uvedeny ve srovnávací tabulce.

Vítězem srovnávací části práce se stala volně dostupná knihovna GPC, která je dostatečně robustní a pracuje velmi rychle. Jedinou nevýhodou je nemožnost použití aritmetiky s volitelnou přesností. Pokud je pro cílovou aplikaci přesnost výpočtů zásadní, pak připadá v úvahu knihovna CGAL.

Za přínos práce považuji nastudování mnoha různých zdrojů a vytvoření uceleného pohledu na problematiku booleovských operací nad obecnými polygony. Dále také vytvoření efektivního testovacího prostředí, které je možné případně rozšířit o další implementace algoritmů.

Další možnost rozvoje práce vidím v implementaci nedávno zveřejněného rozšíření algoritmu Greiner-Hormann a poté v detailním nastudování a implementaci algoritmů založených na teorii simplexů. Tyto algoritmy jsou podloženy matematickým formalismem a přistupují k problému reprezentace polygonů a booleovských operací nad nimi ze zcela jiného úhlu než dosud publikované algoritmy.

Literatura

[1] *Wikipedie, otevřená encyklopedie*, česká verze. URL: http://cs.wikipedia.org

[2] *Wikipedia, the free encyclopedia*, anglická verze. URL: http://en.wikipedia.org

[3] Žára, J., Beneš, B., Sochor J., Felker, P.: Moderní počítačová grafika, Computer Press, 2004.

[4] Sutherland, I. E., Hodgman, G. W.: *Reentrant Polygon Clipping,* Communications of the ACM, Vol. 17, 1974, 32 – 42

[5] Weiler, K., Atherton, P.: *Hidden surface removal using polygon area sorting*. ACM SIGGRAPH Computer Graphics, Vol. 11, 1977, 214 - 222.

[6] Priester, S.: *Polygon clipping*, www.codeguru.com, 2005.
URL: http://www.codeguru.com/Cpp/misc/misc/graphics/article.php/c8965/ (Prosinec 2007)

[7] Greiner, G., Hormann, K.: *Efficient Clipping of Arbitrary Polygons*, ACM Transactions on Graphics, Vol. 17, 1998, 71-83

[8] Sunday, D.: *Fast Winding Number Inclusion of a Point in a Polygon*.
URL: http://www.geometryalgorithms.com/Archive/algorithm_0103/algorithm_0103.htm (Prosinec 2007)

[9] Kim, D.H., Kim, M.-J.: *An extension of Polygon Clipping To Resolve Degenerate Cases*, Computer-Aided Design & Applications, Vol. 3, 2006, 447-456

[10] Vatti, B.R.: *A Generic Solution to Polygon Clipping*, Communications of the ACM, Vol. 35, 1992, 56-63

[11] Schutte, K.: *An edge labeling approach to concave polygon clipping*, ACM Transactions on Graphics, 1995

[12] Holwerda, K.: *Complete Boolean Description*.
URL: http://boolean.klaasholwerda.nl/algdoc/top.html (Prosinec 2007)

[13] Leonov, M.V., Nikitin, A.G.: *An Efficient Algorithm for a Closed Set of Boolean Operations on Polygonal Regions in the Plane (draft English translation),* A. P. Ershov Institute of Informatics Systems, Preprint 46, 1997.
URL: http://www.complex-a5.ru/polyboolean/downloads/polybool_eng.pdf (Prosinec 2007)

[14] Liang, Y.-D., Barsky, B.A.: *An Analysis and Algorithm for Polygon Clipping,* Communications of the ACM, Vol. 26, 1983, 868-877

[15] Weiler K.: *Polygon comparison using a graph representation*, SIGGRAPH '80 Conference Proceedings, Computer Graphics, Vol. 14, 1980, 10-18

[16] Rivero M., Feito, F.R.: *Geometric modelling based on simplicial chains*, Computers and Graphics Vol. 22, 1998, 611-619

