Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

DIPLOMOVÁ PRÁCE

Plzeň, 2007

Vojtěch Hladík

Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

2D editor pro návrh topologie čipu - projekt FlashPoM - 1

Plzeň, 2007 Vojtěch Hladík

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 17.5.2007, Vojtěch Hladík, ……

Poděkování

Rád bych poděkoval Prof. Ing. Václavu Skalovi, CSc. za vedení mého studia a mé diplomové práce, Ing. Janu Kaiserovi za řízení projektu FlashPoM a Františku Novákovi za jeho spolupráci při vytváření našeho editoru. Poděkování patři také mé rodině a přátelům za jejich podporu a porozumění.

Práce byla podporována projekty Mateo – FlasPoM, EU INTERREG IIIC a projektem MŠMT ČR , projekt 2C 06002 Virtual.

Abstract

FlashPoM project is focused on development of techniques for low-cost low lead-time production of customized modular lab-on-a-chip prototypes devices by optimization of the PoM (Polymer-on-Multielectrode array) technology.

This work describes one part of this project, which is to create a low-complexity software tool for chip configuration by combination of a set of predefined elements and modules. These will include polymeric layers with embedded microchannels, vias, microchambers and solid layers with electrode arrays of customizable geometries. Software will also be able to estimate the production cost for a predefined batch size in dependence on selected modules, their arrangement and customized geometrical properties.

Obsah

1. Úvod

Předložená diplomová práce popisuje výsledek spolupráce naší univerzity na mezinárodním projektu FlashPoM. Hlavním cílem tohoto projektu je nalezení nových postupů, které by umožňovaly snížení finančních a časových nákladů při konstrukci neuročipů využívajících PoM (Polymer-on-Multielectrode array) technologii.

Základem takovéhoto neuročipu je vrstva substrátu s polem elektrod pro měření vlastností neuronů. Na substrát jsou naneseny jednotlivé vrstvy různých materiálů, které obsahují komory pro usazování neuronů a kanálky pro růst neuronových spojů. Naším úkolem v tomto projektu je vyvinout jednoduchý programový nástroj pro návrh těchto čipů kombinováním souboru předdefinovaných elementů a modulů.

Vytvořený editor má uživateli poskytnout možnost na vytvořené vrstvy zakreslovat oblasti, které mají být opracovány laserem. Tyto oblasti mají být vyznačovány pomocí základních primitiv, jako je obdélník, elipsa a čára, nebo pomocí složitějšího objektu cesta. Při práci s těmito objekty má mít uživatel k dispozici základní nástroje k jich editaci a distribuci. Editor má také obsahovat některé specializované nástroje, jako je ověření návrhu, odhad ceny a 3D zobrazení.

Vzhledem k rozsahu celého projektu, předložená práce obsahuje pouze oblasti popisující řízení aplikace, GUI, 3D export a implementaci elementu cesta. Na projektu se po vedením Prof. Ing. Václava Skali, CSc. dále podílel Ing. Jan Kaiser, který měl na starosti analýzu, komunikaci s budoucími uživateli softwaru, implementaci některých oblastí editoru (materiály, cut view, zoom,...) a František Novák, který řešil především datovou oblast, kontrolu návrhu a 3D zobrazení. Pro úplné pochopení funkčnosti aplikace je proto vhodné přečíst i diplomovou práci Františka Nováka.

Od čtenáře se nevyžaduje hluboká znalost technologie Microsoft .NET a rozhraní Microsoft Direct3D for Managed code, které náš editor využívá. Rovněž znalost objektově orientovaného programování není nutností, ovšem pro lepší pochopení některých kroků by čtenář měl mít alespoň některé základní znalosti v oblasti vizualizace dat.

2. Teoretická část

V teoretické část této diplomové práce je popsán cíl evropského projektu MATEO a jeho podprojektu FlashPoM, na kterém se naše univerzita podílela. Obsahuje vysvětlení, čím se projekt FlashPoM zabývá, jak se vyrábí PoM technologie a jaké bylo zadání naší práce.

2.1. MATEO [i]

Hlavním cílem projektu MATEO (Matching Technologies and Opportunities) je návrh a realizace dílčích projektů, které mají rozvinout inovační procesy v malých a středních firmách (MSP) a ve výzkumných a vývojových institucích. Podporovat výměnu zkušeností a zajišťovat spolupráci mezi partnerskými regiony v různých oblastech zájmu.

Participující regiony v projektu MATEO jsou:

1. Španělsko: Katalánsko (řídící region zastoupený organizací Catalunya Innovacio – CIDEM)
2. Itálie: Lombardie
3. Holandsko: Severní Brabantsko (Noord Braband)
4. Česká republika: NUTS II Jihozápad

Cílem není pouze výměna teoretických znalostí založených na praktických poznatcích v oblasti zájmu, ale i vytvoření konkrétních technologií a sektorově orientovaných dílčích projektů. Tyto projekty by k sobě měli přivádět odborníky s různých částí světa za účelem společně vytvářet průmyslově využitelné produkty z RTD (Research and Technology Development) znalostí.

Takto zaměřená mezioblastní kooperace se snaží především eliminovat různá omezení, která brání optimální interakci mezi výzkumem a průmyslem. Tyto omezení jsou většinou spojena s faktem, že obě skupiny produkují nové RTD poznatky, nevědomí si toho, že již existují. Mezioblastní kooperace umožňuje tyto znalosti lépe rozšířit a tím zvýšit šanci jejich efektivního využití.

Další cíle projektu MATEO jsou:

- zmapování inovačního potenciálu regionu, diskuse o relevantních částech regionálních inovačních strategií mezi všemi partnery projektu, včetně výměny zkušeností a návrhů optimalizace těchto strategií,
- příprava podprojektů firem či výzkumných pracovišť, které budou podporovat inovační rozvoj sektorů, případně podpora další speciální aplikace výzkumných výsledků.

[i] Převzato z [Mateo]

Dílčí projekty MATEO se zaměřují na osm tématických oblastí, kde čtyři hlavní odvětví jsou potravinářství, farmacie, letectví a obnovitelné zdroje energie a zkoumané technologie jsou biotechnologie, pokročilé materiály, mechatronika a výrobní a produkční technologie.

2.2. FlashPoM

Jedním z podprojektů MATEO je FlashPoM. Oblastí zájmu tohoto projektu je zjednodušení práce s mikrostrukturními polymery, což sleduje cíle jako je zlevnění a urychlení výroby a pružnější reakce na požadavky trhu při používaní zařízení s PoM technologií. Splnění těchto cílů by měly umožnit zkoumané procesy přípravy modulových zařízení vybavených komplexním softwarovým řešením, litografickými systémy a optimalizací volby polymerních materiálů pro danou výrobu. Zásadním prvkem pro předpokládané použití je i hodnocení biokompatibility zvolených materiálů.

Hlavním cílem projektu je tedy snížení nákladů a časové náročnosti výrobních postupů modifikovatelných neuročipů optimalizací PoM (Polymer-on-Multielectrode array) technologie.

Čtyři hlavní úkoly projektu jsou:

1. Vývoj softwaru podporující snadný návrh modifikovatelných PoM zařízení, jehož nástroje by byly navrženy s ohledem na výrobní procesy.

   Tým má za úkol vyvinout jednoduchý programový nástroj pro použití MSP ke konfiguraci jejich prototypů kombinováním souboru předdefinovaných elementů a modulů. Ty budou obsahovat polymerové vrstvy se začleněnými mikrokanálky, cestami, mikrokomorami a pevnými vrstvami s polem elektrod, které mají modifikovatelné rozložení. Podle zadaných parametrů jednotlivých modulů, jejich rozmístění a modifikace jejich tvaru, bude program rovněž schopen odhadnout výrobní náklady, případně určit, zdali čip lze zkonstruovat. Koncový návrh softwaru s optimální funkcionalitou bude uložen v kompaktním souboru vhodném pro distribuci přes internet.

2. Složení prototypu litografického systému s přímým zápisem pro nemaskovanou poloautomatickou produkci elektrodového pole s litografickým základem.

   Výroba mikrofluidika, mikrosenzorů a mikroovladačů, která je obvykle vyžadována pro sestavení laboratorních systému vyžaduje intenzivní využití fotolitografických technik. Konvenční fotolitografické procesy vyžadují nákladné (zhruba 2000 euro/substrát) skleněné substráty produkované e-beam technologií, které poskytují požadované rozlišení 2 μm. Alternativa k tomuto standardnímu postupu je založena na přímém zápisu prvků laserem na fotocitlivý materiál. Ten ukazuje schopnost produkovat vysoce kvalitní prvky s rozlišením k 10 μm s mnohem menšími cenovými náklady (zhruba 20 euro/substrát). Takovýto zápis laserem sice poskytuje menší rozlišení, ovšem to je stále dostačující k produkci velké škály laboratorních prototypů v poloautomatickém přístupu.

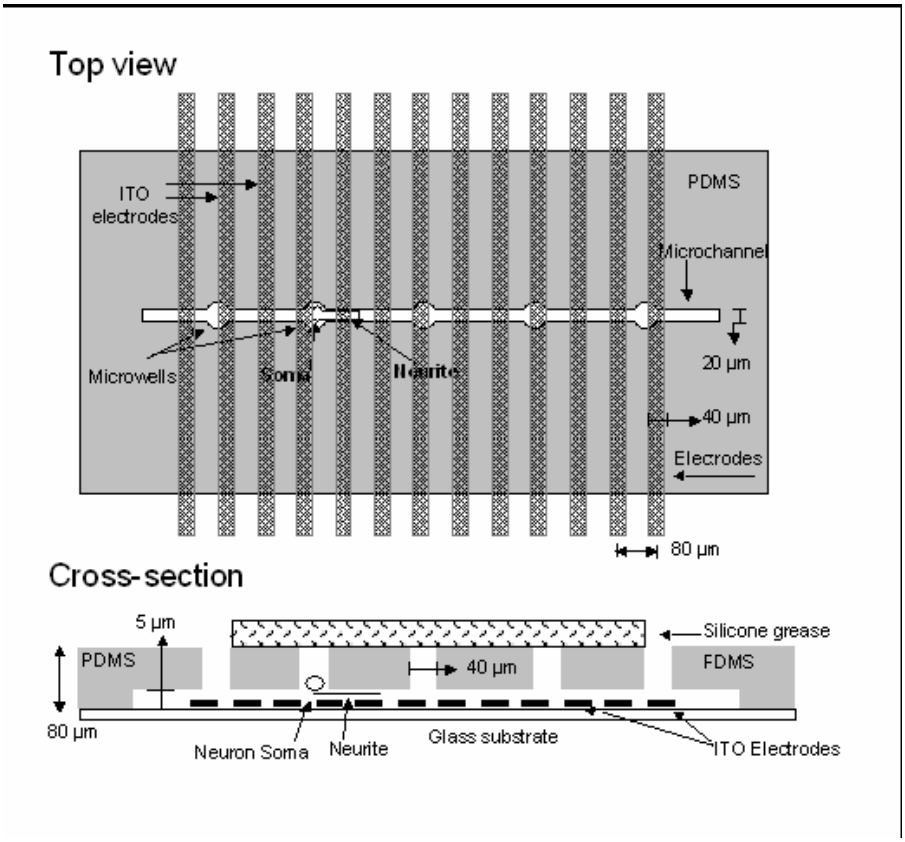
3. Optimalizace biokompatibility a fotocitlivosti polymerových materiálu pro výrobu neuročipů technologií navrženou v projektu FlashPoM

4. Vyhodnocení využitelnosti FlashPoM zařízení pro společnost, diferenciace a elektrotechnická charakteristika základních neuronů.

Cílovým trhem tohoto produktu jsou především menší a střední společnosti zabývající se výrobou miniaturizovaných zařízení v oblasti chemie a biologie, výrobci medicínských přístrojů a elektrotechnických součástek a široké pole společností a organizací s aplikacemi v medicíně a biotechnologiích. Těm všem by se díky snížení finanční a časové náročnosti výroby PoM zařízení měla otevřít případná cesta, jak proniknout na tuto část trhu, která pro ně kvůli složitosti a nákladnosti této technologie zůstávala doposud uzavřená. To může vést nejen k vytvoření alternativ k současných výrobcům, ale i k vytvoření zcela nových odvětví využívající tuto technologii.

2.3. PoM Neuročip [ii]

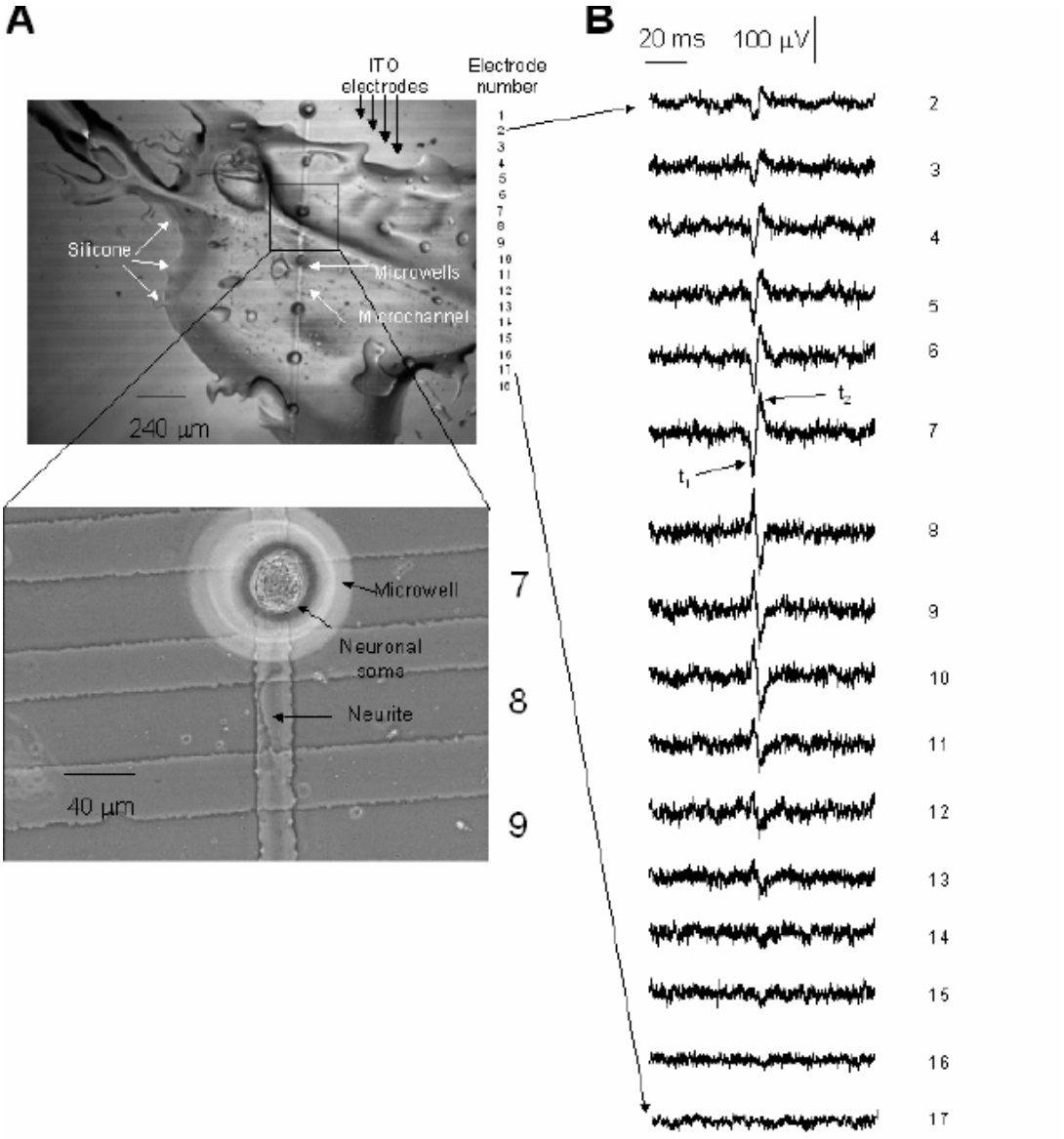
Použití technologie multielektrodových polí ke stavbě neuročipu - zařízení pro charakterizaci aktivit mezi jednotlivými neurony (obr.2.3.1), je slibnou alternativou k doposud používaným plošným metodám .



Obrázek 2.3.1 Návrh neuročipu, převzato z [Clavero05]

[ii] Převzato z [Clavero05]

Multielektrodové pole je soustava paralelních ITO (indium-cín-oxid) elektrod umístěných na skleněném substrátu. Na tomto elektrodovém poli je umístěna elastomerická vrstva (PDMS – Polydimethylsiloxane), která v sobě obsahuje otvory pro vsazení těl neuronů a kanálky pro vytvoření neuronových spojů (obr. 2.3.2-A). Během růstu těchto spojů přicházejí neurony do kontaktu s jednotlivými elektrodami stimulujícími a měřícími jejich potenciálové pole (2.3.2-B).



Obrázek 2.3.2.A: Výřez neuročipu; B: naměřené hodnoty; převzato z [Claver05]

Výhodou popsané technologie oproti doposud používaným plošným metodám, kde růst neuronů nebyly řízen kanálky v PDMS vrstvě, je především možnost jednoznačně identifikovat zdroj naměřené aktivity a stabilnější elektrická vazba mezi neuronovým spojem a elektrodou.

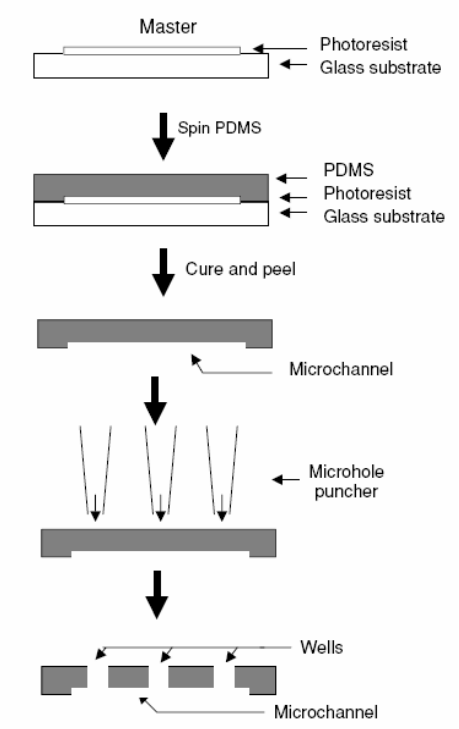
2.4. Současná výroba neuročipu[iii]

Současny proces výroby začíná nanesením fotorezistu (látka citlivá na světlo) na skleněný substrát. Požadovaná tloušťka rezistu 6μm je docílena rotací substrátu, kdy odstředivá síla rozptýlí kapalný rezist do stran. Na této vrstvě je poté fotolitografickou metodou vytvořen vzor určující místa pro vsazení neuronů a růst neuronových spojů.

Druhým krokem výroby je nanesení vrstvy elastomeru (PDMS) o tloušťce 150μm. Způsob dosažení požadované tloušťky je stejný jako u rezistu. Po vytvrzení je PDMS sloupnut a v místech pro vložení neuronů jsou proraženy otvory.

Posledním krokem výroby je sesazení PDMS a multielektrodového pole. Při sesazování mohou být použity dvě konfigurace. V prvním případě jsou elektrody umístěny přímo pod těly jednotlivých neoronů. Tento postup vyžaduje přesné sesazení v rozmezí několika mikrometrů. Při druhé konfiguraci je PDMS pouze přiložen na substrát a rozmístění elektrod je zcela náhodné.

