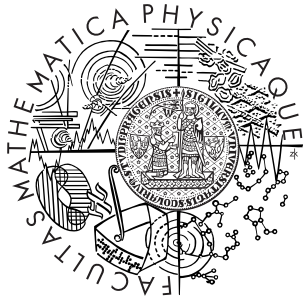
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Stanislav Ulrych

Vektorový editor s podporou vazeb

Katedra softwarového inženýrství

Vedoucí bakalářské práce: RNDr. David Obdržálek, Katedra softwarového inženýrství

Studijní program: Obecná informatika

2007

Děkuji RNDr. Davidu Obdržálkovi za vedení bakalářské práce a jeho cenné rady a zkušenosti.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 25.5.2007 Stanislav Ulrych

Obsah

Název práce: Vektorový editor s podporou vazeb
Autor: Stanislav Ulrych
Katedra (ústav): Katedra softwarového inženýrství
Vedoucí bakalářské práce: RNDr. David Obdržálek
e-mail vedoucího: David.Obdrzalek@mff.cuni.cz

Abstrakt: Tato práce popisuje návrh a implementaci jednoduchého vektorového editoru pro editaci a kreslení vektorové grafiky s možností exportu a částečného importu z vektorového formátu SVG. Narozdíl od většiny vektorových editorů podporuje vytváření vazeb mezi grafickými primitivy, které zachovávají některé poziční vztahy kreslené grafiky.

Klíčová slova: vektorový editor, vektorová grafika, vazby

Title: Vector Graphics Editor With Support For Connections
Author: Stanislav Ulrych
Department: Department of Software Engineering
Supervisor: RNDr. David Obdržálek
Supervisor's e-mail address: David.Obdrzalek@mff.cuni.cz

Abstract: This thesis presents design and implementation of a simple editor of vector graphics with possibility of import and export to SVG (Scalable Vector Graphics) format. In contrast to other vector editors it also supports creation of graphic objects connections that keep some user-defined constraints in the graphics drawn.

Keywords: vector graphics, vector graphics editor, object connection, constraints

1 Úvod

Cílem této práce je navrhnout a implementovat vektorový editor pro kreslení a editaci vektorové grafiky s podporou vazeb mezi objekty. V rámci tohoto návrhu analyzovat možnosti použití vazeb mezi objekty a navrhnout a implementovat potřebné struktury v editoru. Tyto struktury by měly být implementovány dostatečně obecně na to, aby bylo možné implementovaný model v budoucnosti jednoduše nahradit jiným, nejlépe i za běhu programu, popřípadě používat více modelů najednou na základě rozhodnutí uživatele.

Kromě ukládání nakreslené grafiky do vlastního formátu bude editor umožňovat také export do vektorového formátu SVG (Scalable Vector Graphics, viz [5]), a dále export do rastrového formátu PNG (Portable Network Graphics, viz [8]).

Cílová platforma editoru bude především Win32, ale budoucí portace na ostatní operační systémy (především rodina systému Linux) by neměla být vyloučena žádnou z použitých technologií.

Členění práce je následující: v kapitole 2 je uvedena motivace. Návrh editoru, editačních operací a funkcí vazeb je popsán v kapitolách 3 a 4. Hlavní rysy implementace jsou shrnuty v kapitole 5. V kapitole 6 je stručně popsáno uživatelské rozhraní a ovládání editoru. Kapitola 7 obsahuje náměty na rozšíření funkcí editoru.

2
Motivace

Význam grafického designu v poslední době stoupá. Narozdíl od minulých let, kdy průměrná kvalita zobrazení většině uživatelů postačovala, současná doba přináší nároky daleko větší. Co nejdokonalejší vykreslení obrázků v dokumentech, působivé grafické efekty v prezentacích a aplikacích jsou dnes již samozřejmostí, neboť lákají potenciální uživatele i zákazníky. To je také hlavním důvodem, proč se stále více setkáváme s používáním vektorové grafiky místo rastrové. Vektorová grafika na rozdíl od bitmapové nepopisuje barvu jednot-

(a) *rastrový obrázek* 100×100 *bodů*

(b) *zvětšen* $2\times$

Obrázek 2.1. *Příklad zvětšení rastrovaného obrázku*

livých bodů obrázku, avšak dekomponuje obrázek na základní grafická primitiva (např. křivky, viz dále) a ty uloží ve formě jejich analytického vyjádření.

Editace vektorové grafiky nezhoršuje kvalitu výsledku. Například zvětšený rastrový obrázek vypadá bez dalších úprav podstatně hůře než originál, protože originál neobsahuje dostatečné množství informací pro správné vykreslení obrázku většího, než je (viz obrázek

2.1). Naproti tomu zvětšením vektorové grafiky se její kvalita nezhorší. Navíc mohou být dokonce vykresleny nové detaily, které dříve nebyly patrné.

Další výhodou je také obvykle lepší využití rozlišení výstupního zařízení — displeje, tiskárny (viz obrázek 2.2), neboť vektorová grafika je většinou rasterizována až při zobrazení či tisku.

(a) *rastrový obrázek* 200 × 200 *bodů*

(b) *vektorová grafika*

Obrázek 2.2. *Příklad rastrového a vektorového obrázku při tisku*

Hlavní výhodou vektorové grafiky je ovšem možnost editace. Zatímco u rastrového obrázku je velmi obtížné editovat pouze část obrázku — například posunout čáru, za níž se nachází složité pozadí — ve vektorové grafice je tento úkol snadný, celá grafika se skládá z jednotlivých primitiv, které je možné editovat nezávisle a mnohem lépe (např. změna křivky atd.).

3 Návrh

V této kapitole je popsán návrh editoru a editačních operací. Grafické objekty se ve vektorové grafice nazývají základní grafická primitiva. Jsou to: čtverec, obdélník, čára, lomená čára, uzavřená lomená čára (polygon), kruh, elipsa, křivka a uzavřená křivka. U uzavřených primitiv (všechna předchozí s výjimkou čáry, lomené čáry a neuzavřené křivky) může být určena jejich výplň, tj. jak má být vykreslena plocha, kterou ohraničují.

V našem editoru navrhujeme další pomocný objekt — vazbu. Vazba spojuje několik grafických primitiv a přenáší pohyb jednoho primitiva na ostatní. Způsob, jakým pohyb přenáší, závisí na typu vazby a bude popsán později. Vazba je ke grafickým primitivům připojena na jejich definující body (viz dále).

3.1 Cíle návrhu

Vektorový editor Elastique by měl představovat jednoduchý editor pro kreslení a editaci vektorové grafiky. Oproti současným grafickým editorům se liší ve snaze poskytovat možnost vytváření vazeb mezi grafickými primitivy.

Při návrhu a implementaci bude kladen důraz zejména na

- návrh a implementaci vazeb mezi objekty, které většina současných 2D grafických editorů neposkytuje,
- jednoduchost a efektivnost kreslících operací,
- intuitivnost grafického rozhraní,
- rychlost,
- platformní nezávislost,
- minimální požadavky na hardware a software cílové platformy,
- snadnou rozšiřitelnost,
- poskytnutí importu a exportu do některého ze standardních vektorových formátů a exportu do některého běžně používaného rastrového formátu.

Následující cíle nejsou stěžejní a bude na ně brán ohled jen do té míry, aby v návrhu a implementaci nebyly znemožněny:

- implementovat co největší a nejrozsáhlejší množinu grafických objektů a editačních operací,
- vytvořit plnohodnotnou náhradu za existující (komerční i nekomerční) editory,
- konkurovat existujícím (komerčním i nekomerčním) editorům.

3.2 Charakteristika editoru

Předpokládaným použitím editoru je kreslení jednoduché vektorové grafiky, proto bude podporovat kreslení základních grafických primitiv — kruhu, elipsy, čtverce, obdélníku, lomené čára, křivky, uzavřené křivky, spolu se základními editačními operacemi — posunem, zvětšením, rotací, mazáním a nastavováním vlastností primitiv — stylu obrysu a výplně.