[17] Rivero M., Feito, F.R.: *Boolean operations on general planar polygons*, Computers and Graphics Vol. 24, 2000, 881-896

[18] Peng, Y., Yong, J.-H., Dong, W.-M., Zhang, H., Sun, J.-G.: *A new algorithm for Boolean operations on general polygons*, Computers and Graphics Vol. 29, 2005, 57-70

[19] Leonov, M.: *Comparison of the different algorithms for Polygon Boolean operations*, 1998. URL: http://www.complex-a5.ru/polyboolean/comp.html (Prosinec 2007)

[20] *The CGAL Philosophy.*
URL: http://www.cgal.org/philosophy.html (Květen 2008)

[21] Daněk, T.: *DXF Editor* [Bakalářská práce], FIT VUT Brno, 2006

Seznam příloh

Příloha 1: Knihovny – licence, dostupnost

General Polygon Clipper library

Autor: Alan Murta, University of Manchester

Jazyk implementace: C

Dokumentace: Knihovna není příliš dobře zdokumentována.

Distribuce: Knihovna je distribuována ve formě zdrojového kódu. Knihovnu lze díky wrapperům použít v mnoha programovacích jazycích a prostředích: C, C#, Delphi, Java, Perl, Python, Haskell, Lua, VB.Net a další

Licence: GPC je volně použitelná pro nekomerční účely, v případě komerčního využití je nutné získat licenci od University Of Manchester

Ke stažení: http://www.cs.man.ac.uk/~toby/alan/software/ (Poslední verze: 2.32, 17.12.2004)

Boolean, a C++ library

Autor: Klaas Holwerda

Jazyk implementace: C++

Dokumentace: Algoritmus je velmi dobře zdokumentován.

Distribuce: knihovna je distribuována ve formě zdrojového kódu

Licence: Knihovna je volně použitelná pro nekomerční účely. V případě komerčního využití je požadováno informování uživatele o tom, že aplikace používá knihovnu Boolean, např. v dokumentaci aplikace a v dialogu O Aplikaci.

Ke stažení: http://boolean.klaasholwerda.nl/bool.html (Poslední verze: 6.97, 28.6.2007)

ClipPoly

Autor: Klamer Schutte, University of Twente

Jazyk implementace: C++

Dokumentace: Algoritmus je zdokumentován v [10]

Distribuce: Ve formě zdrojového kódu

Licence: Open Source od roku 2005

Ke stažení: http://clippoly.sourceforge.net/ (Poslední verze: pl11, 11.3.2005)

PolyBoolean

Autoři: Michael Leonov, Alexey Nikitin z ruské akademie věd

Jazyk implementace: C++

Dokumentace: Stručně na webových stránkách knihovny, algoritmus je zdokumentován v [12]

Distribuce: Knihovna je distribuována v podobě knihovny tříd (DLL) pro .NET, nebo v podobě zdrojového kódu

Licence: Zdrojový kód je volně k použití pro nekomerční účely. Standardní komerční licence pro .NET stojí 265 Eur (V evaluation verzi je omezen počet vrcholů ve výsledku booleovské operace), Standardní komerční licence pro C++ stojí 235 Eur.

Ke stažení: http://www.complex-a5.ru/polyboolean/index.html (Poslední verze: 8.5.2006 .NET 2.0).