Před samotným umístěním neuronů musí být ještě neuročip vyplněn speciálním roztokem, který umožňuje neuronům přežití a růst.



Obrázek 2.4.2: Výroba neuročipu, převzato z [Clavero04]

V rámci projektu FlashPoM jsou v současnosti vyvíjeny nové technologie umožňující snížení časové a finanční náročnosti tohoto postupu. Jendou z těchto technologií bude i speciální editor poskytující výstup pro průmyslovou výrobu.

[iii] Převzato z [Clavero04]

2.5. Požadovaný software

Jak bylo v popisu projektu FlashPoM napsáno, jedním z jeho cílů je i vytvoření softwaru poskytující uživateli možnost navrhovat čip vrstvu po vrstvě. Tento software budou následně využívat další účastníci projektu. Naším úkolem je, jim tento software dodat. Editor se má řídit následujícími pravidly:

- Základem každého navrhovaného čipu je vrstva substrátu, se kterou je spojen druh materiálu, rozměry, tvar a tloušťka. Na vrstvu substrátu má být možnost nanášet jednotlivé vrstvy různých materiálů. Pokud vrstva patří do skupiny elektrod, může se jednat o skleněnou, nebo vodivou vrstvu. Dalším možným typem vrstev budou mikrofluidika. Každý čip může být složen z několika vrstev různých druhů, které jsou postupně skládány na sebe.

- Na vytvořené vrstvy bude uživatel moc zakreslovat oblasti, které mají být opracovány laserem. Tyto oblasti budou vyznačovány pomocí základních primitiv, jako je obdélník, elipsa a čára, nebo pomocí složitějšího objektu cesta. Při práci s těmito objekty bude mít uživatel k dispozici základní nástroje k jejich editaci a distribuci.

- Editor má dále poskytnout možnost kontroly čipu a odhadu výrobních nákladů poskytující uživateli základní informace o kvalitě návrhu. Pravidla pro kontrolu vyrobitelnosti čipu zaručují, že lze čip vyrobit pomocí technologií vyvíjených v rámci projektu FlashPoM. Výstupem editoru bude rozšířený formát SVG, jenž přímo poskytuje informace pro výrobu. Další schopností editoru má být poskytnutí 2D náhledu na řez čipem a 3D zobrazení vytvořeného návrhu.

Pro co nejlepší splnění požadavků na výsledný software, bude program implementován jako vektorový editor, jehož vlastnosti se budou podobat doposud používaným editorům a dále bude přidávat novou funkčnost specifickou pro postup výroby.

2.5.1. Vektorová grafika

Pojem vektorová grafika označuje jeden ze dvou hlavních způsobů, jak na počítači pracovat s dvourozměrnou grafikou. Na rozdíl od rastrové grafiky, kde je scéna složena z jednotlivých bodů, které nemají žádnou spojitost s okolím, vektorová grafika používá k popisu scény jasně definovaná primitiva, jako jsou čára, mnohoúhelník, elipsa, nebo křivka. Ty pomocí různých parametrů umožňují popsat požadovaný tvar. Tento způsob je nejčastěji používán pro technické výkresy, počítačovou sazbu nebo tvorbu animací.

Hlavními výhodami vektorové grafiky oproti bitmapové je tedy možnost teoreticky neomezené změny velikosti bez zanesení nežádoucích artefaktů a možnost editovat jednotlivá primitiva nezávisle na ostatních elementech. Jestliže máme jednoduchou scénu, dá se mluvit i o menší paměťové náročnosti. S tímto samozřejmě přicházejí i nevýhody, jako náročnější správa elementů, složitější ukládání historie nebo samotné editování elementů. Velkým problémem je také převod z rastrového na vektorový obrázek.

2.5.2. Existující vektorové editory

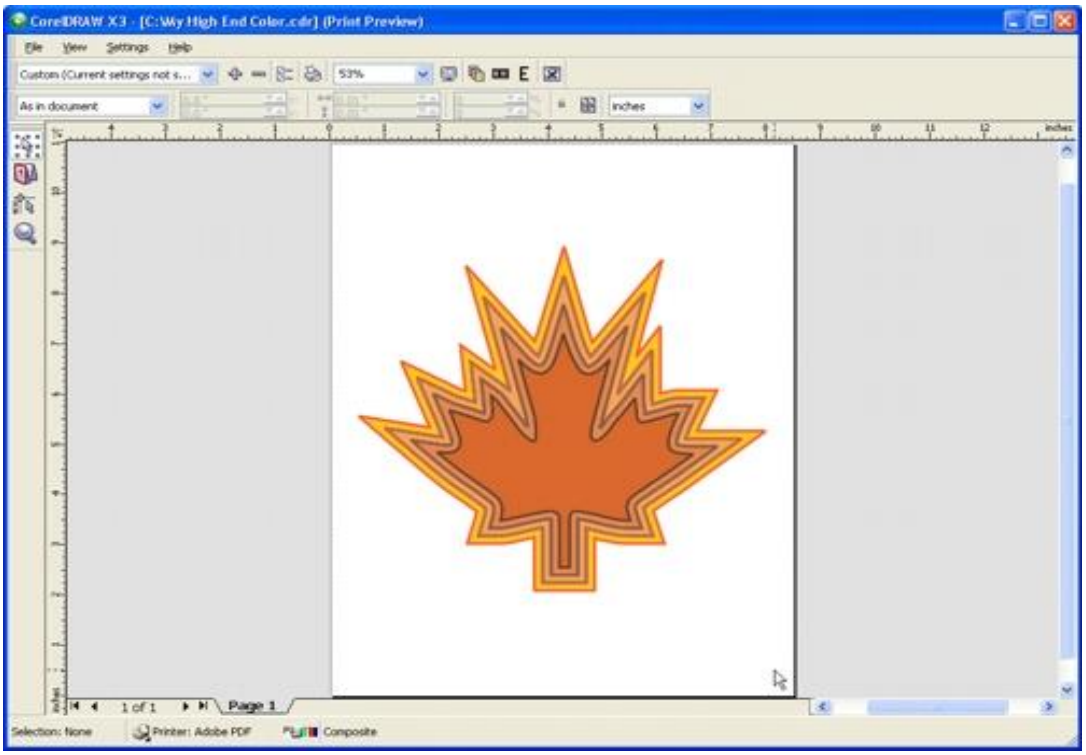
Ačkoliv v současné době existuje řada kvalitních vektorových editorů poskytujících velké množství různých nástrojů, jejich obecnost a zaměření na co nejširší oblast použití je činí nevhodnými a zbytečně složitými pro požadovaný návrh čipů.

Nejznámější vektorové editory:

- § CorelDraw,
- § Adobe ilustrátor,
- § Macromedia Freehand,
- § InkScape.

CorelDraw

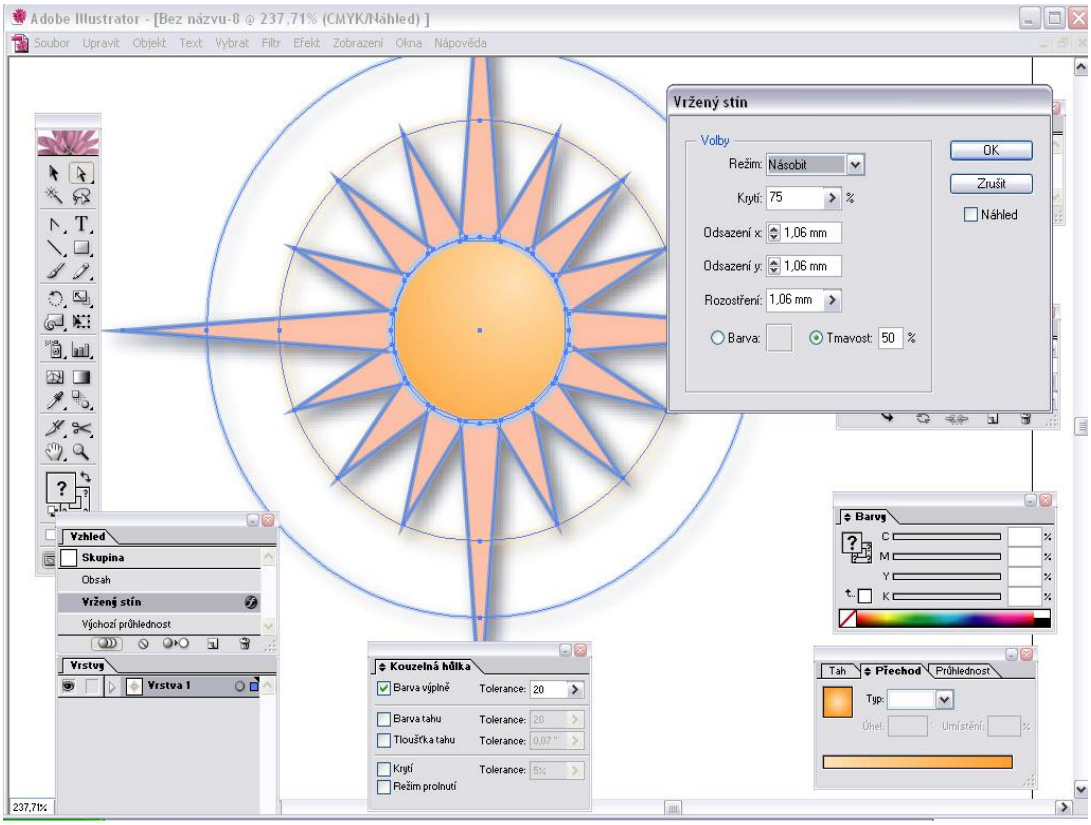
CorelDraw od společnosti Corel Corporation je komplexní softwarový balík obsahující soubor programů určených začínajícím grafikům i profesionálům. Vektorový editor CorelDRAW X3 společně s bitmapovým editorem Corel PHOTO-PAINT X3 obsahují nepřeberné množství nástrojů jednoduše použitelných například pro tvorbu firemní grafiky. Kromě klasických operací se základními elementy dokáže například automaticky vyhladit jednotlivé objekty, které můžete jen načrtnout a Corel z nich vytvoří přesné geometrické tvary.



Obrázek 2.5.3: CorelDraw

Adobe Illustrator

Adobe Illustrator CS2 je vektorový grafický editor od společnosti Adobe Systems. Kromě klasických nástrojů pro kreslení a editaci základních elementů umožňuje například snadno a rychle převést bitmapy na vektorové kresby a následně s nimi intuitivně pracovat. Úzká integrace s ostatními programy jako například Adobe Photoshop, umožňuje vytvářet zajímavé grafiky pro tisk, video, web či mobilní zařízení. V současné době se jedná o jeden z nejlepších komerční editorů na trhu, který o své prvenství soupeří s editorem CorelDraw.



Obrázek 2.5.2: Adobe Illustrator

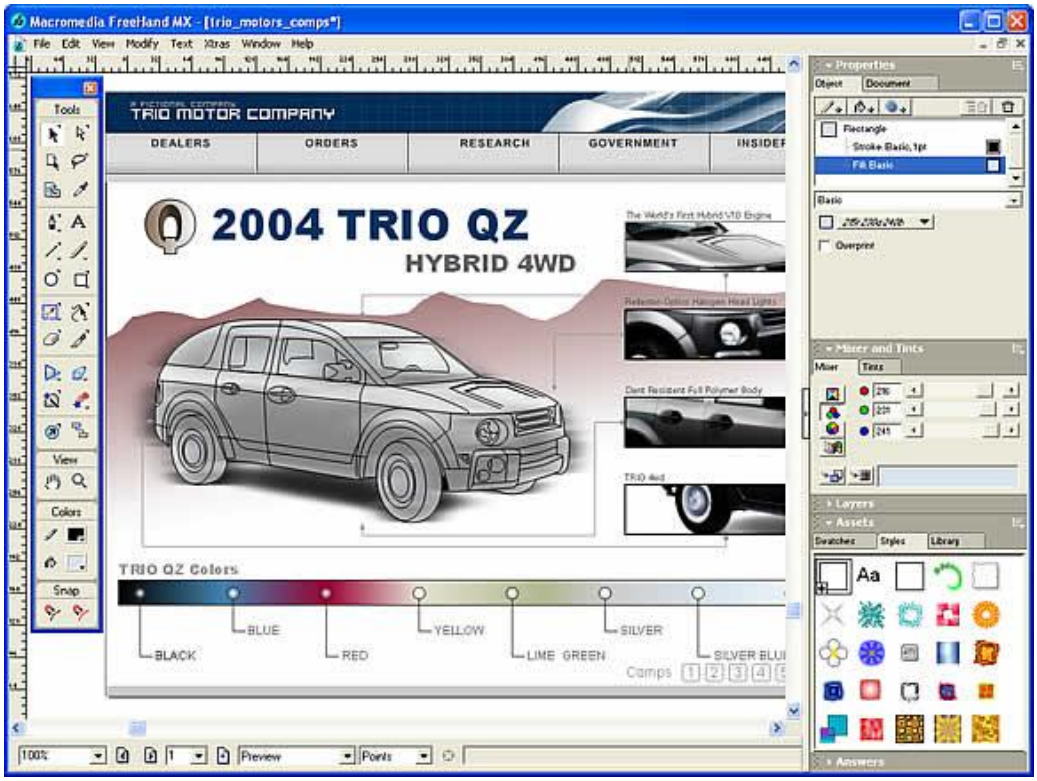
Některé z nadstandardních nástrojů Adobe Illustrator jsou:

- Podpora kompozic vrstev (layer comp) z Adobe Photoshopu
- Tvorba obsahu v mobilní verzi formátu SVG
- Kolorovaní černobílých obrázků
- Rozšířené možnosti tahu

Macromedia Freehand

FreeHand je další z produktů firmy Adobe. Poskytuje nástroje k tvorbě jednoduchých ilustrací i složitých grafických návrhů pro tisk i webové stránky. Kromě klasické vektorové grafiky si FreeHand poradí také s bitmapovými obrázky a vícestránkovou montáží.

Mezi hlavní funkce a nástroje patří podpora barevných standardů CMYK, RGB, HLS, Apple nebo Windows, knihovna přímých barev (Crayon, Dic-Pcn, Focolton, Greys, Mhic, Mun) a ukládání často používaných prvků do knihoven symbolů. Rovněž obsahuje sadu štětců a sprejů s možností úprav jejich stop sada grafických a textových stylů. Dále, stejně jako ostatní editory, podporuje například nástroj Snap To Object pro přichytávání k jiným objektům a některé základní 3D transformace objektů.

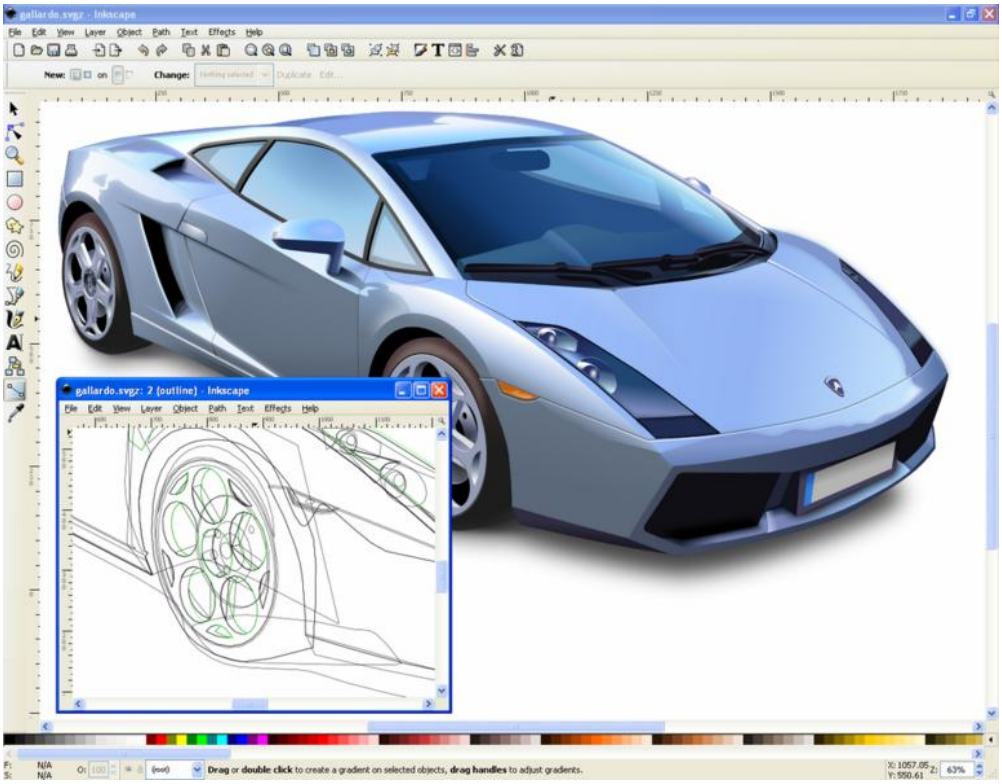


Obrázek 2.5.3: Macromedia Freehand

InkScape

Inkscape je open source vektorový grafický editor. Jeho cílem je stát se praktickým grafickým nástrojem, který bude plně odpovídat standardům XML, SVG a CSS. Inkscape je multiplatformová aplikace, která může běžet pod Microsoft Windows, Mac OS X a Unixovými operačními systémy, nicméně prvořadá vývojová platforma je operační systém Linux. Tento editor bohužel zatím neobsahuje tolik nástrojů a vlastností jako nejlepší komerční vektorové editory.

Kromě standardních nástrojů, vlastnosti kterými InkScape dosahuje úrovně komerčních editorů, jsou například maskování, práce s textem, ořezávání, nebo nastavování stylů. Rovněž jsou podporovány vrstvy nebo třeba matematické operace s vektorovými plochami (analogie cestáře z Illustratoru).



Obrázek 2.5.4: InkScape

2.4.3. Srovnaní s navrhovaným editorem

Výhodami popsaných editorů oproti našemu programu je především větší nabídka nástrojů pro nastavení barev a vzorů objektů, práce s textem, širší nabídka základních elementů, spolupráce s větším množstvím datových formátů a možnost spolupráce s dalšími programy jako Photoshop nebo Corel PHOTO-PAINT. Většina editorů stejně jako náš program podporuje například kreslení ve vrstvách, proporcionální změnu velikosti nebo seskupování elementů (dočasné i trvalé).

Téměř žádný z popsaných editorů ovšem již nepodporuje přiřazování použitého materiálu jednotlivým vrstvám, 2D řez, podporu práce s elementy o rozměrech od jednoho nanometru do několika centimetrů a možnost určovat přibližnou cenu a kvalitu návrhu. Důvodem je, že zmíněné funkce jsou přizpůsobeny technologii výroby, pro kterou je náš editor navržen, a proto bychom je těžko hledali například v CorelDraw. Stejně tak možnost nastavování stylů a práce s textem je zcela nepotřebná pro návrh čipů, a proto v našem editoru tyto nástroje nejsou implementovány.

2.6. FPSVG

Výstup editoru pro další zpracovaní navrženého čipu je popsán pomocí jazyku FPSVG, odvozeného od jazyku SVG (z anglického Scalable Vector Graphics - škálovatelná vektorová grafika). Jedná se o značkovací jazyk a formát souboru, který popisuje dvojrozměrnou vektorovou grafiku pomocí XML.