Pro snadnou rozšiřitelnost editoru je nutné dostatečné zobecnění jednotlivých objektů a operací. Zobecněním grafických objektů bude abstraktní třída obsahující všechna potřebná pravidla a parametry dostatečně specifikující instance konkrétních objektů. Zobecnění editačních operací bude spočívat:

- v implementaci nezávislých objektů reprezentujících jednotlivé operace,
- v rozvrstvení podle účelu (základní matematické operace, komplexní operace pro vazby, uživatelské operace atd.).

První bod je důležitý zejména pro snadnou rozšiřitelnost, druhý pro vyhnutí se psaní duplicitního kódu a efektivnějšího vytváření komplexních operací s použitím operací základních.

Pro správný návrh vazeb je nutné specifikovat požadavky a omezení. Vazby budou spojovat několik objektů a přenášet pohyb jednoho objektu na ostatní vazáné objekty.

Některá grafická primitiva svým tvarem určují vzájemný vztah svých bodů (např. editací čtverce vznikne opět čtverec), a taková omezení by mohla kolidovat s omezeními vynucovanými vazbami mezi body téhož objektu. Proto nebude možné vytvářet vazby spojující body téhož objektu.

Objekty propojené vazbami je možné vnímat jako graf, kde množinu vrcholů tvoří objekty a množinu hran vazby. Tyto grafy budeme nazývat obrazce.

Pokud nestanovujeme žádné omezení na chovaní vazeb, nelze pro graf obsahující cykly (více různých cest mezi dvěma objekty) předem definovat postup výpočtu pro přenášení pohybu přes vazby. Kromě toho, i některé konkrétní operace (např. zvětšení) lze pro obecný graf těžko definovat (viz dále). Proto řešení omezíme pouze na obrazce bez cyklů.

Pokud by nestačily pouze obrazce bez cyklů, šlo by řešení rozšířit na orientované vazby přenášející pohyb pouze jedním směrem a vyloučit obrazce s orientovanými cykly.

3.3 Typy a funkce vazeb

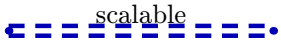
Použití vazeb ve vektorových editorech nebylo podle dostupných informací doposud řešeno, nebo jen ve formě specifikací omezení (např. Autodesk Inventor) nebo jednoduchých

propojovacích prvků (tzv. SmartConnectors) v editoru Microsoft Visio, který ovšem není vektorovým editorem.

Pro základní editační operace by měly postačovat vazby spojující dva objekty. Ná základě analýzy požadovaného chování jsme definovali následující typy a chování vazeb (viz také obrázek 3.1):

fixed: záchovává směr i délku,

scalable: zachovává směr, ale ne délku,

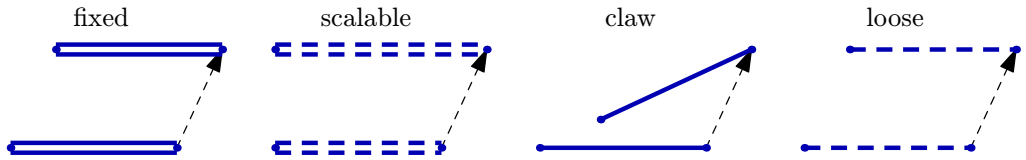


claw: zachovává délku, ale ne směr,

loose: Pokud obrazec nemá ukotvenou větev (viz níže), chová se jako vazba fixed, její speciální funkce bude popsána pozdějí.

Navíc jsme definovali vazbu fixující bod objektu proti pohybu vůči kreslící ploše:

nail: hovoříme pak o obrazci s ukotvenou větví.



Obrázek 3.1. *Chování jednotlivých vazeb (šipka určuje směr pohybu s jedním koncem vazby)*

3.4 Pohyb s obrazcem

Pohyb s objektem bude přenášen dále do obrazce prostřednictvím vazeb (ze zdrojového konce do cílového konce), a to následovně:

fixed: pohyb zdrojového konce vyvolá stejný pohyb v cílovém konci,

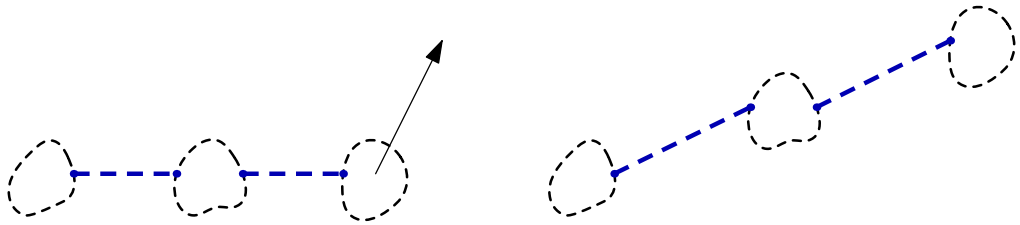
scalable: na cílový konec je přenášena pouze složka kolmá na směrový vektor vazby,

claw: cílový konec je posunut ze své původní pozice ve směru pohybu objektu (tj. po přímce směrem k nové pozici zdrojového konce) tak, aby byla zachována konstantní délka vazby,

loose: pokud obrazec není ukotven vazbou nail, pak se chová stejně jako vazba fixed, neboť jinak by v tomto případě vazba měnila svoji délku a směr a vzhledem k jejímu účelu to není žádoucí.

Pokud má obrazec ukotvenou větev, je pohyb s koncovou částí větve rovnoměrně rozprostřen podél celé větve. Objekty na této větvi jsou posunuty tak, aby:

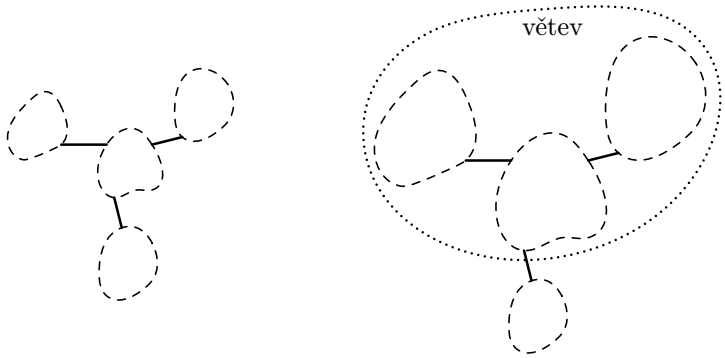
- délka vazeb fixed a claw zůstala zachována,
- bod ukotvený vazbou nail zůstal na místě,
- byl zachován směr vazeb scalable,
- pohybový vektor byl rovnoměrně rozprostřen po celé cestě, podle délek vazeb scalable a loose.



Obrázek 3.2. *Rozprostření pohybu vazbami* loose

3.5 Zvětšení obrazce

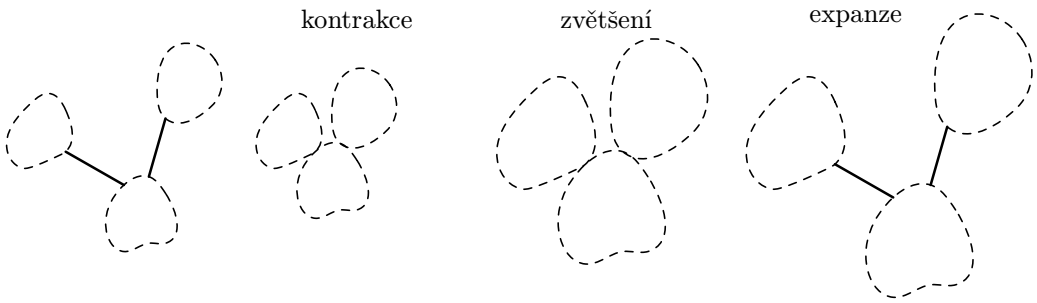
Základním požadavkem na operaci zvětšení je homogenita (pro každý bod obrazce platí $z_{scaled} = \beta z_{original}$). V okamžiku zavedení vazeb konstantní konstatní délky je ovšem nemožné požadavky homogenity pro celý obrazec dodržet.