CGAL

Autoři: Komunita vývojářů

Jazyk implementace: C++

Dokumentace: Knihovna CGAL je velmi dobře zdokumentována

Distribuce: V podobě zdrojových kódů, ze kterých lze sestavit dynamické knihovny. Operací s polygony se týká modul *Boolean_set_operations_2*

Licence: Knihovna je OpenSource, komerční licenci je nutné zakoupit

Ke stažení: http://www.cgal.org/ (Poslední verze: 3.3.1, 25.8.2007)

Wykobi

Autor: Arash Partow

Jazyk implementace: C++

Dokumentace: Není k dispozici

Distribuce: V podobě zdrojových kódů

Licence: Nekomerční použití - GPL (version 2), jinak Wykobi Commercial License

Ke stažení: http://www.wykobi.com/downloads.html (Poslední verze: 0.0.3, 1.6.2007)

Příloha 2: Obsah CD a kompilace

Přiložené CD obsahuje data uložená v následující adresářové struktuře:

bin

└─ DXFEditor

  ├─ DXFEditor.exe - Aplikace DXF Editor

  └─ PolygonClipper.dll – Dynamická knihovna PolygonClipper

data

├─ 43 Performance – Testovací data ke kapitole 4.3

└─ 44 SpecialCases – Testovací data ke kapitole 4.4

doc

├─ DP_xdanek09.pdf (Text diplomové práce)

└─ DXFEditor User Reference.pdf (Uživatelská příručka DXF Editoru v angličtině)

src

├─ DXFEditor – Zdrojové soubory aplikace DXF Editoru

└─ PolygonClipper – Zdrojové soubory dynamické knihovny PolygonClipper

Zdrojové kódy aplikace DXF Editor jsou distribuovány jako VC# projekt pro MS Visual Studio 2005.

Zdrojové kódy dynamické knihovny PolygonClipper jsou distribuovány jako Visual C++ projekt pro MS Visual Studio 2005. Pro sestavení projektu je nutné nainstalovat knihovnu CGAL s odpovídajícími předkompilovanými knihovnami a povolit nastavení cest v proměnném prostředí OS Windows.

Příloha 3: Použití testovacího prostředí

Pro správné fungování testovacího prostředí je nutné zkopírovat adresář DXFEditor z adresáře *bin* na přiloženém CD na pevný disk. Aplikace je spustitelná na OS Windows. Ke spuštění aplikace je nutný .NET Framework verze 2.0 a vyšší (Instalační soubor .NET Framework je v adresáři *bin*).

Začátek práce

Spustit aplikaci *DXFEditor.exe*

Otevřít některý z testovacích souborů (přípona .xml) z adresáře *data* nebo vytvořit nový výkres.

Vytvoření nového výkresu

Nabídka Soubor -> Nový výkres

Správce hladin -> Přidat hladiny s názvem „subject“ a „clip“

Nakreslení obdélníku nebo polyline, případná editace souřadnic vrcholů nebo přiřazení parametru *Otvor* v okně *Vlastnosti*. PolygonClipper umí zpracovat pouze jeden hlavní obrys v každé vrstvě.

Geometrii lze též importovat z formátu DXF.

Vytvořený výkres lze uložit ve formátu XML nebo DXF.

Popis funkcí a nastavení v nabídce PolygonClipper

Spustit XX – Spuštění algoritmu. Po provedení algoritmu se doba výpočtu zobrazí ve stavové liště aplikace

Počet opakování: X – Nastavení počtu cyklů provedení algoritmu

Průměr z 5-ti spuštění – Knihovní funkce bude zavolána 5x po sobě a bude vypočten průměr doby trvání vykonávání (při každém volání funkce je proveden zadaný počet opakování)

Automaticky načíst výsledek – Po vykonání booleovské operace bude automaticky založen nový výkres, který bude obsahovat vrstvy původního výkresu a nové vrstvy s výslednými polygony

Vykreslit výplně – Vykreslí poloprůhledné výplně polygonů

Generovat polygon – Umožňuje vygenerovat náhodný polygon typu krystal nebo typu pravidelný n-úhelník. Je možné zadat rozměry, počet vrcholů a rozptyl. Vygenerovaný polygon bude vložen do nové vrstvy

Uložit bitmapu – Umožňuje uložit obsah pohledového okna do souboru s bitmapou

Průnik/Sjednocení/Rozdíl – Umožňuje nastavit typ prováděné booleovské operace