Samotný jazyk SVG definuje tři základní typy grafických objektů:

- vektorové tvary (obdélník, kružnice, elipsa, úsečka, lomená čára, mnohoúhelník a křivka)
- rastrové obrazy
- textové objekty

Tyto objekty mohou být různě seskupeny, formátovány pomocí atributů nebo stylů CSS a polohovány pomocí obecných prostorových transformací. SVG též podporuje ořezávání objektů, alpha masking, interaktivitu, filtrování obrazu (konvoluce, displacement mapping, atd…) a animaci.

Použitá úprava FPSVG má následující vlastnosti:

- popis vlastností čipu
- rozdělení do vrstev
- popis použitých materiálů
- veškeré rozměry jsou udány v mikrometrech
- odebraní nepotřebných parametrů jako je průhlednost a barva

Popis výstupního souboru:

Protože se jedná o XML formát, soubor začíná standardní XML hlavičkou,
<?xml version="1.0" encoding="utf-8"?>

Dále následuje inicializace čipu popisující jeho materiál a počet vrstev:
<chip numberOfLayers="intValue" substrateMaterial=”strValue” substrateThickness=”floatValue”>//začátek čipu

[popis vrstev čipu]

</chip> //konec čipu

Popis vrstev čipu se skládá z jejich výpisu, kde u každé vrstvy je uveden použitý materiál, její rozměry a výpis všech jejich elementů.
<layer width=“floatValue” height=“floatValue” material=”strMaterial” thickness=“floatValue” negative=”boolValue” layerNumber=“intValue”> // začátek vrstvy

[popis elementů]

</layer> //konec vrstvy

Popis elementů FPSG:

Čára
<line x1=”floatValue” y1=”floatValue” x2=”floatValue” y2=”floatValue” style=”strokewidth:floatValue;” />

(x1,y1): počáteční bod
(x2,y2): koncový bod

Obdélník:
<rectangle x=”floatValue” y=”floatValue” width=”floatValue” height=”floatValue” style=”fill:boolValue; stroke-width:floatValue; rotate:floatValue;” />

x, y: souřadnice levého horního rohu
width: šířka obdélníku
height: výška obdélníku
Střed rotace je definován jako rx = x + width/2, ry = y + height/2.

Elipsa:
<ellipse cx=”floatValue” cy=”floatValue” rx=”floatValue” ry=”floatValue” style=”fill:boolValue; stroke-width:floatValue; rotate:floatValue;” />

cx: střed elipsy x
cy: střed elipsy y
rx: velikost hlavní poloosy
ry: velikost vedlejší poloosy
Střed rotace je definován jako rx = cx, ry = cy.

Cesta:
<path style=”fill:boolValue; fillrule:evenodd; stroke-width:floatValue; rotate: floatValue;” d="strValue" />

kde střed rotace je určen jako střed ohraničujícího obdélníku (minX, minY, maxX, maxY) a parametr **d** je složen z následujících příkazů:

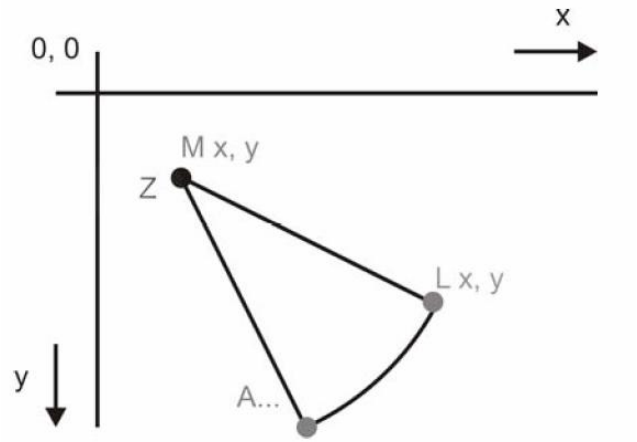
M - Přesun pera bez kreslení na nové souřadnice (X, Y) - vytvoří nový, oddělený segment (sub-path) v rámci celé cesty.

L - Obecná linka vedoucí na nové souřadnice (X, Y)+. V jednom atributu lze zadat celou posloupnost souřadnic.

A - (rx ry x-rotace větší-výseč orientace-výseče X Y)+

Vykreslí eliptickou výseč ze současného bodu do bodu x,y. Hodnoty rx,ry jsou poloměry elipsy a x-rotace její pootočení. Máme-li dány dva body, tak jako v našem případě, je možné jejich spojení provést čtyřmi způsoby. Parametr větší-výseč definuje, zdali se zvolí kratší (hodnota 0) nebo delší cesta (hodnota 1), zatímco "orientace-výseče" určuje směr kterým bude výseč vypouklá.

Z - Uzavře segment vektorové cesty vykreslením rovné čáry z aktuální polohy do počátečního bodu tohoto segmentu.



Obrázek 2.6.4: Výsledek postupné aplikace příkazů M, L, A a Z

Parametry elementů:

rotate(angle): úhel rotace elementu (ve stupních)
stroke-width: šířka čáry
fill: 0 : element není vyplněn
fill: 1 : element je vyplněn
(fill-rule: způsob vyplnění)

Parametr *fill* u popisovaného elementu určuje, zdali se jedná o vyplněný objekt. U jednoduchým objektů je snadno rozpoznatelné, která oblast leží uvnitř, ovšem u složitějších cest, kde se jednotlivé části mohou překrývat, je dále potřeba přidat pravidlo *fill-rule* popisující, jak v těchto případech rozhodovat.

V našem zápisu je použito pravidlo *evenodd.* Toto pravidlo určuje zdali je bod uvnitř popsané plochy podle počtu průsečíků hran s vyslaným paprskem v libovolném směru z daného bodu. Je-li počet průsečíků sudý, bod leží vně, je-li počet lichý bod leží uvnitř oblasti. Druhý možný popis *nonzero* dále u průsečíku rozlišuje, jedná-li se o zápis segmentů cesty po směru, nebo v protisměru hodinových ručiček.

Obrázek 2.6.2: Výsledek vyplnění různých cest metodou evenod., [W3TR]

ukázka výstupu:

```
<chip numberOfLayers="2" substrateMaterial=”glass” substrateThickness=”200.25”>
    <layer width="5000.1" height="5000.2" material="SU-8" thickness="20.75" negative=”true”
    layerNumber="1">
        <rectangle x="76" y="113" width="156" height="148" style="fill:False; stroke-width:0;
        rotate:0;" />
        <ellipse cx="297" cy="307" rx="98" ry="30" style="fill:False; stroke-width:0;
        rotate:0;" />
        <circle cx="297" cy="307" r="98" style="fill:False; stroke-width:0; rotate:0;" />
        <line x1="280" y1="91" x2="280" y2="91" style="stroke-width:0;" />
        <path style="fill:0; stroke-width:”3.5”;" d="M19 18 L40 45 L400 200 A3 4 10 5 8 1 0 Z"/>
    </layer>
    <layer width="5000.1" height="5000.2" material="SU-8" thickness="20.75" negative=”true”
    layerNumber="2">
    /* … */
    </layer>
</chip>
```

Jazyk SVG je patrně nejpoužívanější formát pro práci s vektorovou grafikou a je podporován většinou současných editorů. Alternativou k tomuto zápisu je například jazyk Post script.

3. Realizační část

Následující kapitola je věnována vlastnímu návrhu editoru, popisu implementace a funkcím vytvořeného softwaru.

Protože na celém vývoji aplikace se podílel i František Novák, který zpracovával práci s datovou oblasti, editaci elementů a renderování 3D modelu a Jan Kaiser, který kromě vedení projektu implementoval práci s měřítkem, přichytávání, 2D řez a mnoho dalších věcí, následující část nepopisuje všechny části programu. Mnou zpracovávaná část obsahuje především řešení GUI, práci s elementem cesta a přípravu dat pro 3D zobrazení.

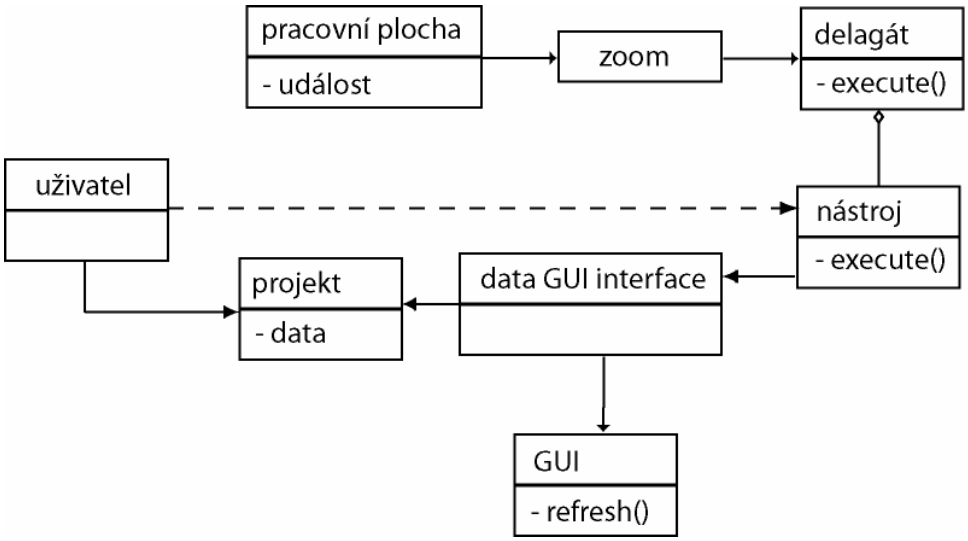
Celý editor byl napsán v objektově orientovaném jazyce C# a využívá technologie .NET. Pro samotné 3D zobrazení se využívá knihovny Microsoft Direct3D 9.0c. Program byl převážně vyvíjen pod systémem Windows XP a byl i úspěšně testován pod systémem Windows Vista.

Minimální doporučená hardwarová konfigurace je:

- Procesor Intel Pentium III nebo 4,
- 512 MB RAM ,
- Rozlišení 1 024x768.

3.1. Řídící vzor aplikace pomocí GUI

Na obrázku 3.1. je znázorněn řídící vzor, který byl použit pro ovládání našeho editoru pomocí GUI. Jednotlivé objekty diagramu zde nepředstavují konkrétní třídy, ale pouze tématické oblasti (např. pod *nástroj* spadá *newElement, EditElement, select,....*).



Obrázek 3.1:Řídící vzor GUI

Uživatel
- vybírá aktivní projekt a aktivuje pracovní nástroj. Aktivací nového nástroje je zajištěno odebraní všech současných reakcí při vyvolání delegáta. Aktivováním projektu dojde překreslení pracovní plochy a aktualizaci GUI.

Uživatel vytvoří nový čip a aktivuje nástroj draw rectangle, což zajistí, že na stisknutí tlačítka myši se již nebude editovat element.

Projekt
- jeden projekt reprezentuje jeden navržený čip. V aplikaci může být v jednom okamžiku otevřeno více projektů, ale pouze jeden může být aktivní. Každý projekt má svou vlastní pracovní plochu.

Po aktivování projektu se automaticky zobrazí jeho pracovní plocha.

Pracovní plocha
- je zdrojem událostí vyvolaných používáním myši. Odchytávané události jsou *mouseDown, mouseUp, mouseClick a mouseMove.* Po jejich vyvolání jsou následně přes třídu zoom přeposlány delegátu.

Když uživatel začne kresli obdélník, pracovní plocha pošle přes zoom delegátu zprávu o stisknutí tlačítka.

Zoom
- filtr zajišťující přepočet mezi obrazovkovými souřadnicemi a souřadnicemi čipu.

Převede souřadnici pixelu [30,30] na světové souřadnice [67mm, 93cm].

Delegát
- delegát odchytává události od všech pracovních ploch a vyvolává zaregistrované metody nástrojů. Jednoznačnost je zajištěna tím, že pouze aktivní projekt může událost vyvolat.

Při vyvolání delegáta, objekt zajistí vyvolání zaregistrovaných metod nástroje draw Rectangle.

Nástroj
- každý nástroj má své různé reakce na události pracovní plochy. Při jeho aktivaci si je zaregistruje. Nástroj nerozlišuje, od kterého projektu událost přišla a vždy pracuje s aktivním projektem. Po aktivaci nebo dokončení úkonu předá zprávu do *data GUI interface*

Při aktivaci nástroje draw rectangle si nástroj zaregistruje u delegáta metody pro obsloužení událostí pracovní plochy. Například reakce na delegáta vyvolaného stisknutím tlačítka je začátek kreslení nového obdélníku. V této metodě se dále nastaví, že při pohybu myši se má kreslit navrhovaný obdélník.

Data GUI interface
- zajišťuje promítnutí provedené akce do GUI a do dat aktivního projektu.

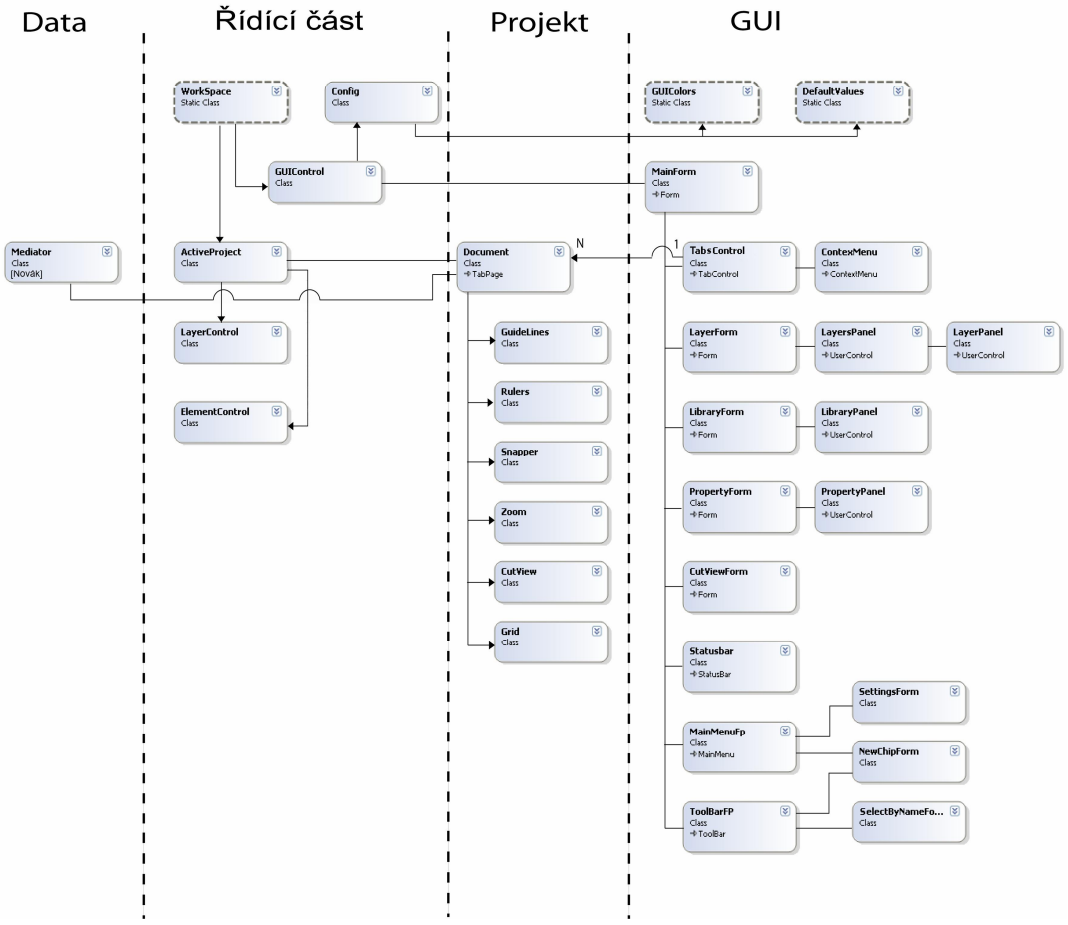
Vložení nakresleného obdélníku na aktivní vrstvu a vyvolání překreslení čipu.

GUI
- zajištění správného vykreslení komponent GUI.

Při aktivaci nástroje draw rectangle zajistí aktivaci příslušné ikony a deaktivaci ostatních ikon.

3.2. Návrh GUI

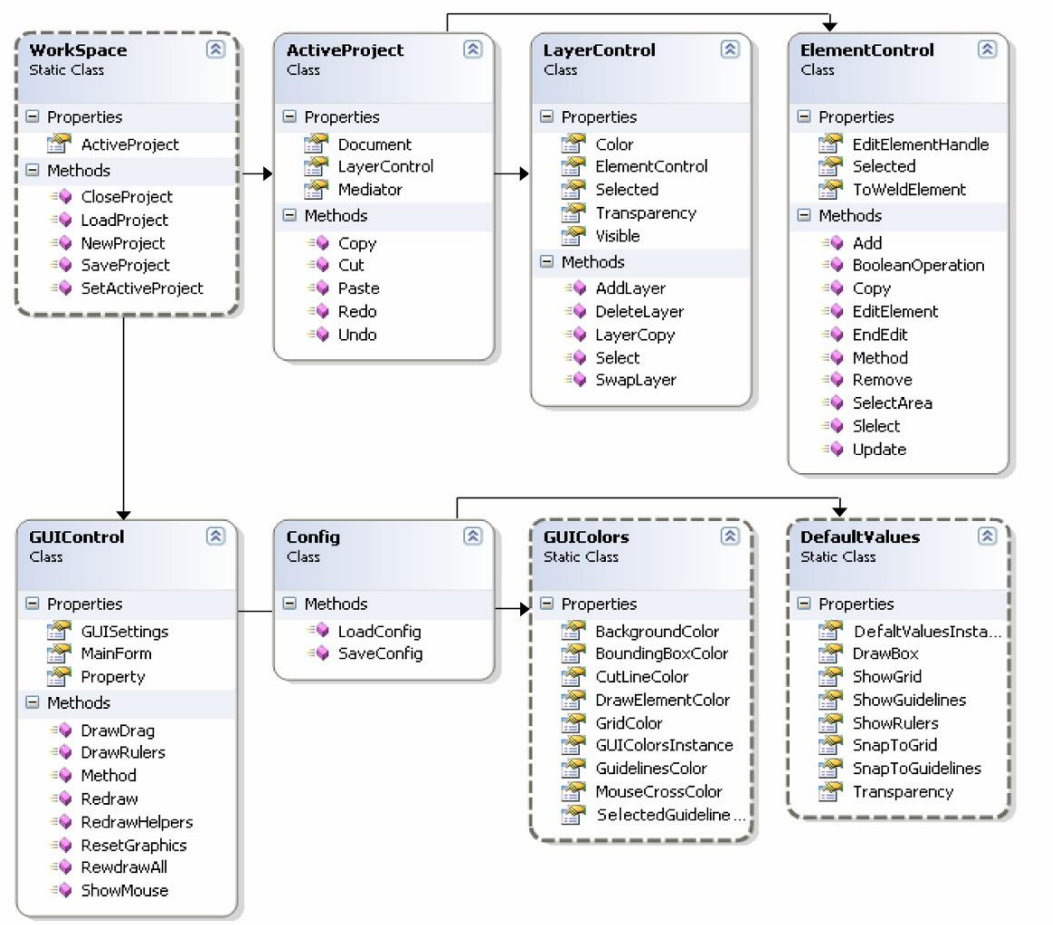
Třídy sloužící k práci s GUI jsou rozděleny do tří částí. První část zajišťuje komunikaci s datovou částí (řídící část), druhá reprezentuje třídy spojené s aktivním projektem a třetí obsahuje jednotlivé formuláře a ovládací panely.