Obrázek 3.3. *Zvětšení obrazce*

Operace zvětšení je tedy aplikována pouze na jednu uživatelem zvolenou cestu, na níž jsou fixní objekty kontrahovány a ostatní objekty zvetšeny stejným koeficientem, a poté jsou kontrahované objekty navráceny do původní podoby (viz obrázek 3.3). Ostatní větvě připojené k cestě jsou posunuty podle definice pohybu s obrazcem v sekci 3.4.

Při praktickém testovaní po implementaci předchozího postupu vznikla dvě další možná řešení: kontrahovat všechny vazby fixní délky (fixed a claw) v obrazci, provést zvětšení vzniklého obrazce a obnovit kontrahované vazby do původní podoby (znázorněno na obrázku 3.4). Dalším možným řešením by mohlo být zvětšování všech objektů, včetně vazeb konstantní délky, neboť v praxi by i toto mohlo být užitečné. Tato dvě řešení už nebyla implementována, nicméně architektura pozdější implementaci nevylučuje.



Obrázek 3.4. *Alternativní řešení zvětšení obrazce*

3.6 Rotace obrazce

Základním požadavkem na rotaci objektu je stejná úhlová rychlost pro všechny body rotovaných objektů. V případě vazeb nezachovávajících svůj směr (lze si představit, že úchyty vazeb jsou otočné) tento požadavek ale nemá smysl. Proto vazba claw přeměňuje rotační pohyb na pohyb přímočarý, ostatní vazby se při rotaci chovají jako vazba fixed, tj. přenášejí rotační pohyb. Pokud obrazec obsahuje ukotvenou větev, není možné s ním rotovat.

4 Koncepce řešení

Tato kapitola obsahuje podrobný popis funkčnosti vazeb v editačních operacích.

V matematickém popisu editačních operací použijeme následující značení: Jednotlivé body budeme reprezentovat pomocí dvourozměrných vektorů, tj. $p \in \mathbb{R}^2, p = (p_1, p_2)$. Na množině bodů $\mathbb{R}^2$ pak definujeme následující operace

- součet, rozdíl a součin vektorů po složkách,
- násobení skalárem,
- skalární součin vektorů: $\langle \cdot, \cdot \rangle : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}, \quad \langle a, b \rangle = a_1 b_1 + a_2 b_2,$
- vektorový součin vektorů: $\cdot \times \cdot : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}, \quad a \times b = a_1 b_2 - a_2 b_1,$
- norma vektoru: $\| \cdot \| : \mathbb{R}^2 \to \mathbb{R}, \quad \|a\| = \sqrt{a_1^2 + a_2^2}.$

4.1 Rozklad vektoru vzhledem k jiné bázi

V následujících výpočtech je často potřeba určit složky vektoru $z = (z_1, z_2)$ v bázi dané vektory a a b ($a \neq kb$), tj. nalézt hodnoty z_a, z_b a α, β tak, aby platilo $z = z_a + z_b$, kde $z_a = \alpha a, \ z_b = \beta b$ (situace na obrázku 4.1).

Vyjdeme tedy z rovnice pro vektor z

$$z = \alpha a + \beta b,$$

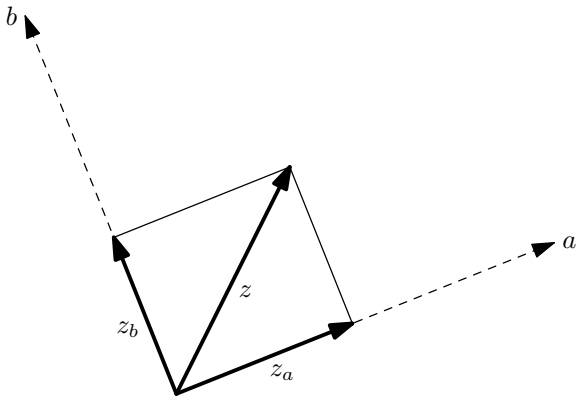
vynásobením rovnice vektorem $(a_2, -a_1)$, sečtením jejích složek a úpravou dostáváme

$$\beta = \frac{a_2 z_1 - a_1 z_2}{a_2 b_1 - a_1 b_2} = \frac{z \times a}{b \times a}.$$

Podobně pro α spočítáme $\alpha = \frac{b \times z}{b \times a}$. Tím dostaneme výsledné vztahy pro složky z_a, z_b

$$z_a = a \frac{b \times z}{b \times a}, \qquad z_b = b \frac{z \times a}{b \times a}.$$

V mnoha případech, které se dále vyskytnou, platí, že $b = a^{\perp}$ (ortogonální báze), a v tomto případě počítáme ortogonální projekci vektoru z na vektor a, což budeme značit



Obrázek 4.1. *Rozklad vektoru na složky v jiné bázi*

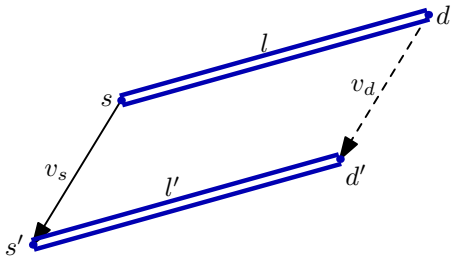
$z|_a$. Vzorec lze v tomto případě zjednodušit na

$$z|_a = a\frac{a^\perp \times z}{a^\perp \times a} = (z \times a^\perp)\frac{a}{\|a\|}.$$

4.2 Pohyb s obrazcem

Pohyb s obrazcem bez ukotvené větve je jednoduchou operací, která je následně využívána při dalších editačních operacích. Pohyb bodu některého z objektů obrazce se přenáší dále do obrazce prostřednictvím vazeb, které určují, jakým způsobem bude pohybový vektor přenesen z jednoho konce vazby na druhý.

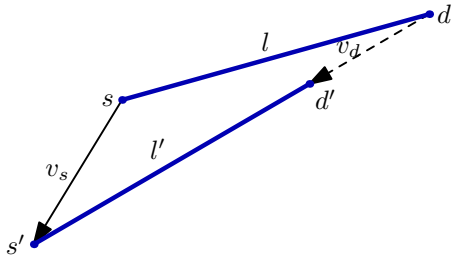
Nejjednodušším případem je vazba fixed: pohyb se zdrojovým koncem vazby s o vektor v_s do bodu s' vyvolá stejný pohyb druhého konce d o vektor $v_d = v_s$ (obrázek 4.2).



Obrázek 4.2. *Přenos pohybu přes vazbu* fixed

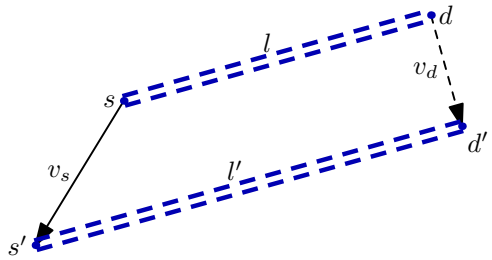
V případě vazby claw pohyb se zdrojovým koncem vazby z bodu s do bodu s' vyvolá pohyb s koncovým bodem d ve směru s' tak, aby délka vazby $\|l'\| = \|d' - s'\|$ zůstala konstantní (obrázek 4.3). Při kontinuálním pohybu se zdrojovým bodem je finální pozice objektů určena opakovaným použitím předchozího výpočtu relativně vzhledem k pozici z předchozí iterace, čímž se vytvoří iluze, že je vazba připevněná přes otočné úchyty. To

je také důvod, proč se při vrácení zdrojového konce na počáteční pozici cílový konec na svoji počáteční pozici dostat nemusí. Další možností by mohl být výpočet vždy vzhledem k počáteční pozici.



Obrázek 4.3. *Přenos pohybu přes vazbu* claw

Vazba scalable přenáší na cílový konec pouze složku vektoru v_s kolmou na směrový vektor vazby. Označíme-li vektor vazby l, pak pohyb cílového konce vazby je spočítán jako $v_d = v_s|_{l^\perp}$ (obrázek 4.4).



Obrázek 4.4. *Přenos pohybu přes vazbu* scalable

Vazba loose se v případě obrazce bez ukotvené větve chová stejně jako vazba fixed.