Obrázek 3.2.1: UML diagram GUI

3.2.1. Popis důležitých tříd

Řídící část:



Obrázek 3.2.2: Řídící část

Workspace
Jedná se o třídu zapouzdřující celý program. Kontroluje manipulaci s celým editorem a jeho projekty. Řídí jejich zakládání, otevíraní, ukládání, zavírání, přepínání a úklid při ukončení práce.

ActiveProject
Zajišťuje nastavení komunikace mezi nástroji a třídami reprezentující rozhraní mezi daty aktivního projektu a GUI. Dále kontroluje navracení do stavu obdrženého z historie.

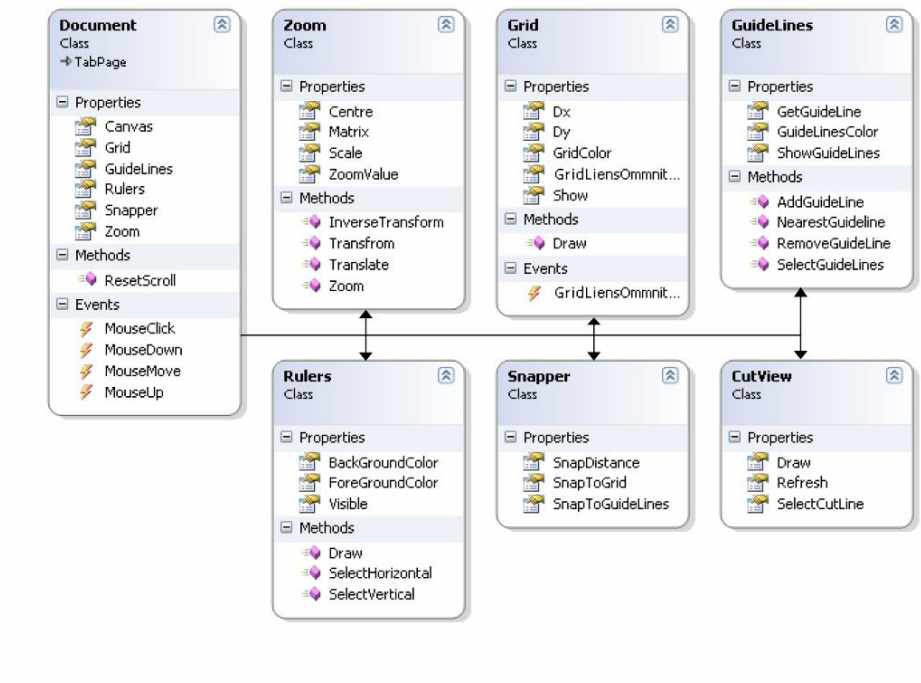
LayerControl
Poskytuje nástrojům rozhraní pro práci s datovými vrstvami projektu (jejich zakládání, rušení, editace, výběr,..) a zároveň zajišťuje, že GUI dostane pokyn k případné aktualizaci zobrazených vrstev. Třída dále uchovává referenci na aktivní vrstvu.

ElementControl
Poskytuje nástrojům rozhraní pro práci s datovými elementy projektu (jejich zakládání, rušení, editace, výběr,..) a zároveň zajišťuje, že GUI dostane pokyn k případné aktualizaci zobrazených elementů. Třída dále uchovává referenci na aktivní element.

GUIControl
Třída řídí chovaní GUI a uchovává informace o současném nastavení. Rovněž kontroluje vykreslování na pracovní plochu podle tohoto nastavení a spravuje potřebné bitmapy, z nichž se výsledný obraz skládá.

Config
Třída obstarává při spuštění aplikace načtení nastavení editoru. Při ukončení aplikace naopak zajišťuje, že hodnoty nastavené uživatelem nebudou ztraceny. Konfigurační soubor má každý uživatel uložen v dokumentech mezi daty patřící aplikaci. Soubor obsahuje například informace o barevném nastavení, počátečních hodnotách, nebo naposled použité cestě pro ukládání. Barevné nastavení aplikace je uloženo do třídy GUIColors a počáteční hodnoty projektů do třídy DefaultValues.

Třídy projektu:



Obrázek 3.2.3:.Třídy projektu

Dokument
Je grafická komponenta oddděná od třídy TabPage reprezentující jeden konkrétní projekt. Třída zajišťuje generování událostí pro nástroje (pohyb myši, zmačknutí tlačítka,…) a spravuje pomocné třídy spjaté s každým projektem.

Snapper
Řízení přichytávání editovaného elementu ke mřížce a vodícím čarám.

Zoom
Třída zajišťuje přepočet parametrů událostí dokumentu z obrazovkových souřadnic na souřadnice čipu, nebo obráceně. Rovněž obsahuje transformační matici určující, která oblast čipu bude vykreslena.
Grid
Obsluhuje vykreslování mřížky dle nastavených hodnot se zvolenou barvou.

GuideLines
Zajišťuje vykreslování vodících čar zvolenou barvou a jejich editaci.

Rulers
Vykresluje pravítka s hodnotami dle nastaveného měřítka do zadané bitmapy.

Cut View
Třída obsahuje nastavení pro 2D řez příslušným čipem.

Ovládací prvky GUI:

MainForm
Hlavní okno aplikace která vlastní formuláře LibraryForm, LayerForm, PropertyForm, CutViewForm a komponenty DokumentTabs, Statusbar, ToolBar a MainMenu. Řídí jejich zakládání a rušení.

LibraryPanel
Komponenta poskytující uživateli nástroje pro manipulaci s knihovnami objektů. Řídí jejich nahrávání a ukládání do systémového adresáře commomAplicationData, který společný všem uživatelům.

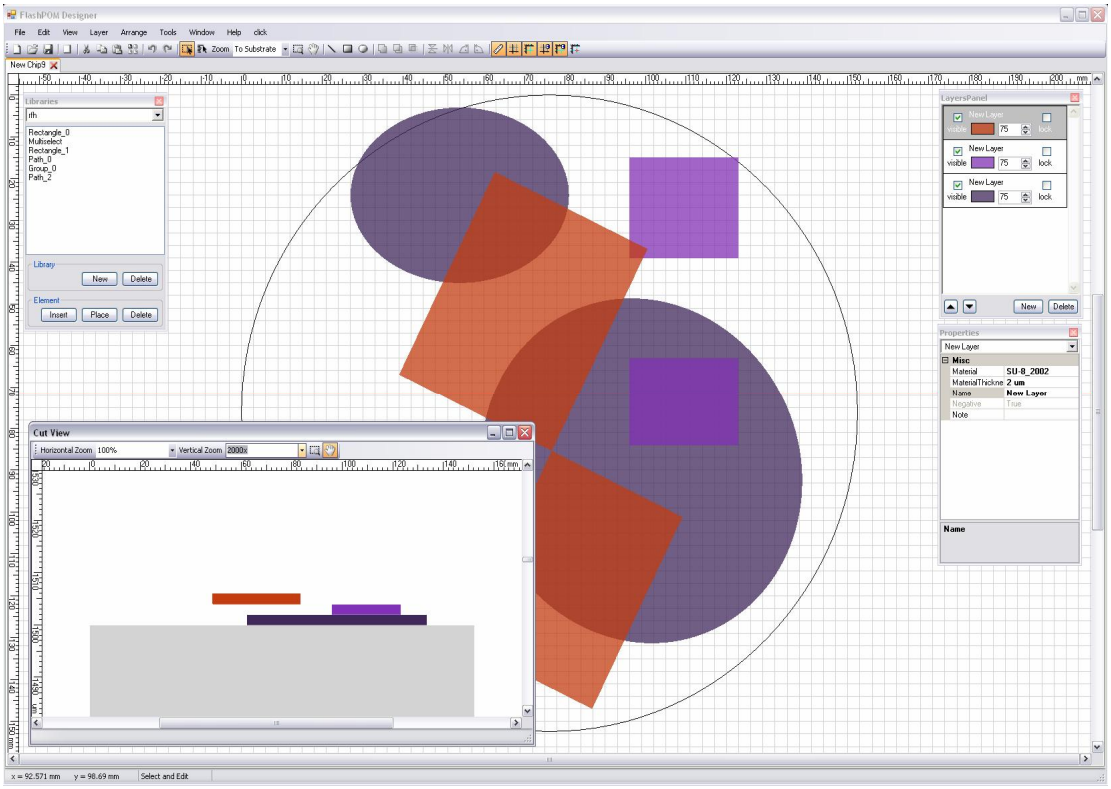
LayersPanel
Panel zobrazující vrstvy čipu. Rovněž poskytuje možnost přidání, nebo odebrání vrstvy a změnu její pozice v hierarchii čipu.

LayerPanel
Panel reprezentující jednu vrstvu čipu. Umožňuje její aktivaci, přesunu nebo změnu některého z parametrů (průhlednost, zamknutí, vykreslování a barva).

PropertyPanel
Komponenta zobrazuje informace o aktivním elementu a výpis všech objektů projektu. Třída zajišťuje, že před nastavením nové hodnoty, bude zkontrolována její platnost.

3.3. Komponenty GUI

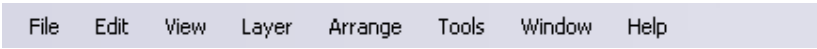
Rozložení typické pracovní plochy je zobrazeno na obrázku 3.3.1. Okno aplikace se skládá z menu, panelu nástrojů, okna knihovny, vrstev a vlastností, a panelu aktivního projektu (projekty lze přepínat pomocí jednotlivých záložek). U všech komponent lze nastavit, zdali se mají zobrazovat. V případě samostatných oken lze nastavit i jejich pozici a velikost (uchovává se i po restartovaní aplikace). Většinu nástrojů je možno ovládat i pomocí klávesových zkratek.



Obrázek 3.3.1: GUI aplikace

U GUI lze také nastavit základní barvy aplikace a výchozí hodnoty pro založení nového projektu.

3.3.1. Hlavní menu



Obrázek 3.3.2: Hlavní menu

File

Nabídka File obsahuje standardní příkazy k manipulaci se soubory (projekty), s nimi spojenými systémovými příkazy a příkaz pro ukončení aplikace. Soubory mohou být ukládány ve formátu .fpm (interní formát), nebo FPSVG.

New (Ctrl+N)
Příkazem se vytvoří nový prázdný čip, jehož parametry (materiál substrátu, tvar, velikost) uživatel zvolí v zobrazeném dialogovém okně. Při zakládání nového čipu je uživatel rovněž donucen k založení první vrstvy. Hodnota *zoom* je při zakládání nového čipu nastavena na *Fit to screen.* Parametry projektu spojené s editorem jsou nastaveny dle počátečních hodnot z nastavení.

Open (Ctrl+O)
Načte uložený návrh čipu ze souboru ve formátu .fpm. V souboru jsou uloženy i veškeré informace ohledně nastavení projektu. Okno o*pen dialog* se otevře na naposled použité cestě při manipulaci se soubory.

Save (Ctrl+S)
Uloží aktivní projekt do souboru ve formátu .fpm. Čip je uložen do souboru, z něhož byl načten (bez dotazu, zdali se má soubor přepsat), nebo jedná-li se o nový návrh, je vyvolána metoda Save as. Přednastavené jméno souboru je zvoleno dle jména ukládaného čipu. Společně s daty jsou uloženy i parametry editoru spojené s projektem (mřížka, vodící čáry, zoom, posun plochy).

Save As (Ctrl+Shift+S)
Save as umožňuje uživateli zvolit, kam se má ukládaný čip uložit. Čip je uložen ve formátu *.fpm.*

Close (Ctrl+X)
Uzavření aktivního projektu. Před uzavřením je uživatel dotázán, zdali chce projekt uložit. Je-li je otevřeno více projektů, po uzavření je zobrazen předchozí projekt.

Import from FPSVG
Načte čip z dat uložených v souboru formátu FPSVG. Název čipu je určen z názvu souboru. Ostatní parametry projektu jsou nastaveny dle počátečních hodnot pro zakládání projektu.

Export to FSVG
Exportování dat editovaného čipu do souboru ve formátu FPSVG.

Chip info
Zobrazí dialogové okno poskytující výpis základních informací o navrhovaném čipu.

Print current chip design. (Ctrl+P)
Vytiskne navržený čip dle zvolených parametrů.

Recenlty opened files
Poskytne uživateli seznam projektů, které byly naposled editovány.

Exit (Alt+F4)
Ukončení aplikace. Než je aplikace ukončena, uživatel dostane možnost každý projekt uložit.

Edit
Menu obsahuje nástroje k základní, editaci a manipulaci s vrstvami a elementy čipu.

Undo (Ctrl+Z)
Vezme zpět naposledy provedený příkaz uložený do historie. Počet prvků historie lze nastavit v menu Settings. Do historie se ukládá jakákoliv editace elementu, přidání a odebrání elementu nebo vrstvy a změna vlastností vrstvy nebo čipu. Každý projekt má vlastní historii.

Redo (ctrl+Y)
Vloží do historie zpět z ní odebraný příkaz funkcí Undo. Jestliže byl po zavolání Undo čip editován, historie Redo je vymazána.

Copy (Ctrl+C)
Zkopíruje aktivní element do schránky. Ve schránce může být uložen pouze jeden element a nelze do ní vložit vrstvu nebo celý čip.

Paste (Ctrl+V)
Zkopíruje element ze schránky na aktivní vrstvu (nemusí se shodovat s vrstvou, z níž byl element do schránky uložen). Souřadnice umístění odpovídají souřadnicím originálu.

Cut (Ctrl+X)
Zkopíruje aktivní element do schránky. Originál je z čipu odebrán.

Delete (Del)
Smaže vybraný element.

Snap to grid
Nastaví přichytávání editovaného elementu ke mřížce.

Snap to guidelines
Nastaví přichytávání editovaného elementu k vodícím čarám.

Move back
Posune aktivní element dolů po ose *z*. Pozice na ose *z* určuje, který element bude první vybrán při výběru jednoho elementu nástrojem *select and edit*.

Move front
Posune aktivní element nahoru po ose z.

View

Zoom in (Ctrl +)
Zvětší hodnotu měřítka na dvojnásobek. Střed transformace odpovídá středu pracovní plochy.

Zoom out (Ctrl -)
Zmenší hodnotu měřítka na polovinu. Střed transformace odpovídá středu pracovní plochy.

Fit to screen (Ctrl *)
Nastaví měřítko tak, aby se celý čip nejlépe vykreslil na pracovní ploše.

Fit To width
Nastaví měřítko tak, aby se čip na šířku nejlépe vykreslil na pracovní ploše.

Fit to height
Nastaví měřítko tak, aby se celý čip na délku nejlépe vykreslil na pracovní ploše.

Layer

New Layer
Zobrazí uživateli dialogové okno pro přidání nové vrstvy. Po nastavení základních parametrů bude vrstva vložena na vrchní pozici v hierarchii čipu. Barva přiřazená vrstvě určující, jak se budou vykreslovat všechny její elementy je volena náhodně.

Delete layer
Odebere aktivní vrstvu a všechny elementy, které se na ní nacházejí.

Arrange

Menu poskytuje nástroje pro seskupování jednotlivých elementů. Seskupovat lze pouze elementy v rámci jedné vrstvy. Elementy lze sloučit buď do skupin, ve kterých jsou původní elementy zachovány, nebo do cest. V případě cesty je vytvořen zcela nový element. Ten kvůli neproporcionální editaci už nelze zpět na původní elementy rozložit.

Group
Vytvoří skupinu z vybraných elementů. Všechny takto sloučené elementy se poté chovají jako jeden a veškeré transformace jsou stejnoměrně aplikovány na každý element skupiny. Skupina se může skládat i z dalších podskupin.

Ungroup
Rozdělí skupinu na elementy, z nichž byla složena.

Ungroup all
Rozdělí skupinu na elementy, z nichž byla složena. Pokud některý z těchto elementů je skupina, je i na ní aplikována tato metoda.

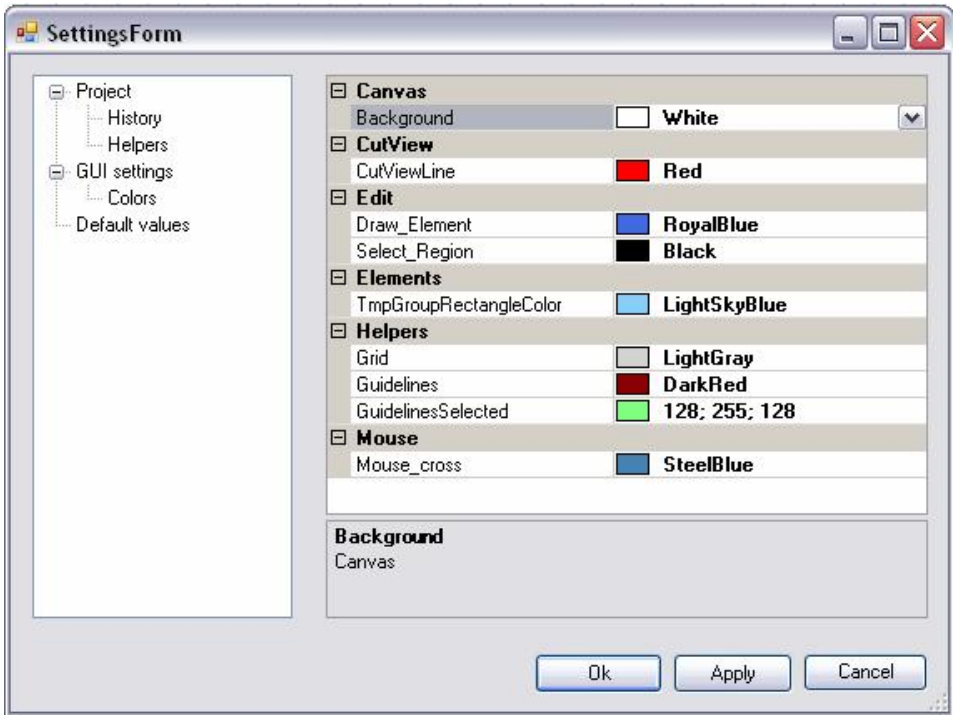
Weld (Ctrl + 6)
Vytvoření nového elementu typu cesta jako sloučení dvou různých elementů.

Trim (Ctrl + 7)
Vytvoření nového elementu typu cesta jako rozdíl dvou elementů (od původně vybraného elementu je odečten nově vybraný).

Intersect (Ctrl + 8)
Vytvoření nového elementu typu cesta jako průnik dvou elementů.

Tool

Settings
Zobrazí formulář s nastavením aplikace a aktivního projektu.



Obrázek 3.3.3: Panel nastavení aplikace

V settings lze nastavit:

- počáteční hodnoty nového čipu (parametry mřížky, nastavení editace,....),
- počet kroků historie,
- současné nastavení GUI (zobrazení formulářů, mřížky, vodících čar, ...),
- barvy aplikace (pozadí, barva mřížky, barva kříže, ...).

Check constrains
Nástroj provede kontrolu, zdali lze čip zkonstruovat a vypíše případné chyby v návrhu.

Estaminate chip cost
Podle počtu vrstev a velikosti povrchu pro opracování odhadne cenu výroby čipu.

Update Materials and Cost File
Program na internetu zjistí, zdali existují novější data určující cenovou funkci, nebo nové materiály a případně provede aktualizaci.

Update the FlashPOM Editor
Zkontroluje na internetu, zdali existuje novější verze editoru a případně provede aktualizaci.

Window
Obsahuje položky menu umožňující skrytí, nebo zobrazování jednotlivých formulářů a panelů.

Layers panel (Ctr+L)
Nastavuje zobrazení formuláře s vrstvami čipu.

Property panel (Ctrl+P)
Nastavuje zobrazení formuláře se zobrazenými vlastnostmi vybraného elementu.

Library panel (Ctrl+B)
Nastavuje zobrazení formuláře s knihovnami objektů.

Tools (Ctrl+T)
Nastavuje zobrazení panelu nástrojů.

Status bar (Ctrl+S)
Nastavuje zobrazení informační lišty.

Cut view (Ctrl + W)
Nastavuje zobrazení formuláře s 2D řezem navrhovaného čipu. Se zobrazením okna se na pracovní ploše rovněž vykreslí čára řezu, jenž může být posunována pomocí nástroje *select and edit.*

3D view
Aplikace vyexportuje 3D model čipu a zobrazí ho v přiloženém rendereru.

Default
Pokud kvůli snížení rozlišení došlo k přesunu některého z oken mimo obrazovku, příkaz umožňuje jejich navrácení do počáteční pozice.

Help

About
Zobrazení základních informací o editoru.

User refence
Zobrazí uživatelskou příručku.

3.3.2. Panel nástrojů

Samotný panel nástrojů je rozdělen na standardní, editační, kreslící a nastavující nástroje. Každá z těchto skupin může být pomocí kontextového menu skryta.


New project
Viz 3.3.1 File


Open project
Viz 3.3.1 File


Save project
Viz 3.3.1 File


New layer
Viz 3.3.1 Layer


Cut
Viz 3.3.1 Edit


Copy
Viz 3.3.1 Edit


paste
Viz 3.3.1 Edit

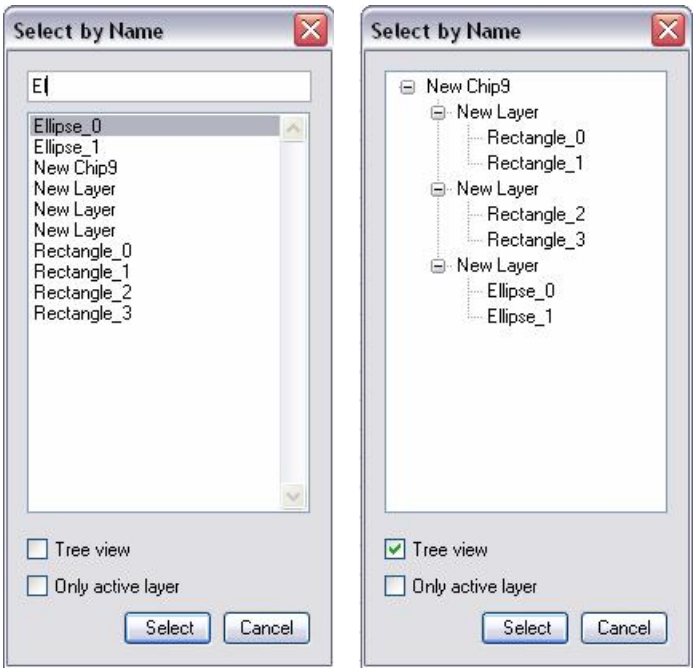

Undo
Viz 3.3.1 Edit


Redo
Viz 3.3.1 Edit


Distribute
Nástroj umožňuje rozkopírovat element po vrstvě. Výstupem je obdélníkový vzor *m x n* kopií, u nichž uživatel může nastavit rozestupy ve směru *x* a *y*. Rozestupy odpovídají vzdálenostem mezi ohraničujícími obdélníky elementů.


Select and edit
Nástroj umožňuje vybrat a editovat elementy pracovní plochy. Při pouhém stisknutí levého tlačítka myši je vybrán jeden element na aktivní vrstvě s největší *z* souřadnicí. Podržíme-li levé tlačítko, můžeme nakreslit obdélníkovou oblast, ze které budou vybrány všechny elementy, které do ní náleží, nebo ji alespoň protínají. Držíme-li zároveň klávesu Ctrl, je nový výběr přidán k současnému. Vybraný element je zobrazen v editačním módu.
Pokud se nepodaří vybrat žádný element, nástroj se pokusí vybrat vodící čáru. Pokud není vybrána ani ta, je vybrána aktivní vrstva.


Select by name
Zobrazí menu, které uživateli poskytne jmenný seznam všech elementů navrhovaného čipu. Elementy mohou být zobrazeny jako abecední seznam, u kterého je možno vyhledat první element začínající daným řetězcem, nebo jako hierarchická struktura, u niž jsou elementy seřazeny dle stáří.



Obrázek 3.3.4: Select by name

Po zaškrtnutí položky *Only selected layer* se z menu odfiltrují všechny elementy ležící mimo aktivní vrstvu.

Zoom

Položka umožňuje nastavení měřítka zobrazení. Minimální hodnota měřítka je 10%, maximální hodnota měřítka je 1000000%. Při 100% jeden cm na obrazovce odpovídá 1cm v reálném světě. Střed transformace odpovídá středu pracovní plochy.
Jelikož měřítko se pohybuje ve velkém rozsahu, je možno zadat hodnotu jako *číslox*, což odpovídá hodnotě *číslo%**100 (5x = 500%).
Menu také obsahuje tři předdefinované hodnoty *fit to screen, width* a *height*, poskytující uživateli optimální pohled na celý čip.

Zoom area
Nástroj umožňuje zvolit střed transformace při změně měřítka. Pokud uživatel pouze klikne na pracovní plochu, je měřítko zdvojnásobeno (s klávesou Ctrl dvakrát zmenšeno) a po změně měřítka je bod kam uživatel klikl ve středu pracovní plochy. Pokud uživatel nakreslí oblast, měřítko je automaticky nastaveno tak, aby vybraná oblast byla co nejlépe zobrazena.

Move Working Area
Umožňuje uživateli posunovat pracovní plochu.

Draw line
Kreslení čáry. Počáteční bod je zakreslen při stisknutí tlačítka myši, koncový bod při uvolnění tlačítka myši. Při kreslení všech elementů lze využít přichytávání.

Draw rectangle
Kreslení obdélníku. Je-li je stisknuta klávesa Ctrl, kreslí se čtverec.

Draw ellipse
Kreslení elipsy. Je-li je stisknuta klávesa Ctrl, kreslí se kružnice.

Weld
Viz. 3.3.1 Arrange

Trim
Viz. 3.3.1 Arrange

Intersect
Viz. 3.3.1 Arrange

Flip Horizontally
Zrcadlení elementu podle osy X procházející středem elementu.

Flip Vertically
Zrcadlení elementu podle osy Y procházející středem elementu.

Rotate 90° Clockwise
Rotace elementu o 90 stupňů doleva podle středu elementu.

Rotate 90° Counter-clockwise
Rotace elementu o 90 stupňů doprava podle středu elementu.

Show/Hide Rulers
Viz 3.3.1 View

Show/Hide Grid
Viz 3.3.1 View

Show/Hide Guidelines
Viz 3.3.1 View

Snap to Grid
Viz 3.3.1 Edit

Snap to Guidelines
Viz 3.3.1 Edit

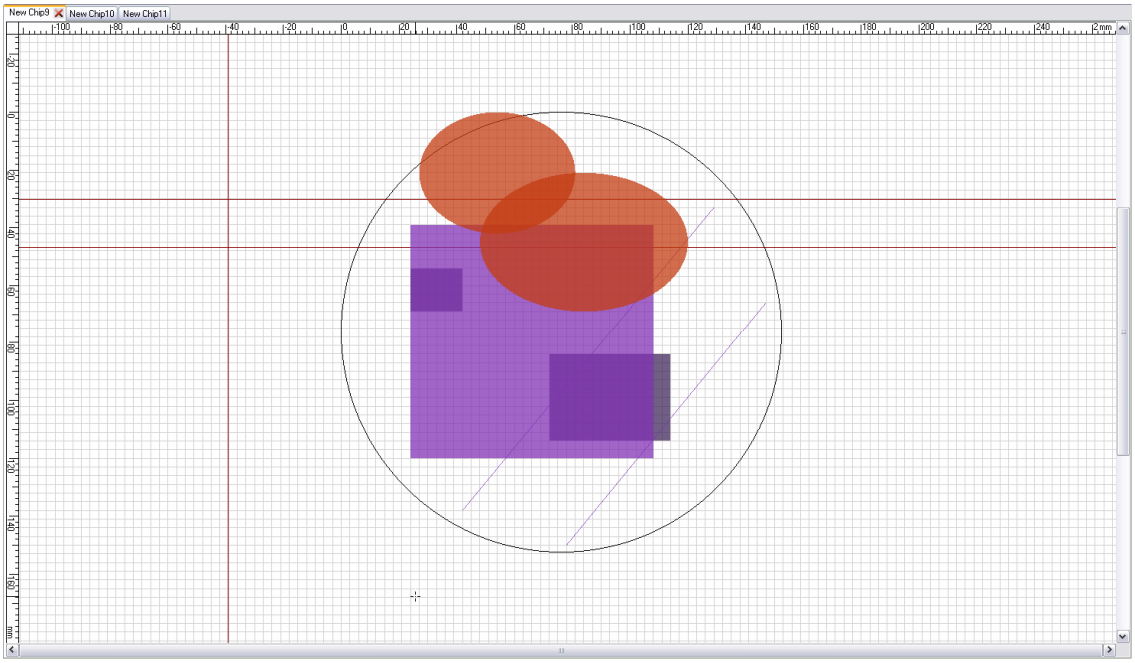
Show/Hide Cross Cursor
Aktivuje vykreslování kříže místo klasického kurzoru. Barva kříže je nastavitelná v položce hlavního menu *settings*.

3.3.3. TabsControl

Komponenta obsahuje dokumenty jednotlivých projektů a umožňuje přepínání mezi nimi. Součástí je i kontextové menu obsahující příkaz k editaci, seskupování, využívání schránky a posunu elementu po ose z.

3.3.4. Pracovní plocha

Pracovní plocha projektu je editační oblast navrhovaného čipu. Každý otevřený projekt má vlastní plochu a její nastavení. Na ploše jsou vykresleny elementy všech viditelných vrstev barvou jí příslušející, mřížka, vodící čáry, pravítka a editační kopie elementu.

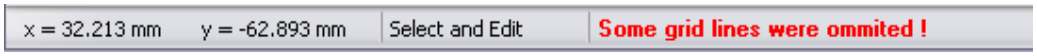


Obrázek. 3.3.5: Pracovní plocha

Při práci na ploše lze vždy editovat pouze elementy na aktivní vrstvě projektu. Pokud návrh čipu neobsahuje žádnou vrstvu, je pozadí vyplněno šedou barvou a nelze na něj nic vykreslit.

3.3.5. Status bar

Stavový řádek obsahující výpis současnou pozice kurzoru, název aktivního nástroje a varovné hlášení v případě, že vykreslená mřížka kvůli velkému měřítku neobsahuje všechny čáry.



Obrázek. 3.3.6: Status bar

3.3.6. Formulář knihovny

Komponenta umožňuje manipulaci s knihovnami obsahující uživatelem vytvořené elementy, nebo skupiny elementů. Komponenta umožňuje přístup ke všem knihovnám v adresáři .commonAplicationData./objectLib/, kde je každá knihovna uložena jako seznam elementů.

Tlačítkem *New* může uživatel vytvořit novou knihovnu (název se nesmí shodovat s již existující knihovnou), tlačítkem *Delete* je vybraná knihovna smazána.

Element je do knihovny vložen pomocí tlačítka *Insert*, tlačítkem *Place* je element vložen na aktivní vrstvu na stejné souřadnice, ze kterých byl do knihovny vložen. *Delete* vybraný element z knihovny vymaže.

Další možnost, jak umístit element z knihovny na aktivní vrstvu je metoda *drag&drop*. Uživatel tak může požadovaný element přímo přetáhnout z knihovny na požadovanou pozici (pozice kurzoru určuje střed řídícího obdélníku)

Obrázek. 3.3.7: Formulář knihovny objektů

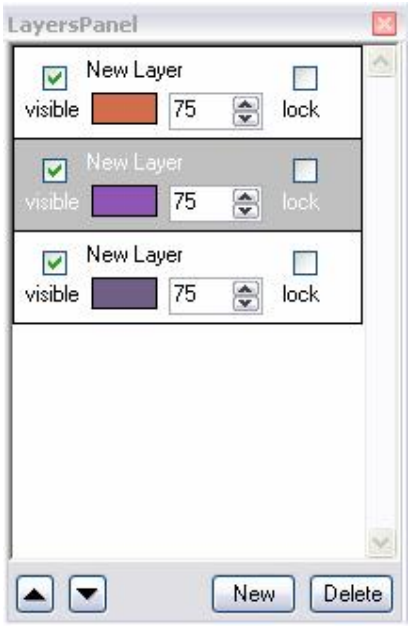
Soubory jednotlivých knihoven obsahují serializovaný seznam elementů a pro uživatele je nečitelný, aby se zamezilo případné snaze editovat prvky knihovny přímo v souboru.

3.3.7. Formulář vrstev

Komponenta je určena pro navigaci mezi vrstvami čipu (substrát se jako vrstva nepočítá). Nová vrstva je vždy přidávána nahoru. Panel vybrané vrstvy je zobrazen systémovou barvou aktivní kontrolky. Na panelu každé vrstvy je možno nastavit barvu, jakou jsou její elementy vykreslovány, jejich průhlednost, zdali se má vrstva vykreslovat a je-li vrstva zamknutá.

Novou vrstva lze přidat kliknutím na tlačítko *New*, které vyvolá dialog pro vytvoření nové vrstvy.

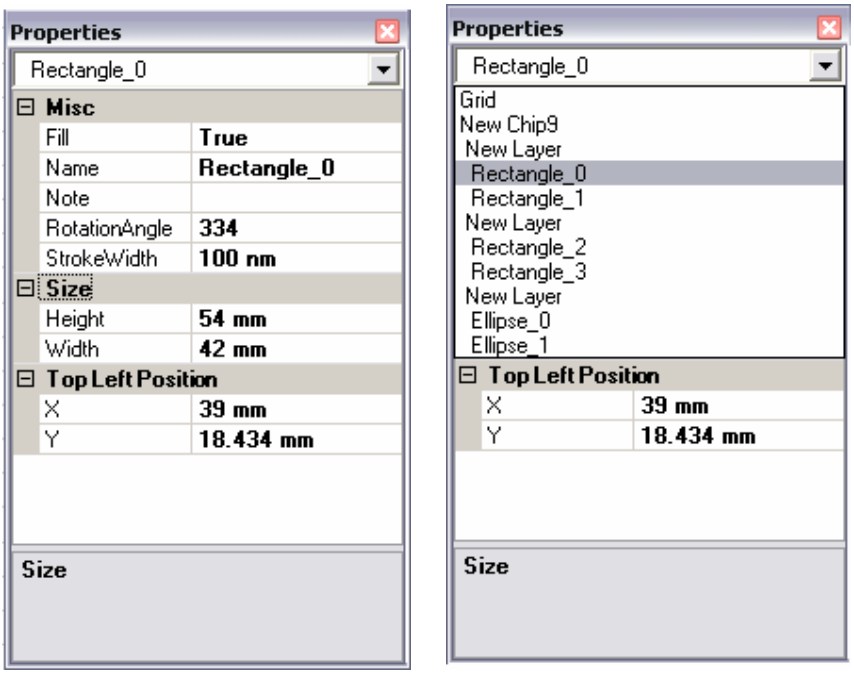
Tlačítkem *Delete* je aktivní vrstva odebrána (jako aktivní vrstva je poté nastavena vrstva následující). Pomocí tlačítek se šipkami nahoru/dolů lze měnit pozici vrstvy v rámci hierarchie čipu. Pozice vrstvy lze také měnit pomocí *drag&drop*.



Obrázek. 3.3.8: Formulář vrstev

3.3.8. Formulář vlastností

Formulář automaticky zobrazuje naposledy vytvořený, nebo editovaný prvek projektu. Editujeme-li některý element, jeho nové vlastnosti jsou na formuláři zobrazeny ihned po dokončení jednotlivých kroků. Pokud editujeme některý prvek projektu přes tuto komponentu, je nejprve provedena kontrola, je-li zadaný parametr platný (tj. jestli nebyl zadán nulový rozměr) a až poté dojde k vykreslení zadané změny.



Obrázek. 3.3.9: Formulář vlastností elementu

Zobrazitelné prvky a jejich nastavitelné vlastnosti:

Grid

- dx	rozestup mřížky ve směru X (nejsou-li uvedeny jednotky, pracuje se s mikromertry)
- dy	rozestup mřížky ve směru Y (nejsou-li uvedeny jednotky, pracuje se s mikromertry)

Guidelines

- lock	*true/false* hodnota umožňující uzamknutí vodící čary
- name	název vodící čáry
- orientation	orientace vodící čáry (horizontal nebo vertical)
- position	pozice vodící čáry

Chip

- material	materiál substrátu čipu
- material thicknes	tloušťka substrátu
- name	název čipu
- note	uživatelem napsaná poznámka k čipu
- size	velikost substrátu

layer

- color	barva, jakou se vykreslují všechny elementy na dané vrstvě
- lock	*true/false* hodnota umožňující uzamknutí vrstvy
- material	materiál vrstvy

- material thicknes	tloušťka vrstvy
- name	název vrstvy
- negativ	hodnota určující, jedná-li se o pozitivní či negativní vrstvu
- note	uživatelem napsaná poznámka k vrstvě
- transparency	procentuální hodnota určující průhlednost elementů na vrstvě při vykreslování
- visible	*true/false* hodnota, má-li se vrstva vykreslovat

element

- Fill	hodnota *true/false* určující, zda se má element vykreslovat jako vyplněný
- Name	název elementu
- Note	uživatelem zapsaná poznámka k elementu (nelze u dočasného výběru)
- Rotation angle	úhel ve stupních určující natočení elementu
- Stroke width	šířka čáry (uplatňuje se pouze, je-li Fill nastaveno na *false*)
- Top left posiotion	pozice levého horního rohu řídícího obdélníku

element line

- begin XY	počáteční bod úsečky
- end XY	koncový bod úsečky

element rectagle

- width	šířka obdélníku
- height	výška obdélníku

element ellipse

- major	velikost hlavní osy elipsy
- minor	velikost vedlejší osy elipsy

multi select

- width	šířka transformovaného hraničního obdélníku
- height	výška transformovaného hraničního obdélníku

group

- width	šířka transformovaného hraničního obdélníku
- height	výška transformovaného hraničního obdélníku

path

- width	šířka transformovaného hraničního obdélníku
- height	výška transformovaného hraničního obdélníku

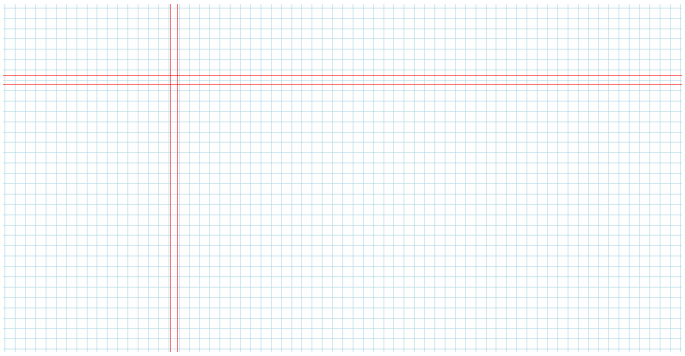
3.4. Vykreslovaní

Jelikož během editování je potřeba vykreslovat poměrně velké množství na sobě nezávislých objektů, jejich přímočaré vykreslování by při každé změně vedlo k přílišnému zatížení procesoru (vykreslování je operace zabírající celkově největší množství času). Proto bylo nutné vytvořit systém, který by zajistil, že bude překreslováno pouze to, co je třeba. To vedlo k rozložení vykreslovaného čipu a prvků GUI do několika nezávislých bitmap, kde každá je překreslována pouze v případě, je-li změněn její specifický obsah. Výsledný obraz na pracovní ploše tak vzniká až spojením bitmapy reprezentující čip, pozadí a bitmapy, kde jsou zakresleny aktivní prvky. Tento postup sice přináší větší paměťovou náročnost, ovšem výsledné urychlení má mnohem větší vliv na plynulý běh aplikace.