4.3 Pohyb s obrazcem s ukotvenou větví

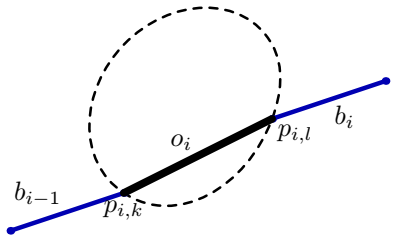
Pro danou cestu $P = o_0 b_1 o_1 b_2 o_2 \dots o_n$, kde o_i je objekt a b_i je vazba, definujme množiny $B_i = \{b_1, \dots, b_i\}$, pro $i \in \{1, \dots, n\}$. Vazba b_i pak spojuje objekt o_{i-1} a o_i a to v bodech $p_{i-1,k}$ a $p_{i,l}$. Označením b_i v následujících vzorcích budeme rozumět vektor vazby $b_i = p_{i-1,k} - p_{i,l}$ (obrázek 4.5).

Dále definujme (pro zjednodušení) součet vazeb na dané cestě až po i-tý objekt (obrázek 4.6) jako

$$l_{(i)} = \sum_{b \in B_i} b,$$

dále také součet roztažitelných vazeb resp. volných vazeb jako

$$l^s_{(i)} = \sum_{b \in B_i, b \in scalable} b, \qquad l^l_{(i)} = \sum_{b \in B_i, b \in loose} b,$$

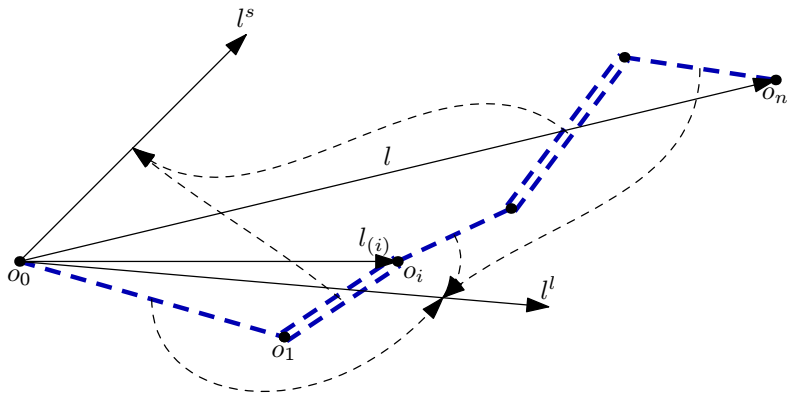


Obrázek 4.5. *Definice vektoru objektu o_i*

a nakonec také součet jednotlivých vazeb na celé cestě

$$l = l_{(n)}, \qquad l^s = l^s_{(n)}, \qquad l^l = l^l_{(n)}.$$

Zdrojem pohybu s větví obrazce je změna polohy objektu o_n o vektor v, která se pomocí vazeb musí přenést na ostatní vázané objekty. Objekty se musí posunout o příslušný násobek vektoru v při zachování parametrů vazeb a objekt vázaný vazbou nail (o_0) musí zachovat svoji pozici. Kromě toho, musí být zachován směr vazeb scalable.



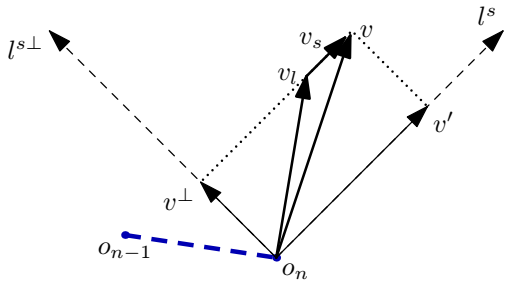
Obrázek 4.6. *Definice potřebných vektorů (objekty zkontrahovány do bodů)*

Pokud označíme vektor pohybu s počátečním bodem jako v, pak tento vektor nejprve rozložíme na složku rovnoběžnou s vektorem vazeb scalable v' a složku k ní kolmou $v^\perp$. Tento rozklad je nutný proto, že vazby scalable musí zachovávat svůj směr.

$$v^\perp = v|_{l^{s\perp}}, \qquad v' = v|_{l^s}.$$

Nyní je třeba spočítat, jak se jednotlivé složky podílí na pohybu vazeb scalable a loose. Složka $v^\perp$ nemá žádný vliv na vazby scalable a složku v' je třeba rozdělit podle norem vektorů vazeb scalable a loose (obrázek 4.7)

$$v^l = v^\perp + v' \frac{\|l^l\|}{\|l\|}, \qquad v^s = v' \frac{\|l^s\|}{\|l\|}.$$



Obrázek 4.7. *Výpočet vektorů*

Část v^l se uplatní při pohybu vazeb loose, část v^s při pohybu vazeb scalable. Zbývá už jen spočítat pohybový vektor $v_{(i)}$ objektu o_i. Nejprve spočítáme podíl předchozích dvou složek příslušný vazbám od objektu o_0 až po o_i.

$$v^l_{(i)} = v^l \frac{\|l^l_{(i)}\|}{\|l^l\|},$$

$$v^s_{(i)} = v^s \frac{\|l^s_{(i)}\|}{\|l^s\|},$$

Výsledný vektor pohybu objektu o_i je pak

$$v_{(i)} = v^s_{(i)} + v^l_{(i)}.$$

4.4 Zvětšení obrazce s ukotvenou větví

Definujme také množiny $O_i = \{o_0, \ldots, o_i\}$, pro $i \in \{0, \ldots, n\}$. Ve výpočtech budeme označením o_i rozumět vektor vzhledem k cestě P definovaný následovně za předpokladu, že k jeho bodům $p_{i,k}$ a $p_{i,l}$ jsou přichyceny vazby b_{i-1} a b_i (obrázek 4.5), $fixed$ značí bod ukotvený vazbou nail a $origin$ bod, za který je obrazec zvětšován:

$$o_i = \begin{cases} p_{i,k} - fixed & i = 1, \\ p_{i,k} - p_{i,l} & 1 < i < n, \\ origin - p_{i,l} & i = n. \end{cases} \tag{4.1}$$

Pro zjednodušení opět zaveďme značení

$$o_{(i)} = \sum_{o_k \in O_i} o_k, \qquad l^f_{(i)} = \sum_{b_k \in B_i, b_k \in fixed} b_k,$$

$$o = o_{(n)}, \qquad l^f = l^f_{(n)}.$$

Střed $center_{(i)}$ a koeficient zvětšení $alpha$ objektu i jsou spočítány podle vzorců

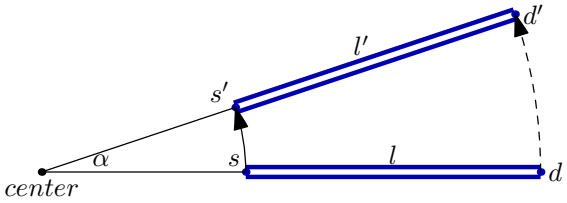
$$center_{(i)} = fixed - l^f_{(i)},$$

$$\alpha = \frac{\|fixed - origin - v - l^f\|}{\|o + l^s\|}.$$

Zvětšování pak probíhá s koeficientem α a se středem $center_{(i)}$ — bodem obrazce ukotveným vazbou nail posunutým o vektor vazeb fixed.

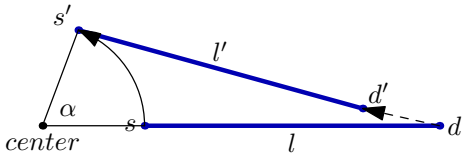
4.5 Rotace obrazce

Při rotaci je zadán střed *center* a úhel α rotace. Připomeňme, že vazby scalable se v případě bez ukotvené větvě chovají jako fixed. Z toho plyne, že rotační pohyb zdrojového konce všech vazeb s výjimkou claw vytváří rotační pohyb cílového konce, tj. cílový konec je rotován také kolem středu *center* o úhel α (obrázek 4.8).