3.4.1. Vykreslení pozadí

Obsah podkladu pracovní plochy určuje barva pozadí, pravítka, vodící čary, mřížka a čára 2D řezu. Pro tyto účely jsou vytvořeny dvě pomocné bitmapy, kde první obsahuje vykreslená pravítka a barvu pozadí a druhá mřížku, vodící čary a 2D řez. Toto rozdělení zajišťuje, že pokud bude změněn některý z obsažených prvků pro získaní bitmapy pozadí není potřeba překreslovat vše (tj. ušetří se především překreslování pravítek při editaci vodících čar a 2D řezu). Výsledná bitmapa pozadí je určena jejich spojením.

Obrázek 3.4.1: Bitmapa s pravítky+ šedé pozadí



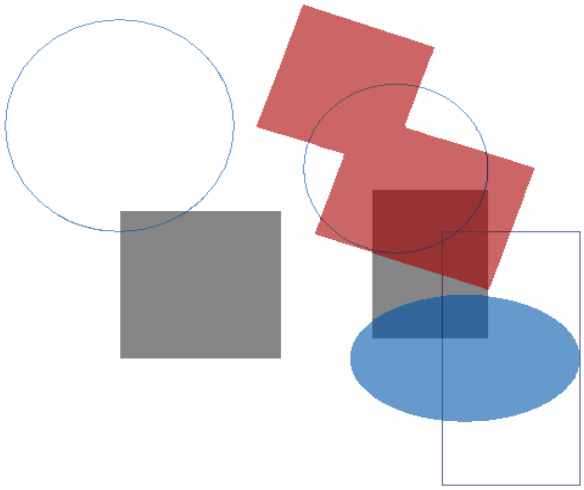
Obrázek 3.4.2: Poloprůhledná bitmapa s mřížkou a vodícími čarami

Protože rozlišení mřížky může být často vyžadováno mnohem jemnější, než je rozlišení obrazovky, od určité úrovně jsou vykreslovány pouze některé čáry. V tomto případě je uživatel upozorněn na stavovém řádku varovným výpisem.

3.4.2. Vykreslení čipu

Stejně jako pro pozadí pracovní plochy, i pro navržený čip je vytvořena vlastní bitmapa, do které je vykresleno spojení poloprůhledných bitmap jednotlivých vrstev. Pokud došlo ke změně na některé z vrstev, je bitmapa této jedné vrstvy před spojením s ostatními překreslena. Podrobnější popis optimalizace vykreslování vrstev a elementů vrstvy lze nalézt v [Novák] .

Jelikož samotná změna pozadí je relativně málo častá, je bitmapa pozadí použita jako podklad pro vykreslení čipu. To má za následek, že je-li překresleno pozadí, musí dojít i ke překreslení čipu (pouze spojení vrstev), ale je ušetřeno jejich spojovaní při překreslování jako reakce na pohyb myši.

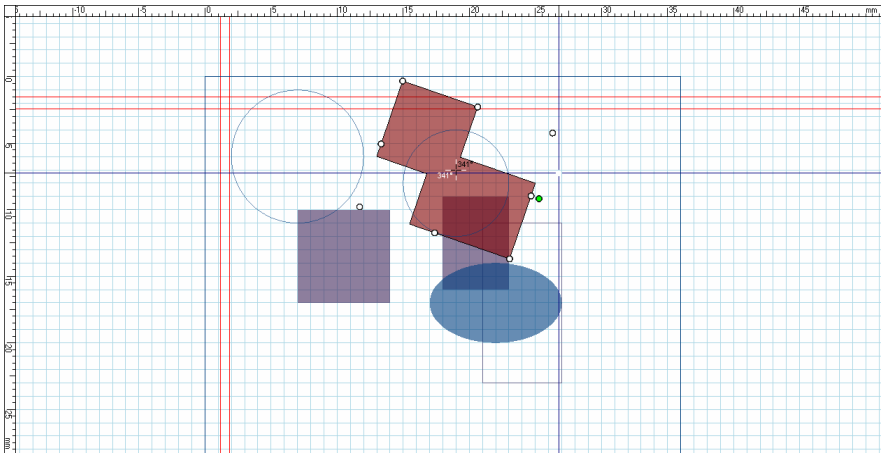


Obrázek 3.4.3: Bitmapa čipu

Překreslení vrstev čipu je vyvoláno pouze v případě dokončení editace některého z elementů, nebo samotné vrstvy a v případě změny některého z parametrů třídy *zoom* (změna měřítka, posun pracovní plochy).

3.4.3. Vykreslování editovaného elementu

Jelikož editovaný element je překreslován i při pohybu myší (kvůli zobrazení provedené editace elementu), při výběru velkého množství elementů by jejich neustálé překreslování způsobovalo citelné zpomalení. Proto třída *editElement* obsahuje vlastní bitmapu, kam editované prvky vykreslí pouze jednou a dokud nedojde k jejich změně používá tuto bitmapu.



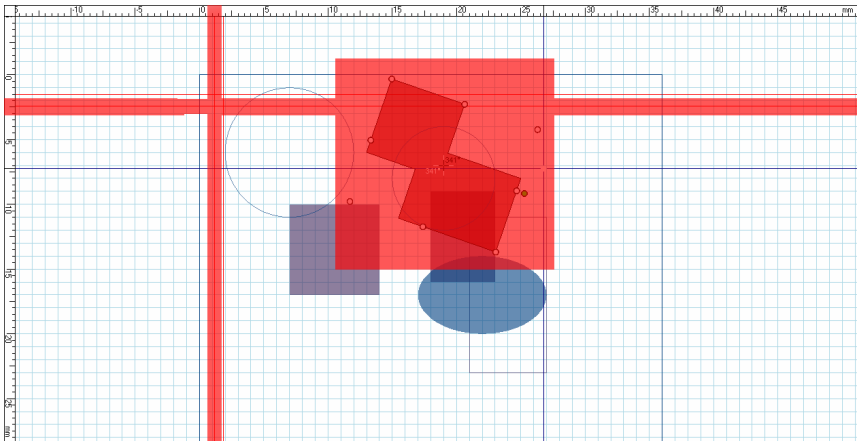
Obrázek 3.4.4: Vykreslení editovaného elementu přidá na plochu jeho obrys editační prvky

3.4.4. Kreslení při pohybu myší po plátně

Kvůli frekvenci volání, je překreslování jako reakce na pohyb myší po pracovní ploše, jednou z nejnáročnějších metod aplikace. Naštěstí samotný pohyb myši nepodněcuje překreslení pozadí nebo čipu, a proto můžeme využít již vytvořené bitmapy. Při pohybu myší se překresluje:

- Ukazatele pozice na pravítkách,
- editovaný element,
- kříž kurzoru,
- editovaná vodící čára, nebo čára řezu.

Pro tyto potřeby je vytvořena bitmapa aktivních prvků, kam si každý z těchto požadavků zakreslí oblast potřebnou k překreslení. Do těchto oblastí jsou poté překopírovány oblastí ze spojené bitmapy pozadí a čipu a bitmapy s editovaným elementem. Stejná oblast je poté vykreslena na pracovní ploše, což zajistí, že zbylá oblast zůstane nezměněna od minulého vykreslení.

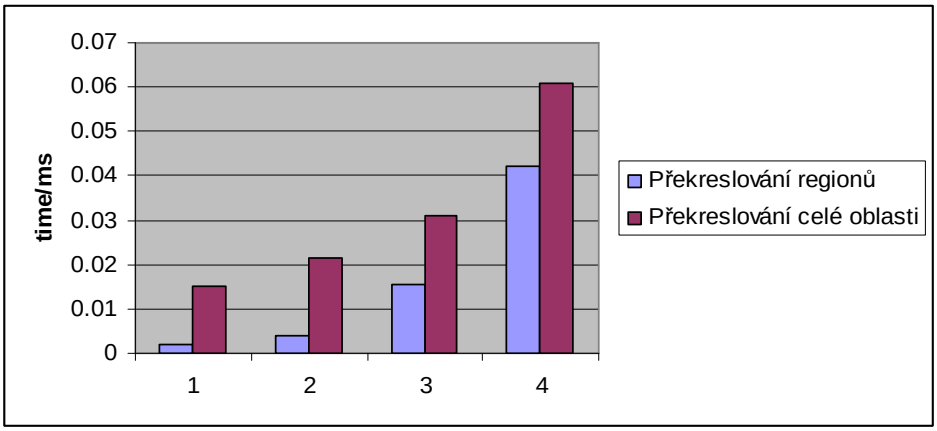


Obrázek 3.4.5: Výsledek překreslení se zvýrazněnou oblastí překreslení, jenž byla určena editovaným elementem, ukazateli pravítek a křížem myši.

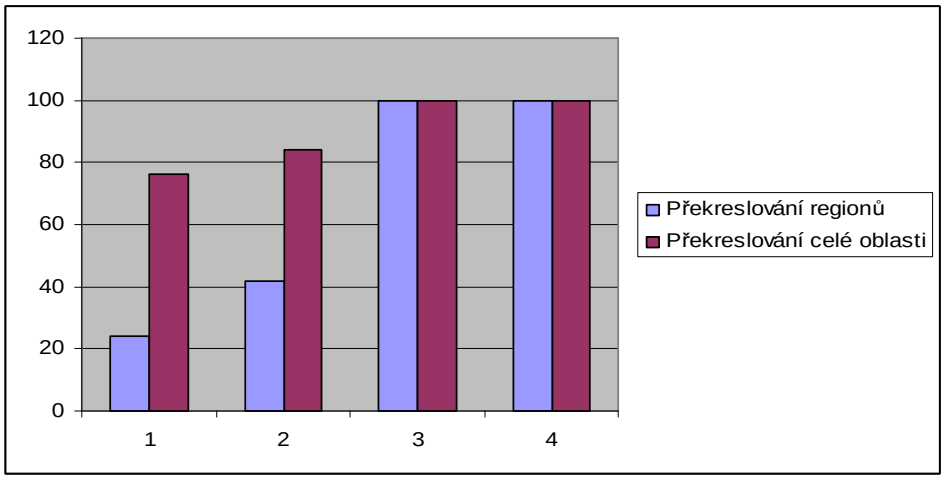
Další snížení náročnosti je omezení překreslovaní na maximálních 30 snímků za vteřinu.

3.4.5. Urychlení

Na grafech 3.18 a 3.19 je možno vidět výsledek snížení časové a výpočetní náročnosti překreslování pracovní plochy při pohybu myší. Na obou grafech je znázorněno porovnání metod, kdy se překresluje pouze editovaná, nebo celá plocha.



Graf 3.4.6: Časové urychlení



Graf 3.4.7: Snížení procentuální zátěže procesoru

Popis sloupců grafu:

1. Překresluje se pouze kříž a ukazatele na pravítkách.
2. Překresluje se pouze kříž, ukazatele na pravítkách a editovaný element.
3. Ukázka zvýšení náročnosti překreslování, pokud by pravítka nebyla kopírována z bitmapy pozadí, ale neustále překreslována.
4. Ukázka zvýšení náročnosti překreslování, pokud by mřížka a vodící čary nebyla kopírována z bitmapy pozadí, ale neustále překreslována.

Testování probíhalo na počítači s procesorem Intel 1.6GHz, 512MB při rozlišení 1280x1024 se zapnutým vykreslování kříže kurzoru, mřížky vodících čar a pravítek.

3.5. Element cesta (path)

Pro popis netriviálních oblastí editor obsahuje element *cesta*, který vzniká permanentním spojením jednotlivých elementů. Jejím použitím je možno popisovat složité oblasti bez nutnosti překrývaní několika elementů ve vrstvě (vynutilo by si vícenásobný průchod laseru) a tím snížit náklady na výrobu (viz. [Novák 3.5]). Další výhodou je, že se již nejedená o základní objekt, a proto není použita proporcionální editace (z obdélníku se může stát kosodélník). Nevýhodou cest je zvýšená výpočetní a vykreslovací náročnost a z důvodu neproporcionální editace nelze rozbít cestu na původní primitiva.

3.5.1. Sestrojení cesty

Pro sestrojení cesty jako výsledek logických operací součet, součin a rozdíl nad dvěma elementy jsem vyzkoušel dva algoritmy. První, Weilert Athertonův algoritmus založený na sledování orientovaných hran protínajících se elementů poskytuje menší algoritmickou složitost a jeho výstupem je opět orientovaný n-úhelník. Tento postup ovšem nebyl použit, jelikož vede k velkému počtu singulárních případů, kde selhává (např. u sledování souběžných, opačně orientovaných hran).

Druhý, použitý postup, je rozšířením zametacího algoritmu, kde singulární případy nepředstavují problém, ale bohužel neuchovává orientaci n-úhelníku a ani nijak neodlišuje díry v popsané oblasti.

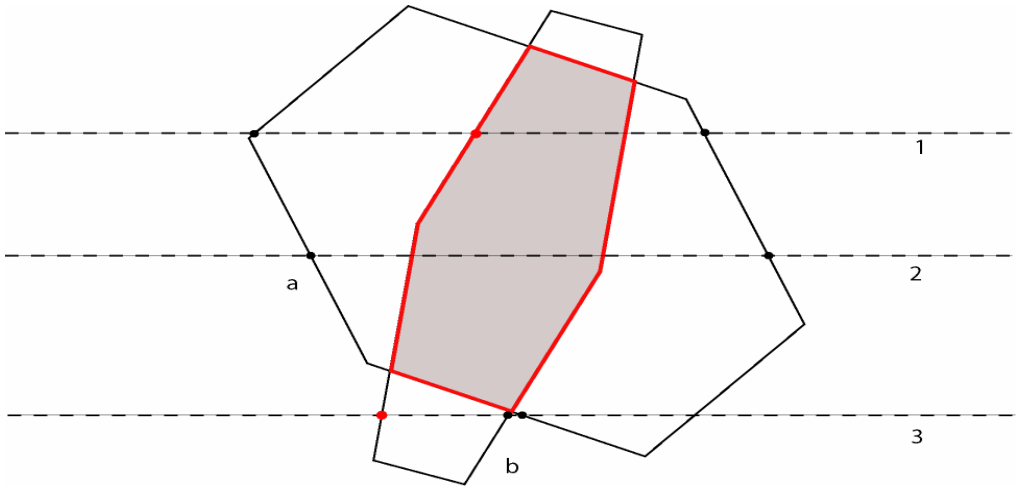
V prvním kroku algoritmu je potřeba nejprve napočítat průsečíky vstupních elementů a vytvořit z nich cesty. Tyto cesty jsou složeny z původních (neprotnutých) segmentů a segmentů nově vytvořených rozděleným protnutých hran.

Segmenty mohou být:

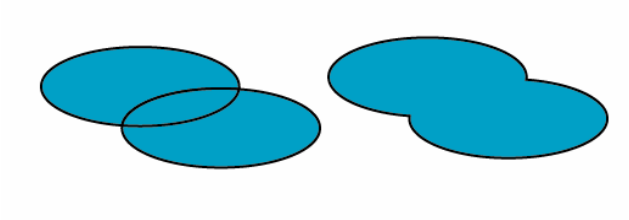
samostatné čary:	Vznikají z elementu typu line a jsou určeny pouze dvěma body.
hrany polygonů:	Hrana určena počátečním a koncovým bodem. Každá hrana má odkaz na následující a předchozí hranu, informaci je-li jeden z bodů průsečík a případně obsahuje odkaz na hranu druhého elementu, s níž se protnula.
eliptická úseč:	Vzhledem k obtížné manipulaci s eliptickými úsečemi dochází zde k aproximován posloupností hran, který tento segment zapouzdřuje. Při exportu dochází k převedení zpět na elipsu.

V dalším kroku je každá hrana obou elementů testována, zdali je uvnitř nebo vně druhého elementu. Podle výsledku a použité operace je rozhodnuto, bude-li segment použit ve výsledné cestě. Při testování je v případě segmentu, jehož počáteční bod nevznikl jako průsečík o pozici rozhodnuto podle sousedního segmentu. V opačném případě je při testování segmentu určen průsečík s horizontální přímkou procházející středem hrany (nebo vertikální, je-li úhel větší než 45 stupňů). Pozice tohoto průsečíku je poté porovnána s průsečíky této přímky a druhého elementu.

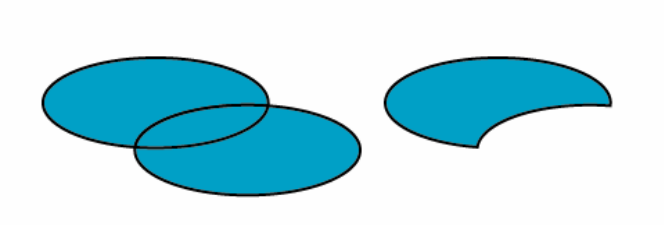
Je-li počet předcházejících průsečíků sudý, hrana leží vně, je-li lichý, hrana uvnitř. Pokud nelze rozhodnout, je třeba dopočítat průsečíky i s prvním elementem a rozhodnout dle použité operace.



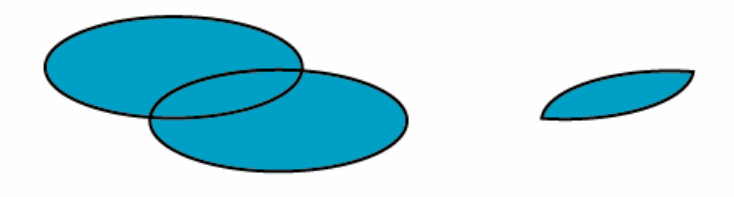
Obrázek 3.5.1: Testovaní hran cesty b při určování průniku. Při testu přímkou 1, je hrana přijata, jelikož průsečíku s hranou cesty b předchází lichý počet průsečíků. U přímky 2 není potřeba testovat. U přímky 3 je hrana zamítnuta.



Obrázek 3.5.2: Sjednocení (weld)



Obrázek 3.5.3: Rozdíl (trim)



Obrázek 3.5.4: Průnik (intersect)

V závěrečném kroku algoritmu je ze schválených segmentů sestavena výsledná cesta. V tomto kroku se využívá referencí hran na hrany druhé cesty, s nimiž se protnuly. Bohužel narozdíl od algoritmu WA, je směr orientace náhodný.

Jelikož tato třída obsahuje metody pro určování průsečíků netriviálních objektů, je využita i pro kontrolu možné sestrojitelnosti čipu [Novák 3.6].

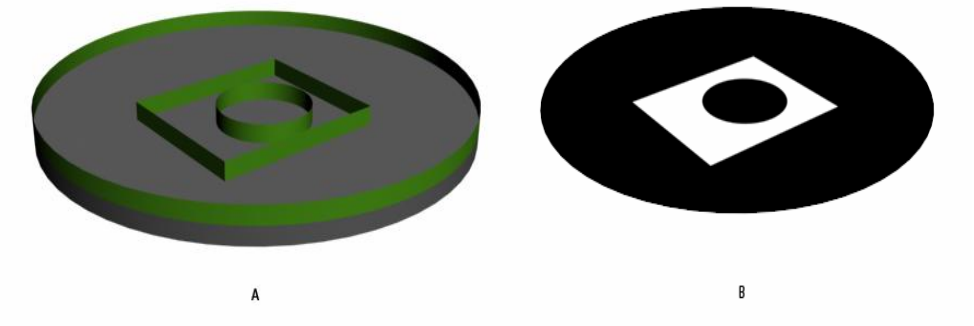
3.6. 3D Zobrazení

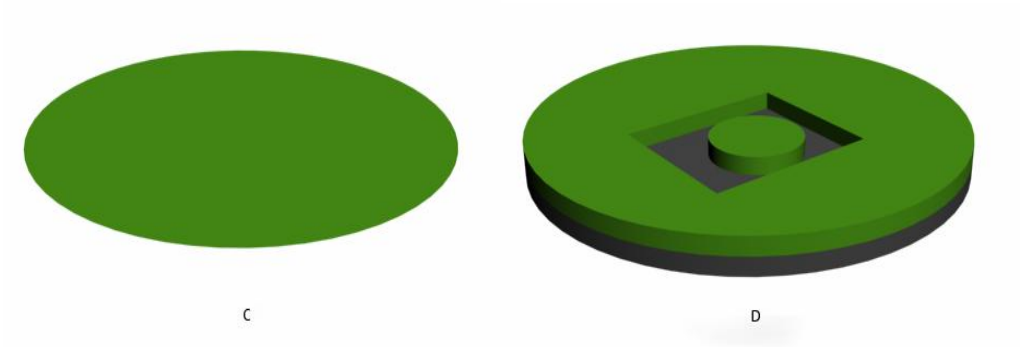
Součástí našeho editoru je i nástroj, který umožňuje uživateli zobrazit navržený čip ve 3D. Při spuštění editor vygeneruje soubor modelů, které jsou následně automaticky zobrazeny v dodaném rendereru. Tento renderer využívá knihovnu DirectX a je dodáván jako samostatná aplikace. Důvodem tohoto oddělení je zajištění, že běh samotného editoru nebude ovlivněn případnou nekompatibilitou grafické karty, nebo špatnou verzí DirectX. Popis rendereru lze nalézt v [Novák 3.7].

3.6.1. Návrh zobrazení

Protože přímá triangularizace vrstvy pro 3D zobrazení by byla příliš složitá a vedla k vytvoření velkého množství trojúhelníků, byl navržen postup, který umožňuje vytvoření 3D modelu čipu triangularizací jednotlivých elementů nezávisle na svém okolí. Tento postup je dále nezávislý na tom, jedná-li se o negativní, či pozitivní vrstvu a odstraňuje problém s děrami v cestách.

Základní myšlenkou je využití stencil bufferu (SB), který nám umožňuje definovat oblast, kde je možno kreslit. Při renderování každé vrstvy nejprve vykreslíme vytažené hrany objektu. Následně do SB vykreslíme oblasti, jež budou zasažené laserem a oblasti děr v cestách. Zapisovaná hodnota se bude lišit podle druhu použitého materiálu. V dalším kroku vykreslíme spojitý povrch vrstvy, který díky použitému SB vytvoří obraz vrstvy odpovídající našemu návrhu.





Obrázek 3.6.1: Ukázka postupného vykreslování 3D modelu. A: Geometrie vytažených hran a substrátu. B: Model pro určení hodnot ve SB. C: Vrchní část vrstvy. D: Výsledný model

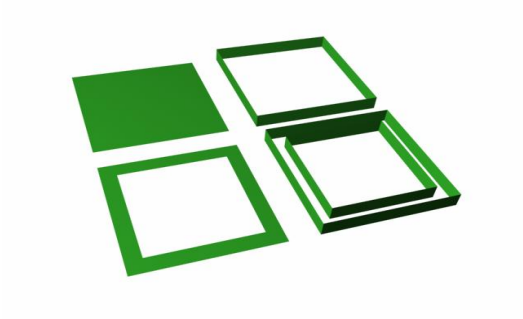
3.6.2. Převod elementů do 3D

Převod elementů do 3D se skládá ze dvou kroků. V prvním kroku jsou pro všechny vrstvy a její elementy vytvořeny modely stěn o výšce odpovídající tloušťce vrstvy (obrázek 3.6.2). V druhém kroku je provedena triangularizace oblastí, kterou jednotlivé elementy reprezentují.

Vytažení hran

U vytahování hran se rozlišuje, jedná-li se o vyplněný, či nevyplněný objekt. V prvním případě je potřeba pouze vytvořit dva trojúhelníky pro každou hranu a její kopii se zvýšenou souřadnicí z a nasměrovat jejich normálu dle materiálu (do středu objektu u pozitivní vrstvy, od středu u negativní).

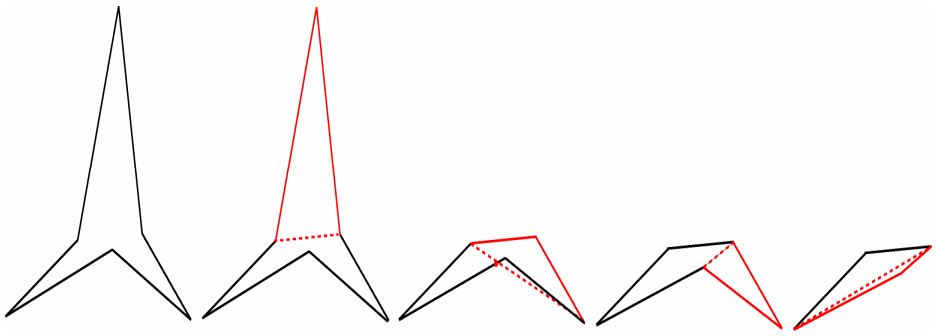
Složitější situace nastává v druhém případě, kdy je potřeba vytvořit vnitřní a vnější obrys podle použité šířky čáry. To je provedeno posunem vrcholu po ose úhlu sousedících hran. Tento současný postup bohužel zatím umožňuje vznik falešných hran v případě průniku vnitřního a vnějšího obrysu. Možným řešením tohoto problému je vytvoření elementu se zmenšeným a zvětšeným obrysem již ze základního primitiva, z něhož je cesta sestrojena, ovšem toto řešení by vedlo k omezení editovatelnosti cesty (možnost změny šířky čáry, vyplnění) kvůli 3D zobrazení, jehož účelem je pouze vizuální prezentace.



Obrázek 3.6.2: Výsledek po vytažení hran vyplněného a nevyplněného obdélníku

Triangularizace

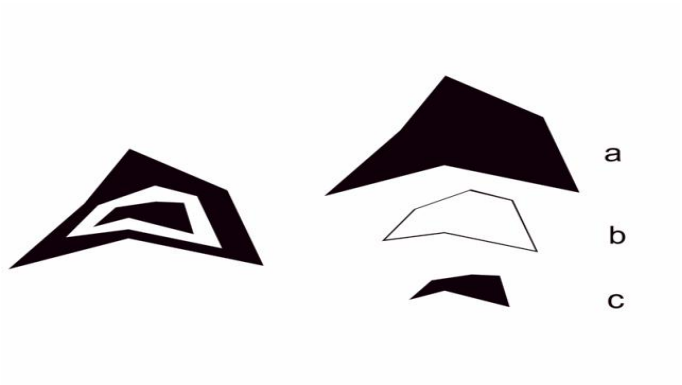
Samotná triangularizace využívá algoritmus „ořezávaní uší“, kde se postupně na objektu vyhledávají sousední hrany, které svírají konvexní úhel. Je-li takováto dvojice nalezena, přidá se hrana, která spojuje jejich vzdálenější vrcholy. Pokud tuto novou hranu nekříží jiná hrana n-úhelníku, je uložen nový trojúhelník, které tyto tři hrany dohromady tvoří. Původní dvě hrany jsou poté z n-úhelníku odebrány a hledá se další dvojice, dokud objekt obsahuje alespoň 4 vrcholy. Algoritmus využívá Meistersova teorému 2 uší: každý polygon s $n \geq 4$ vrcholy, má nejméně 2 nepřekrývající se uši



Obrázek 3.6.3: Ukázka postupné triangularizace třícípé hvězdy.

Na obrázku 3.6.3 je vidět, že výstupem mohou být i trojúhelníky, kde jednotlivé strany svírají extrémní úhly. Tyto případy není potřeba odchytávat, jelikož všechny trojúhelníky budou ležet v rovině a budou mít totožnou normálu.

Tento algoritmus nebude fungovat, bude-li použit na element cesta, který v sobě obsahuje díry, nebo je složený z více samostatných částí. V takovémto případě je nejprve nutné jednotlivé části cesty separovat a případné díry zapsat do SB s opačnou hodnotou, než pevné plochy.



Obrázek 3.6.4: Do SB bude místo původního objektu postupně zapsán segment a s hodnotou 1, segment b s hodnotou 0 jako poslední segment c opět s hodnotou 1.(u negativní vrstvy)

3.6.3. Výstup pro zobrazeni

Výstupem editoru pro 3D renderer jsou tři sady trojúhelníkových sítí ve formátu .X (formát popisující objekt mesh pro DirectX3D). První sada obsahuje popis povrchů jednotlivých vrstev určující na SB oblast vykreslování. Druhá sada obsahuje popis stěn jednotlivých elementů a vrstev a třetí pouze horní část vrstvy.

Každý soubor kromě standardní hlavičky definující formát souboru obsahuje:

výpis použitých materiálů:.

```
 Material PDX01_-_Default {                  //název materiálu
  float4;;                                   //difuzní složka
  float;                                     //index odlesku
  float4;;                                   //spekulární složka
  float4;;                                   //ambieltní složka
  TextureFilename {
   “String“                                  //použitá textura - nepovinné
  }
}
```

a popis hierarchie objektů a jejich geometrie:

```
Frame layer01 {                              //začátek první vrstvy – Frame je jeden blok

FrameTransformMatrix { float 4x4 }           //transformační matice vrstvy

layer mesh Data {}                           //popis geometrie vrstvy

   Frame element01 {                         //začatek popisu elementu

   FrameTransformMatrix { float 4x4 }        //transformační matice elementu

      Element mesh Data {}                   //data lementu
      }

   Frame element02 {                         // záčátek popisu druhéhé elementu

   FrameTransformMatrix { float 4x4 }        // transformační matice elementu

      Element mesh Data {}                   //popis geometrie druhéhé elemntu
      }
   }
}                                            //konec frame layer01
```

kde mesh data obsahuje:

```
mesh{

      int;                                   //počet vrcholů;
      float;float;float;,                    //souřednice vrcholů
      float;float;float;,
      …
      float;float;float;;
```

```
        int;                                    //počet trojúhelníků
        3;int;int;int;,                         //indexi vrcholů v troúhelníku
        3;int;int;int;,                         //číslo 3 značí, že používáme trojúheůlníky
        …
        3;int;int;tin;;



MeshNormals  {                                  //normály
        int;                                    //počet normál
        float;float;float;,                     //vektor normály
        float;float;float;,
        …
        float;float;float;;
        int;                                    //počet trojúhelníků
        3;int;int;int;,                         //indexi normál pro jednotlivé vrcholy
        3;int;int;int;,
        …
        3;int;int;int;;
        }

 MeshMaterialList  {                            //materiály
          int;                                  //počet materiálů
          int;                                  //počet trojúhelníků
          int,                                  //index materiálu
          …;
          { PDX01_-_Default }                   //název nadefinované textury/materiálu
        }

 MeshTextureCoords  {                           //texturovací souřadnice
        int;                                    //počet vrcholů
        float; float;,                          //texturovací souřadnice vrcholů
        float; float;,
        …
        float; float;;
        }

}                                               //konec nesh data
```

3.7. Možná rozšíření

Současná verze editoru poskytuje velký prostor pro možná rozšíření. Mezi ty patří například možnost přidávat a editovat jednotlivé elementy přes příkazovou řádku a tím umožnit rychlejší manipulaci s přesně definovanými objekty.

Druhé možné rozšíření je přidání základních primitiv, jako je n-úhelník, eliptická výseč a úseč, nebo lomená čára. Všechny tyto elementy lze sice nyní vytvořit pomocí cesty nebo pomocí předdefinovaných elementů v knihovně objektů, ovšem tento postup je pro uživatele zbytečně komplikovaný.

Další možná rozšíření mohou být například dokovatelnost formulářů, složitější vzory pro kopírování elementů, nebo vytváření cest ze skupin objektů.

4. Závěr

Diplomová práce popisuje vlastnosti námi navrženého editoru, který by měl být schopen poskytnout uživateli dostatečný komfort a funkcionalitu pro návrh čipů založených na PoM technologií.

V současné době je editor plně funkční a splňuje všechny zadané požadavky specifické pro náš projekt, jako je možnost návrhu čipu po jednotlivých vrstvách, 2D řez, 3D zobrazení, výstup pro další zpracování ve formátu založeném na jazyce SVG a snadná ovladatelnost. Editor rovněž poskytuje mnohé z funkcí, na které jsme zvyklí z klasických vektorových editorů, jako je například uchovávání historie, nastavitelnost GUI, pomocná mřížka a snadná práce s existujícími projekty.

Aby uživatel mohl náš program používat, není vyžadována širší znalost profesionálních vektorových editorů, které současný trh nabízí, nebo jakékoliv znalosti v oblasti IT. Během celého vývoje byla snaha se zákazníkem probírat veškerou zásadní funkcionalitu a ovládací prvky, pro co nejlepší splnění zadaných požadavků.

Samotné zadání mé práce považuji za velice zajímavé a mělo pro mě velký přínos v možnosti vyzkoušet si podílení se na větším softwarovém projektu. Kromě samotné implementace jednotlivých funkcí jsem měl i možnost si řádně vyzkoušet týmovou práci, obtížnost komunikace se zákazníkem a náročnost udržování takto rozsáhlého programu.

Alfa verze programu byla odevzdána 20.3. 2007. Po předvedení prototypu byly sepsány a opraveny funkční nedostatky. Odevzdání finální práce je plánováno na 5.6.2007.

Literatura

[Claverol05] E. Claverol-Tinturé, J. Cabestany, X. Rosell: Multisite recording of extracellular potentials produced by microchannel-confined neurons invitro, J. Neural Eng. 2 (2005) L1–L7

[Claverol04] E Claverol-Tintur´e, M Ghirardi, Fiumara, X Rosell and J Cabestany: Multielectrode arrays with elastomeric microstructured overlays for extracellular recordings from patterned neurons , TBME-00477-2005

[Berg] de Berg, Mark, van Kreveld, Marc, Overmars, Mark, Schwarzkopf, Otfried: Computational Geometry, Algorithms and Applications, Springer Verlag, 1.vydání, 1997

[Novák] Novák František: Diplomová práce; 2D editor pro návrh topologie čipu - projekt FlashPoM – 2

[Krüger03] Krüger Mike, mike@icsharpcode.net: C♯ Coding Style Guide, Version 0.3

[Rourke] O' Rourke, Joseph: Computational Geometry in C, Cambridge University Press, 2.vydání, 2000

[Mateo] http://mateoproject.org/public/index.php?option=com_content&task=view&id=58&Itemid=117

[Rera] http://www.rera.cz/index.php?documentID=36

[W3TR] http://www.w3.org/TR/SVG/

Příloha A

Tutoriál

Následující tutoriál v několika jednoduchých krocích ukazuje, jak snadno a rychle vytvořit návrh neuročipu podobný návrhu z obrázku 2.2.3. Dále bude ukázáno použití specializovaných nástrojů, jako je kontrola návrhu, odhad ceny a 2D řez.

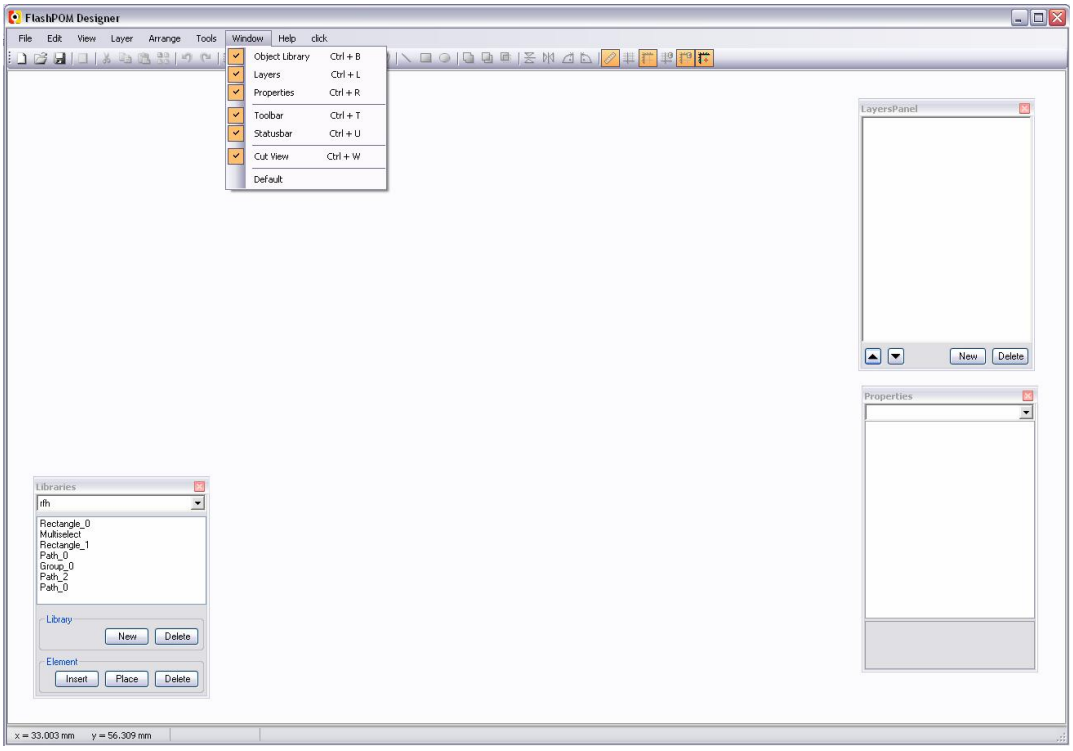
1. Instalace editoru

Samotný editor nepotřebuje instalaci. Díky použité technologii .Net stačí pouze spustit soubor FlashPoM.exe. Pro samotný běh programu je ovšem nutné mít nainstalovaný Microsoft Framework 2.0 a pro 3D zobrazení Microsoft DirectX 9.0c (není nutné pro běh samotného editoru).

Framework 2.0 i DirectX 9.0c jsou zdarma ke stažení na stránkách firmy Microsoft, nebo je lze nalézt na přiloženém CD.

2. Založení nového projektu

Po spuštění programu se zobrazí hlavní okno editoru bez založeného projektu. Pokud program spouštíme poprvé a nemáme zobrazena žádná pomocná okna (vlastnosti, vrstvy, knihovna, 2D řez), můžeme jejich zobrazení zapnout v menu View.