Obrázek 4.8. *Rotace vazeb* fixed, scalable *a* loose

Jedinou výjimkou je vazba claw, v jejímž případě se rotační pohyb se zdrojovým koncem přemění na posun (obrázek 4.9) podle definice v kapitole 3.4.



Obrázek 4.9. *Rotace vazby* claw

5 Implementace

V této kapitole jsou shrnuty základní principy a klíčové části implementace. Kompletní programátorská dokumentace se nachází v balíčku spolu s editorem Elastique[1].

5.1 Základní grafická primitiva

Pro kreslení vektorové grafiky je potřeba implementovat základní grafická primitiva. Na základě zkušeností s ostatními vektorovými editory a směru vývoje formátu SVG byly implementovány následující:

- lomená čára,
- čtverec, obdélník,
- kruh, elipsa,
- uzavřená a neuzavřená křivka složená ze segmentů Bézierových křivek.

Abstraktní třída EPrimitive je rodičovskou třídou pro všechny grafické objekty editoru, které se vykreslují na kreslící ploše editoru. Obsahuje všechny potřebné vlastnosti specifikující objekty. Kromě grafických primitiv jsou od ní odvozeny také rodičovské objekty pro vazby (EBond), křivky (EPath) a pomocné objekty editoru (EGadget).

5.2 Kreslící plocha

Hlavní částí editoru je kreslící plocha, která je reprezentována třídou EScene. Obsahuje seznam grafických primitiv (potomky EPrimitive), které vykresluje, aktuální výběr objektů (reprezentovaný třídou ESelection), seznam uživatelem prováděných operací (potomci EModificator, viz kapitola 5.6) a další vlastnosti. Jejím hlavním účelem je vykreslovat primitiva v kreslící ploše a přeposílat události prováděným operacím.

[1] Na přiloženém CD, viz popis obsahu na konci této práce

5.3 Výběr

Vybrané objekty tvoří výběr, tento výběr je reprezentován třídou ESelection. Vybrán může být buď celý objekt, nebo jeho konkrétní body. Informace o výběru pro jednotlivé objekty jsou obsaženy ve třídě ESelectionItem. Třída ESelection obsahuje jejich seznam a je tedy jakýmsi chytrým seznamem grafických primitiv, který je použit také k vykonávání libovolné operace nad skupinou objektů i jednotlivými objekty. K tomuto účelu slouží metoda const int ESelection::execute(EFunctorInterface &), které se jako parametr předá instance funktoru vykonávané operace (implementace rozhraní EFunctorInterface, viz kapitola 5.4).

5.4 Základní matematické operace

Z hlediska snadné rozšiřitelnosti editoru a efektivity implementace je důležité, aby množina operací byla rozšiřitelná bez nutnosti větších zásahů do existujícího kódu editoru.

Proto jsou v editoru Elastique hojně využívány funktory (třída EFunctorInterface), které reprezentují základní (většinou matematické) operace prováděné nad grafickými primitivy — posun, rotace, zvětšení, ale i nastavování stylu a stavu objektů aj.

Cílem je, aby se všechny editační operace vykonávaly pomocí těchto funktorů. To usnadní nejen snadnou rozšiřitelnost operací, ale také možnost v budoucnosti implementovat do editoru skriptovací jazyk, kterým se budou volat funktory.

Další výhodou tohoto přístupu je, že základní operace jsou implementovány jen zde, spolu s ošetřením podmínek pro volání, vlastností primitiv, akcí nutných vykonat před a po dané operaci atd. Kdekoli jinde jsou již pro potřebné operace používány funktory. Vytváření složitějších editačních operací (např. pro implementaci výpočetního modelu vazeb) se pak více podobá skládání jednotlivých základních operací a kód pro podobné operace není nikde duplikován.

Pokud se volání funktorů neimplementuje pro jednotlivé grafické objekty, ale pro jejich seznam, což je v našem případě výběr některých objektů, získáme vhodný a efektivně použitelný základ pro práci s jednotlivými objekty i s vícenásobným výběrem najednou pomocí jednotného volacího mechanismu.

5.5 Výpočetní model vazeb

Jedním z cílů implementace vazeb je, aby bylo možné navrženou implementaci upravit, popř. nahradit implementací jinou. Základní idea implementace v editoru Elastique je taková, že vazby samy o sobě neobsahují žádnou "vlastní inteligenci", ale pouze své parametry a úchyty (odkaz na vázané body objektů).

Je pak plně na používaném výpočetním modelu (implementovaném třídou EEngine nebo jejími potomky), jak bude na základě vlastností vazeb realizovat výpočty jednotlivých operací, přičemž může nebo nemusí parametry vazeb respektovat.

Díky tomuto přístupu je možné výpočetní model jednoduše nahradit jiným. Dokonce by bylo možné používat více výpočetních modelů současně a nechat uživatele vybírat, který si přeje právě použít, a to vše za běhu aplikace.

Třída EEngine implementuje třídu EFunctorInterface, takže zhlediska svého volání se chovají totožně jako funktory, a tudíž není potřeba měnit implementovaný mechanismus volání funktorů (avšak implementace grafických primitiv musí s použitím vazeb počítat).

Třída EEngine i její potomci by měli konkrétní základní operace (posun, zvětšení aj.) vykonávat prostřednictvím mechanismu volání funktorů přes výběr.

Je vhodné rozčlenit funkce pro výpočet ve třídě EEngine tak, aby byly vhodné části implementovaného výpočtu použitelné z jiné části třídy. Ostatní větvě připojené k ukotvené větvi je třeba posunout podle definice posunu se stromem. V současné implementaci to jednoduše znamená, že se na každé větvi připojené k ukotvené větvi vykoná EMoveEngine reprezentující posun se stromem.

5.6 Uživatelská část editačních operací

Jak již bylo uvedeno dříve, základní operace nad objekty jsou implementovány ve funktorech. Je důležité si uvědomit, že funktory reprezentují skutečně pouze matematickou část dané operace. Uživatelskou část editační operace (např. pohyb myši, úprava pozice bodu v dialogu atd.) je nutné nejprve převést na čistě matematické údaje očekávané jako parametry funktoru.

Vzhledem k požadavku, aby byly operace snadno rozšiřitelné bez nutnosti úpravy existujícího kódu, je nutné implementovat takovou transformaci mimo základní objekt editoru, třídu EScene, a vzhledem k návrhu funktorů i mimo ně.

Transformaci mezi uživatelským ovládáním editoru (pohyb myši, vstup z klávesnice aj.) proto reprezentuje jiná vrstva, a to potomci třídy EModificator. Tyto třídy přijímají události od myši a klávesnice (popř. jiné) prostřednictvím scény, převádějí je na matematickou reprezentaci (např. místo pohybu myši na delta vektor pohybu objektů) a řídí volání funktorů jednotlivých operací.

5.7 Přenositelnost

Editor je implementován v jazyce C++ bez proprietárních vlastností konkrétního kompilátoru s použitím grafické knihovny GTK, jedinou podmínkou pro přenositelnost na cílovou platformu je proto přítomnost této knihovny a kompilátoru jazyka C++. Portace tedy není vyloučena pro většinu současných platforem.

V současné době editor používá knihovnu GTK verze 2.8 a je zkompilován a testován na platformách Win32 (MS Windows XP) a linux (Gentoo linux).

5.8 Knihovna GTK

Knihovna GTK (viz [2] a [3]), která byla zvolena pro vytvoření grafického uživatelského rozhraní, není objektová, a proto byl vytvořen objektový minimální wrapper EGtk zapouzdřující její funkce do objektové podoby. S výjimkou volání funkcí části cairo knihovny GTK a správy signálů jsou všechna volání do GTK uskutečňována přes tento wrapper, nebylo by tedy ani příliš náročné implementovat editor s použitím jiné grafické knihovny než GTK, a to přepsáním EGtk pro jinou grafickou knihovnu.

5.9 Import a export do formátu SVG

Pro práci s formátem SVG je použita knihovna TinyXML (viz [6]). TinyXML obsahuje jednoduchý parser XML dokumentů do struktury DOM, který pro import SVG postačuje. Výhodou použití této knihovny je, že její zdrojový kód lze lehce integrovat do zdrojového kódu programu a na cílové platformě nevzniká závislost na další knihovně.

6 Editor

V této kapitole jsou shrnuty základní funkce a vlastnosti editoru. Kompletní uživatelská dokumentace se nachází v balíčku spolu s editorem Elastique[1].

6.1 Uživatelské rozhraní

Editor se skládá z:

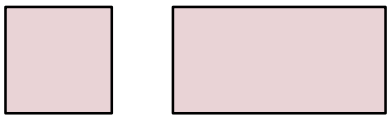
- kreslící plochy
- menu a panelu nástrojů
- panelů pro nastavení — panely Object, Scene a Point v levé, resp. horní části editoru.

6.2 Kreslení

Editor podporuje kreslení základních grafických objektů:

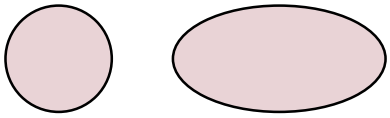
- čára, lomená čára,

- čtverec, obdélník,

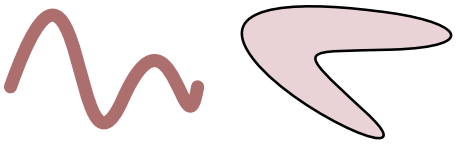


[1] Na přiloženém CD, viz popis obsahu na konci této práce

- kruh, elipsa,



- uzavřená a neuzavřená křivka složená ze segmentů Bézierových křivek.



Kreslení v editoru Elastique probíhá volbou typu grafického objektu v menu Draw či v panelu nástrojů a poté zadáváním jednotlivých jeho bodů klikáním levým tlačítkem myši. Vytváření se ukončí stiskem pravého tlačítka myši.

Pokud má objekt předem pevně daný počet bodů, jeho vytváření se ukončí, pokud jsou všechny zadány (např. u čtverce stačí dva rohové body). V případě, že při přerušení vytváření objektu nebyl zadán dostatečný počet bodů (např. pro čtverec pouze jediný bod), pak je vytvářený objekt smazán. V případě kreslení uzavřené křivky se křivka uzavře.

U každého objektu je možné nastavit pozici, styl obrysové čáry a také, pokud objekt tvoří uzavřenou křivku (čtverec, obdélník, kruh, elipsa a uzavřená křivka), je možné nastavit styl jeho výplně.

6.3 Obrys, výplň a průhlednost objektů

Nastavování parametrů vykreslování objektů probíhá prostřednictvím panelu Object v levé části editoru. U obrysu objektu je možné nastavit parametry (uvedeny anglické názvy jako v editoru):

color: barva obrysové čáry objektu,

alpha: průhlednost barvy obrysu v intervalu $[0, 1]$, 0 pro plně průhlednou, 1 pro neprůhlednou,

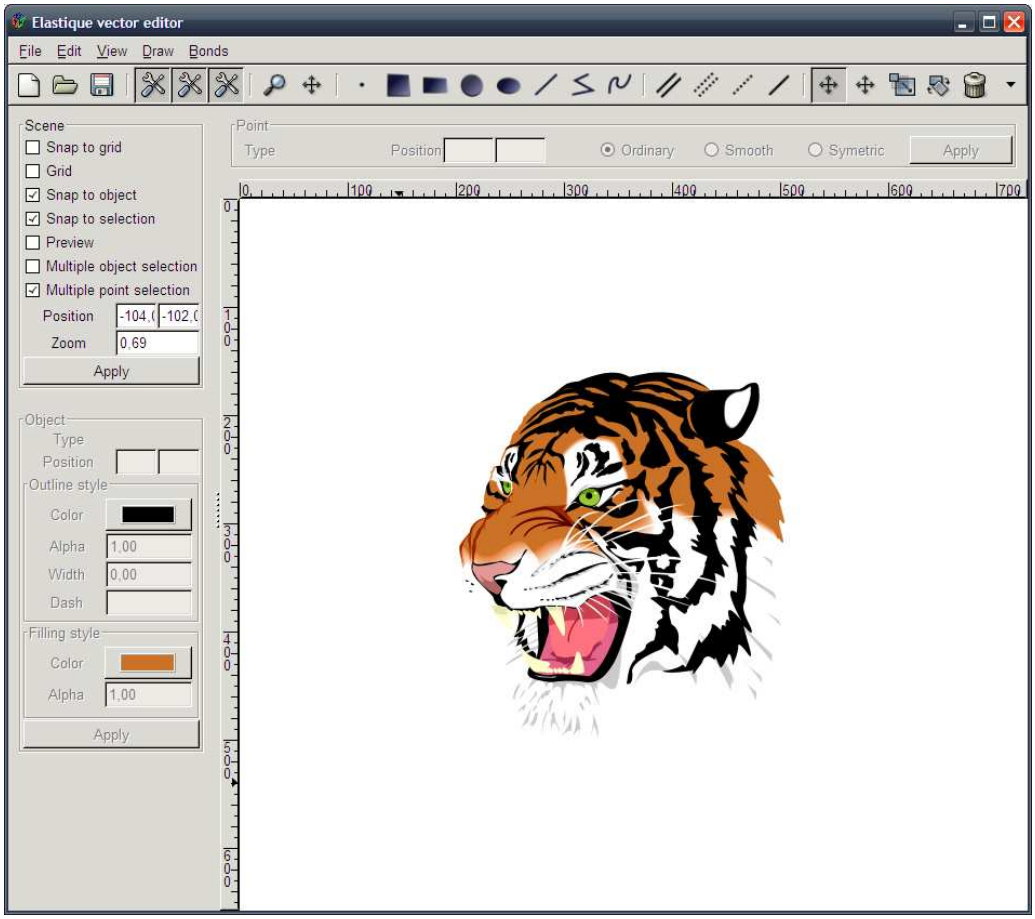
linewidth: šířka obrysové čáry v pixelech,

dash pattern: specifikace čárkování čáry ve formátu $a_1, a_2, ...$, kde a_1 je délka kresleného úseku, a_2 délka vynechaného úseku atd.

U výplně lze nastavit parametry:

color: barva výplně objektu,

alpha: průhlednost barvy výplně v intervalu $[0, 1]$, 0 pro plně průhlednou, 1 pro neprůhlednou.



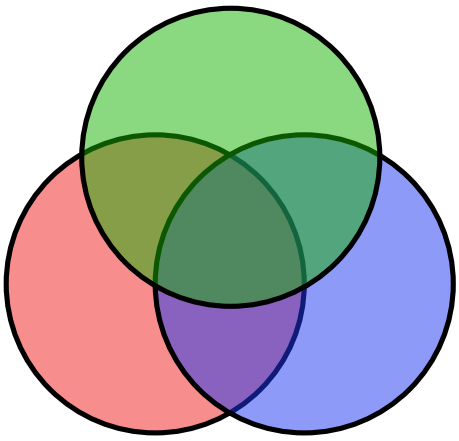
Obrázek 6.1. *Uživatelské rozhraní editoru*

6.4 Editační operace

Editační operace pracují s aktuálním výběrem. Objekty mohou být vybírány před započetím editační operace, ale výběr může být také libovolně měněn v jejím průběhu. Kliknutím mimo výběr se aktuální výběr zruší, kliknutím na objekty se stisknutou klávesou CTRL je možné k aktuálnímu výběru přidávat další objekty.

Objekt je vybrán kliknutím kdekoli do jeho plochy (přesněji do jeho konvexního obalu). Kliknutím na některý bod objektu je spolu s objektem vybrán také tento bod. Některé objekty mají povolen současný výběr více svých bodů najednou (např. křivka či lomená čára). Současný výběr více bodů jednoho objektů je umožněn zapnutím volby Multiple point selection v panelu Scene.

Dalším způsobem je výběr výběrovým obdélníkem. Při kliknutí mimo objekty a tažení se vykresluje výběrový obdélník a po puštění levého tlačítka jsou vybrány objekty zasahující do označené oblasti.