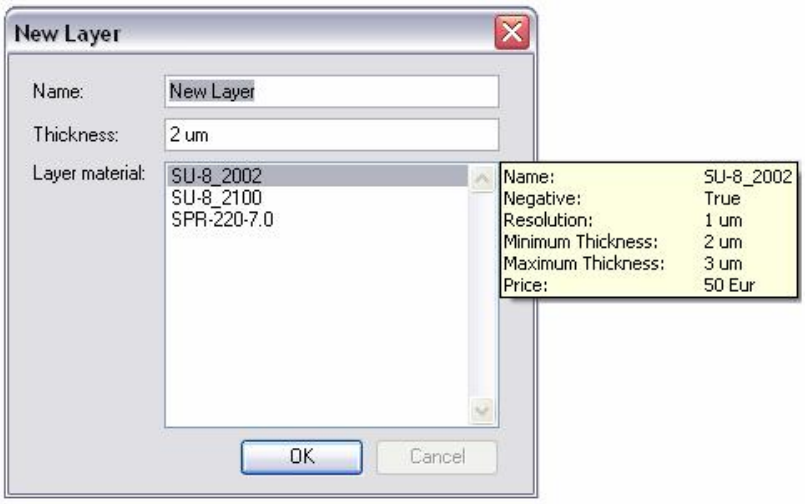
Obrázek A..1: Editor po spuštění aplikace.

Pokud kvůli snížení rozlišení monitoru došlo k přesunu některého z oken mimo plochu, příkazem *Default* v menu *View* jej můžeme vrátit do počáteční pozice.

Nový projekt založíme příkazem *New* v menu *File*. Při založení se editor zeptá, jaký materiál chcete použít jako substrát.

Obrázek A..2: Založení čipu.

Po zvolení substrátu je dále nutné zvolit materiál první vrstvy. Další vrstvu je možno založit pomocí příkazu New layer v menu Layer.



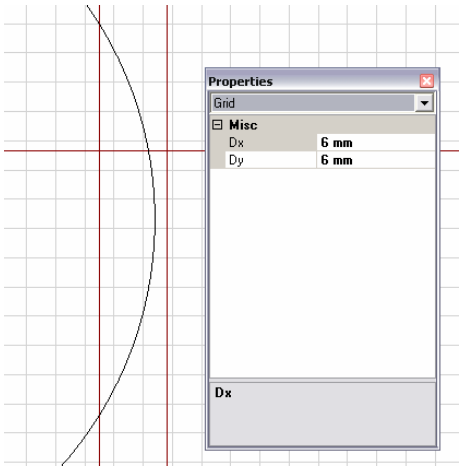
Obrázek A.3:Založení vrstvy.

Pro získaní návrhu podobného obrázku 2.2.3 si založíme vrstvu s positivním materiálem SPR-220-7.00 – tj. budeme kreslit oblasti určené k odstranění.

3. Kreslení základních primitiv

Než začneme kreslit, nejprve si zvolíme oblast, kam chcete kreslit. K tomu slouží nástroj Zoom area. Oblast vybereme nakreslením výběrového obdélníku a editor sám nastaví měřítko tak, aby co nejlépe zobrazovalo vybranou oblast.

V dalším kroku si nastavíme rozestup mřížky, tak aby nejlépe odpovídal velikosti kreslených elementů. Nastavení pro mřížku lze nalézt v panelu vlastností pod položkou *Grid*. Do projektu si rovněž můžete přidat vodící čary jejich vytažením z pravítek umístěných kolem pracovní plochy.



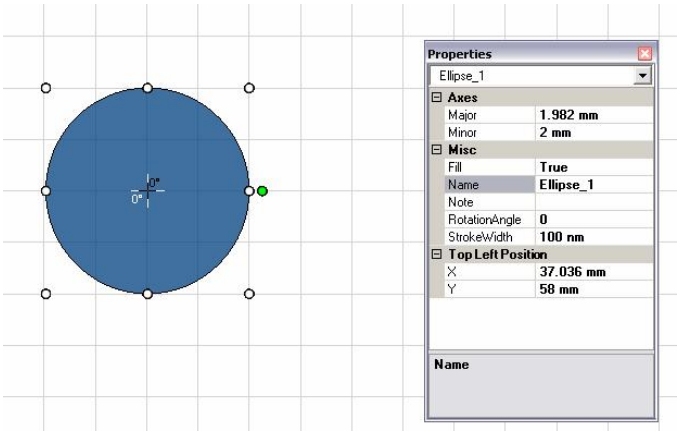
Obrázek A.4: Nastavení mřížky.

Před samotným kreslením je potřeba ještě vybrat element, který chceme kreslit. V panelu nástrojů si můžete vybrat z čáry, obdélníku a elipsy. Složitější objekty se vytváří jejich sloučením. Pro náš návrh zatím postačí nakreslení jedné elipsy. Ta bude reprezentovat otvor pro umístění neuronu.

Je-li při kreslení, zapnutý nastroj *snap to grid/guidelines*, bude se kreslený element automaticky přichytávat k mřížce/vodícím čarám. Jestliže podržíme klávesu Ctrl, bude se místo obdélníku/elipsy kreslit čtverec/kružnice.

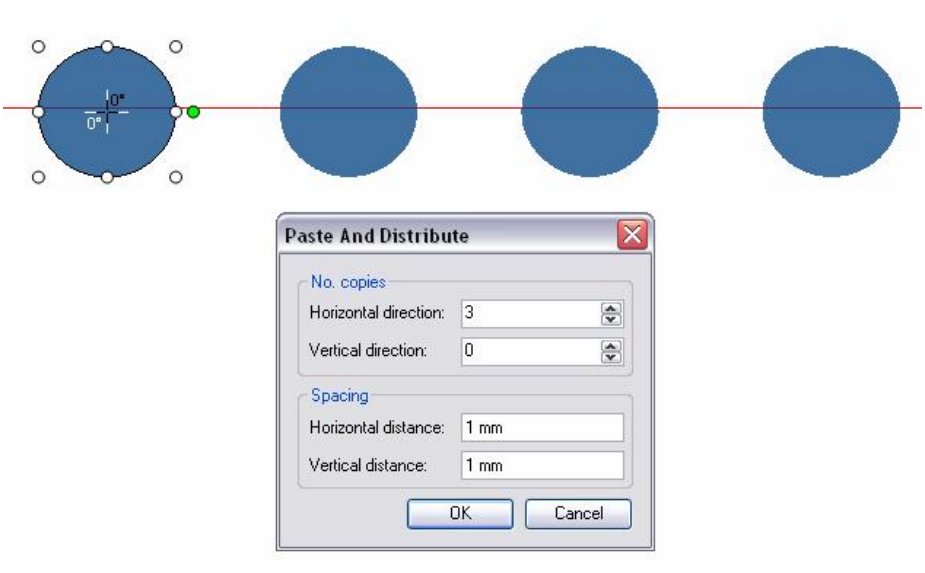
4. Editace elementu.

Jestliže nakreslený element neodpovídá původním představám, lze jeho parametry změnit v panelu vlastností, nebo aktivováním nástroje *Select and edit.* Tento nástroj převede vybraný element do editačního módu, kdy pomocí řídících bodů můžeme měnit jeho natočení, velikost a pozici elementu.



Obrázek A.5: Editace elementu..

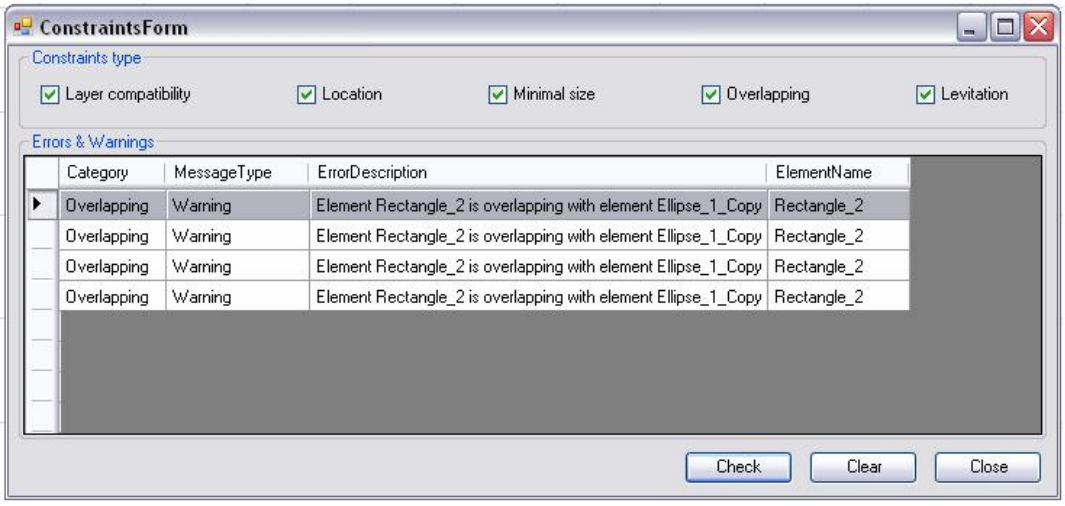
Potřebujeme-li vytvořit kopie elementu, je možno tak učinit přes copy&paste, nebo pomocí nástroje *Distribute*. Pro náš návrh rozkopírujeme nakreslenou elipsu ve směru x podle obrázku 5.1.6.



Obrázek A.6: Distribute.

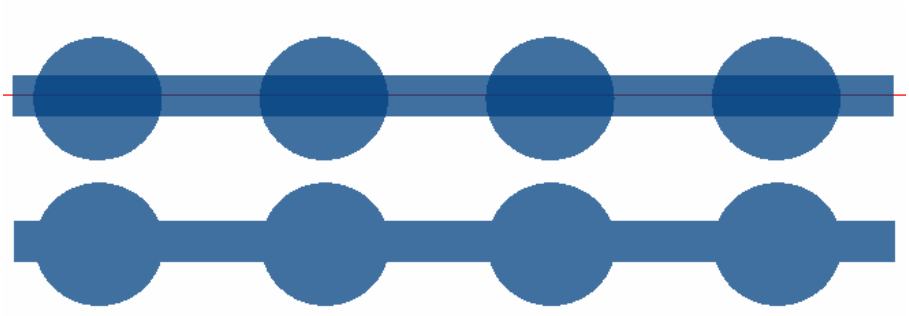
5. Kontrola čipu, cesty

Pro dokončení našeho návrhu by nyní mělo pouze stačit do spodní vrstvy umístit obdélník reprezentující kanálek pro růst neuronových spojů (obr 5.1.9a). Pokud pak nástrojem *Check Constraints* ověříme návrh, dostaneme varování, že se nám jednotlivé objekty v první vrstvě protínají (oblast bude opracována dvakrát).



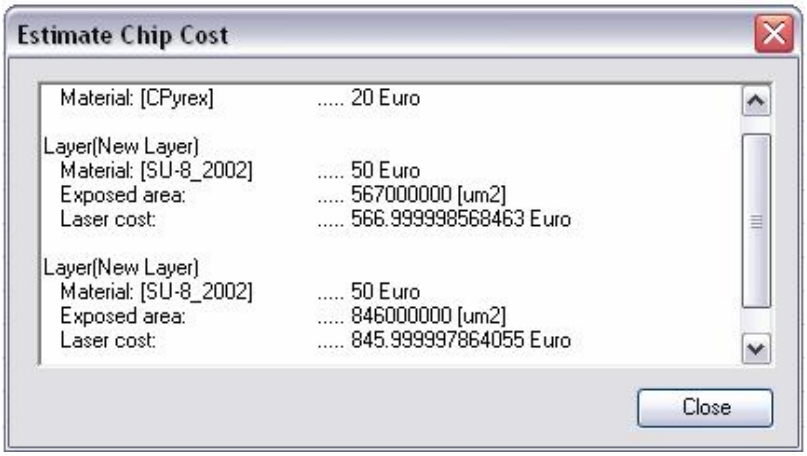
Obrázek A.7:Constrains

Této situaci lze zabránit využitím nástroje *Weld*, který může sloučit díry a kanálek do jednoho složitějšího tvaru. Při použití stačí pouze *Weld* aktivovat a vybrat element, se kterým se má aktivní element sloučit.



Obrázek A..9: Výsledek použití sloučení

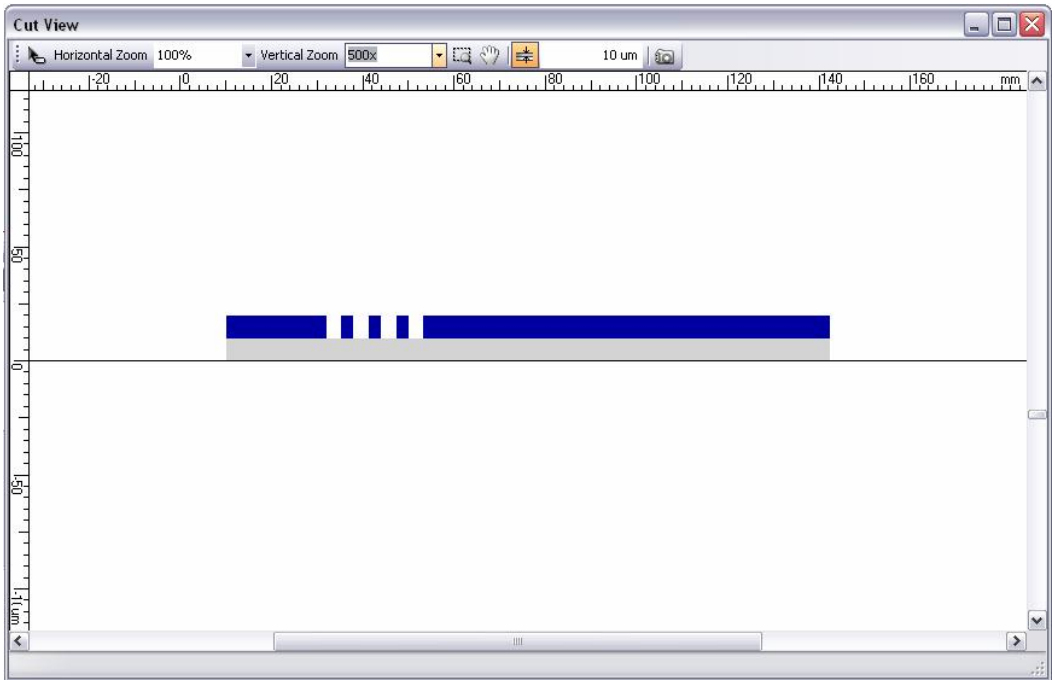
Na závěr můžeme ještě pomocí nástroje *Estaminate chip cost* zobrazit odhad ceny čipu a případně porovnat, o kolik cena klesne, použijeme-li cestu.



Obrázek A..8: Odhad ceny.

6. Cut View

Zaškrtnutím položky *Cut View* v menu *View* zobrazíme náhledové okno 2D řezu. Společně s jeho aktivací se na ploše objeví čára řezu. Její pozice lze měnit pomocí nástroje *Select and edit.* Panel 2D řezu automaticky reflektuje změnu pozice čáry řezu.



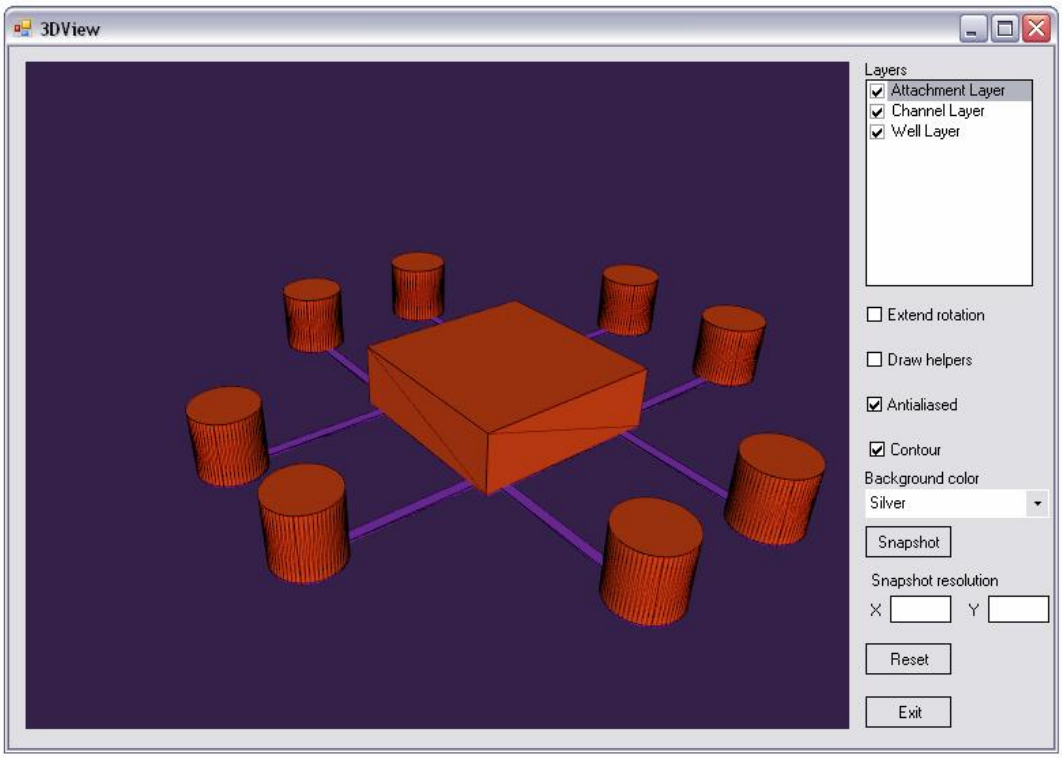
Obrázek A.10: Cut view.form

Protože tloušťka vrstvy je proti velikosti substrátu zanedbatelná, pro lepší zobrazení je možnost měnit vertikální a horizontální měřítko samostatně, nebo lze nastavit pro všechny vrstvy stejnou tloušťku, jakou má substrát.

7. 3D View

Další položkou v menu *View* je 3D zobrazení. Tento příkaz vyexportuje náš návrh a zobrazí ho v dodaném rendereru. Jelikož některé vrstvy se svou tloušťkou mohou blížit hranici zobrazitelných rozměrů, je v nastavení aplikace možnost zapnout exportování vrstev s konstantní tloušťkou.

Zobrazený model lze ovládat pomocí myši, kdy při stisknutém levém tlačítku posunujeme model v rovině *XY*, při pravém rotujeme model dle středu obrazovky a kolečkem myši ovládáme měřítko. U samotného zobrazení můžeme zapnout/vypnout zobrazení jednotlivých vrstev, povolení rotace pod úroveň čipu, pomocné transformační prvky, vyhlazování čáry a zobrazení kontur elementů.



Obrázek A..11: 3D zobrazení skutečného návrhu čipu od koncového uživatel.

Příloha B

Obsah přiloženého CD:

/FlashPomEditor/
- FlashPoM editor v.0.99
- 3D View

/DiplmovePrace/
- Hladík Vojtěch: Diplomová práce; 2D editor pro návrh topologie čipu - projekt FlashPoM – 1

- Novák František: Diplomová práce; 2D editor pro návrh topologie čipu - projekt FlashPoM – 2

/Souce/
- Ukázky zdrojových kodů FlashPom editor v 0.99

/Install/
- Microsoft DirectX 9.0c

- Microsoft Framework 2.0