Obrázek 6.2. *Vzájemné překrývání částečně průhledných objektů*

V případě zapnuté volby Multiple object selection jsou nasvěcovány i vybírány všechny body nacházející se pod kurzorem. Pokud je volba Multiple object selection vypnutá, pak se nasvěcuje pouze aktivní objekt, který bude vybrán a klávesou TAB lze z objektů pod kurzorem aktivní objekt volit.

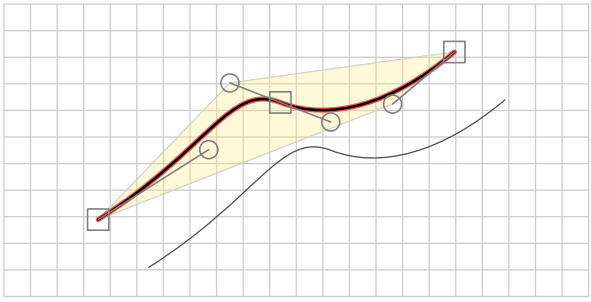
Editor podporuje následující editační úpravy:

- pohyb s objekty,
- pohyb s body objektů,
- zvětšování a změnšování objektů,
- rotace objektů,
- mazání objektů,
- nastavování parametrů objektů.

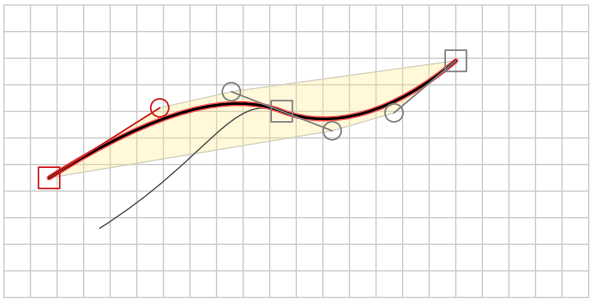
Operace Pohyb s objekty (obrázek 6.3) vždy provádí posun celých objektů bez jejich změny, Pohyb s body (obrázek 6.4)posun s konkrétními body, pokud byly nějaké vybrány. Důvod tohoto rozlišení je, aby uživatel mohl posunovat celý objekt, i když ho uchytí za jeden z jeho bodů.

V případě, že je zapnutá volba Snap to selection, je vybraný bod přichycen přesně na pozici kurzoru (i když souřadnice kurzoru nebyly totožně se souřadnicemi bodu). Pokud je zapnutá volba Snap to object, je kurzor při provádění editační operace přichytáván k jednotlivým bodům ostatních (nevybraných) objektů. Kombinací předchozích dvou voleb je možné například posunout body dvou objektů (nebo dva celé objekty), tak aby se přesně překrývaly. Je možné také zapnout volbu Snap to grid, a potom je kurzor přichytáván také k mříži.

Pro rotaci objektů je nutné nejprve zvolit dvojklikem levého tlačítka střed rotace, který je pak označen značkou (obrázek 6.6). Poté kliknutím na výběr (resp. vybráním některého objektu) a tažením je výběr rotován kolem zadaného středu. Objekty jsou zvětšovány vzhledem ke svým středům (obrázek 6.5).



Obrázek 6.3. *Pohyb s objektem*



Obrázek 6.4. *Pohyb s body objektu*

6.5 Grafické formáty

6.5.1 Export do vektorového formátu SVG

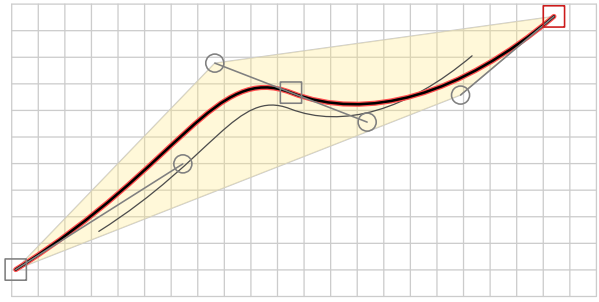
Nakreslenou grafiku lze exportovat do formátu SVG. Pro vazby neexistuje v SVG odpovídající interpretace, a proto jsou exportovány jako čáry. Jejich styl, který má být použit při exportu, je možné nastavit stejně jako pro všechny ostatní grafické objekty.

6.5.2 Export do bitmapového formátu PNG

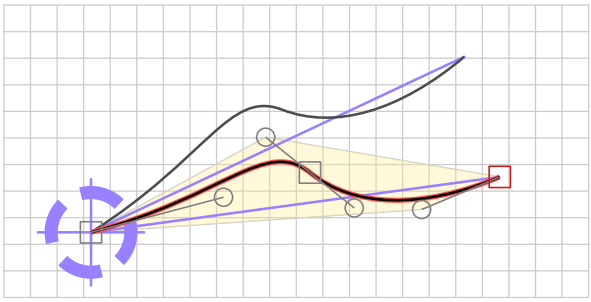
Veškerou grafiku vytvořenou editorem lze exportovat do formátu PNG. Vazby jsou exportovány stejně jako u formátu SVG čárou zvoleného formátu.

6.5.3 Import z vektorového formátu SVG

Grafická data je možné importovat z formátu SVG. Množina podporovaných grafických objektů je prakticky totožná s formátem SVG, proto všechny objekty z formátu SVG je teoreticky možné importovat editorem. Problém však nastává při importu jejich vlastností (atributů), které mohou obsahovat transformační matice souřadnic nebo odkazy na styly ve formátu CSS apod.



Obrázek 6.5. *Zvětšení objetku*



Obrázek 6.6. *Rotace objektu*

SVG je rozsáhlý a sofistikovaný formát, který je schopen pojmout grafická data komplexnější než základní grafické objekty a styly implementované v editoru Elastique, a import tohoto formátu je tedy omezen.

Jelikož je vektorová grafika v SVG uložená jako XML data, omezení na podmnožinu SVG znamená omezit import XML pouze na některé elementy a atributy (tabulka 6.1).

Tabulka 6.1: Importovatelná podmnožina SVG

Element	Atributy
`ellipse`	`cx, cy, rx, ry,` *styl*
`circle`	`cx, cy, r,` *styl*
`rect`	`x, y, width, height,` *styl*
`line`	`x1, x2, y1, y2,` *styl*
`path`	`d,` *styl*
`polyline`	`points,` *styl*
`polygon`	`points,` *styl*
`g`	*styl*
styl	`style, fill, stroke, opacity, fill-opacity, stroke-opacity, stroke-width`

I s tímto výběrem lze importovat dosti komplexní obrázky, viz obrázek 6.7. Ostatní

elementy a atributy jsou ignorovány, protože nemají v editoru Elastique odpovídající reprezentaci.

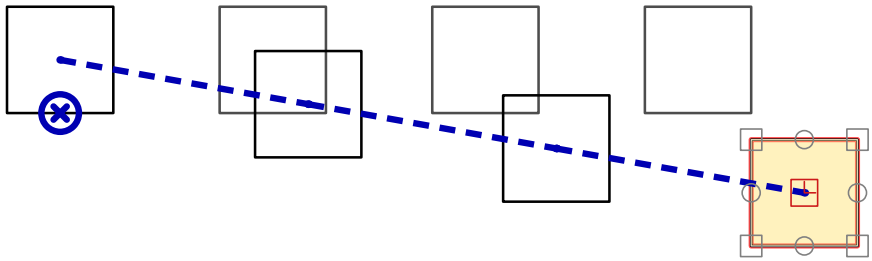
Obrázek 6.7. *Příklad obrázku importovaného z SVG*

Pro import nejsou podporovány všechny atributy objektů v SVG souboru. Importovaný obrázek tedy nemusí vypadat správně, neboť atributy objektů v SVG mohou obsahovat transformační matice, které jsou ignorovány. Příklad srovnání importu SVG a exportu do PNG v editoru Elastique a přímého vykreslení SVG je v uživatelské dokumentaci manuálu[2].

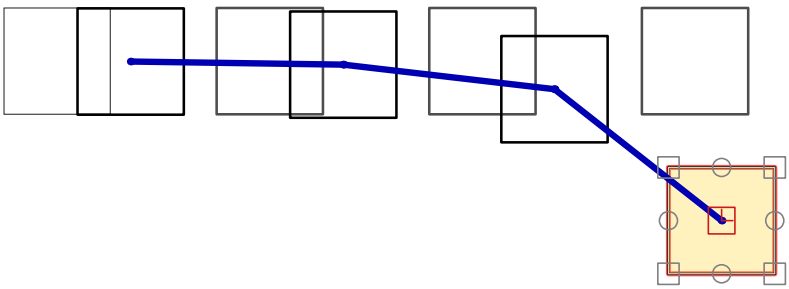
6.6 Vazby

Vazby se vytvářejí volbou typu vazby v menu Bonds. Pak je kliknutím na dva body objektů vytvořena vazba zvoleného typu mezi těmito dvěma objekty. Operací Edit bonds v menu Edit je možné vazby přepojovat mezi objekty — kliknutím a tažením požadovaného konce vazby. Ukotvení větve vazbou nail se provádí dvojklikem pravým tlačítkem myši při zvolené editační operaci (posun, zvětšení a rotace). Příklady obrazců z vazeb jsou na obrázcích 6.8 a 6.9.

[2]Na přiloženém CD v adresáři `guide/formats.html`



Obrázek 6.8. *Ukotvená větev objektů spojených vazbami* loose *při pohybu s jedním z koncových objektů zachovávájí stejné poměry vzdáleností*



Obrázek 6.9. *Pohyb s objekty spojenými vazbou* claw *simuluje pohyb s řetězcem objektů*

7
Možnosti budoucího vývoje

Při vývoji a implementaci editoru se objevila spousta myšlenek, jak editor dále vyvíjet. Implementace dalších funkcí by mohla být značným přínosem pro uživatele pracujícího s editorem, a spousta dalších námětů by mohla být zajímavým tématem jak pro další teoretické, tak praktické práce, ale bohužel většina z nich svým rozsahem dalece přesahuje tuto práci.

Pro pohodlnější práci s editorem by bylo velmi užitečné implementovat následující funkce:

- pořadí vykreslení objektů a operace Bring to front, Send to back,
- vrstvy objektů, práce pouze s vybranou vrstvou, pořadí vrstev,
- seskupování objektů do skupin, vytváření celků pomocí základních objektů,
- efektivnější vnitřní struktury pro výběr a podsvěcování objektů a tím by bylo umožněno editovat libovolně rozsáhlou grafiku,
- konverze objektů mezi sebou, hlavně konverzi libovolného objektu (včetně textu) na křivku, a konverzi čáry na vazbu,
- nastavování středu při zvětšování objektů či obrazců. Obrazce bez ukotvené větve by mohly být zvětšovány vzhledem ke svému středu.
- kreslení od ruky a následná vektorizace kreslených křivek,
- plnou podporu formátu SVG, gradientní přechody, text, skupiny objektů, transformace, animace, popřípadě alespoň ukládat nepodporované části SVG a exportovat je v nezměnené podobě při exportu,
- využít komentáře SVG pro uložení informací o vazbách, formát SVG by pak šel využít pro ukládání grafiky včetně vazeb a mohl by se použít jako nativní formát editoru,
- implementovat více typů vazeb, popřípadě různé chování vazeb (pomocí dědění EEngine).
- použít obecnější výpočetní model vazeb (např. některou z metod Constraint programming) a povolit i vazby tvořící cyklus, vazby uvnitř objektů,

- přidat vazbám priority a definovat všechny objekty jako křivku omezenou vazbami. Zjednodušila by se tak rozšiřitelnost editorů pomocí modulů, neboť všechny kreslící operace by mohly předpokládat editaci křivky,
- navrhnout a implementovat skriptovací jazyk, kterým by bylo možné efektivně kreslit (resp. ovládat editor). Bylo by pak možné kreslit také prostřednictvím skriptování. Některé kreslící postupy by bylo možné zefektivnit pomocí cyklů a smyček. Editor by pak mohl být spouštěn také z příkazové řádky pro automatické vytváření obrázků (např. kreslení grafů atd.). Skript by mohl být také vytvářen při kreslení myší. Vzhledem k tomu, že je v editoru implementována vrstva všech editačních operací samostatně, podpora skriptovacího jazyka spočívá v podstatě pouze v napsání vhodného parseru a vhodného mechanismu volání funktorů,
- přidat funktorům (potomkům EFunctor) schopnost invertovat reprezentovanou operaci, a této schopnosti využít pro vícenásobné Undo/Redo. K jednotlivým okamžikům v kreslení obrázku by se bylo možné vrátit kdykoli jen na základě historie kreslících příkazů. Nabízí se tak i možnost selektivního Undo/Redo, kdy by bylo možné se vrátit v historii kreslení, provést změny, a replikovat zbytek historie znovu,
- skriptovací jazyk by mohl být podmnožinou METAPOSTu. Generovaný skript by pak za předpodkladu definice vhodných maker mohl sloužit přímo jako zdrojový soubor obrázku přeložitelný METAPOSTem. Zároveň by byla umožněna (částečná) konverze mezi SVG a METAPOSTem (bez mezikroku přes PDF a SKetch formáty).

8 Závěr

Ná základě analýzy kreslení a použití základních grafických primitiv jsem navrhnul jednoduchý editor pro editaci a kreslení vektorové grafiky umožňující specifikovat některé poziční vztahy mezi objekty prostřednictvím vazeb.

Řešení se v pruběhu práce postupně vyvíjelo, neboť rysy chování jednotlivých editačních operací nebyly zpočátku do všech detailů známé a jejich v praxi užitečná podoba se mnohdy ukazála až při praktickém testování již implementovaných částí.

Matematický popis výsledného řešení je součástí této práce, stejně jako implementovaný výpočetní model, který sice neposkytuje řešení pro naprosto obecné případy propojení objektů vazbami, ale v běžných případech práce s jednoduchou grafikou postačuje.

Implementace použitých struktur však nezabraňuje doimplementování jiného, obecnějšího výpočetního modelu (např. pomocí metod Constraint Programming). Tato doimplementace by neměla vyžadovat větší změny v ostatních strukturách editoru.

Editor dále umožňuje uložení nakreslené či editované grafiky do vlastního formátu a export do vektorového formátu SVG a bitmapového formátu PNG. Je možné importovat podporovanou podmnožinu grafických objektů a vlastností formátu SVG.

Editor je zkompilován pro platformu Win32 a linux. Portace na jiné cílové platformy vyžaduje pouze přítomnost grafické knihovny GTK a kompilátoru jazyka C++, takže je možná portace na většinu současných platforem.

Literatura

[1] Foley, Van Dam, Feiner, Hughes: *Computer Graphics, Principles and Practice in C*, Addison-Wesley, 1995

[2] Gimp Drawing Toolking: *http://www.gtk.org*

[3] Glade/GTK+ for Windows *http://gladewin32.sourceforge.net*

[4] Jiří Žára, Bedřich Beneš, Jiří Sochor, Petr Felkel: *Moderní počítačová grafika*, Computer Press, 2004

[5] SVG Specification 1.1: *http://www.w3.org/TR/2003/REC-SVG11-20030114*

[6] Tiny XML parser *http://sourceforge.net/projects/tinyxml*

[7] Vektorový editor Inkscape: *http://www.inkscape.org*

[8] W3C, Portable Network Graphics Specification *http://www.w3.org/TR/PNG*

Popis přiloženého CD

Tabulka 8.1: Popis struktury přiloženého CD

`bc_elastique.ps`	text práce ve formátu PostScript,
`bc_elastique.pdf`	text práce ve formátu PDF,
`elastique-bin/win32/`	adresář s binárními soubory programu pro platformu Win32,
`elastique-bin/linux/`	adresář s binárními soubory programu pro platformu linux,
`elastique-bin/user-guide/`	adresář s uživatelskou dokumentací (vstupní bod elastique-bin/user-guide/index.html),
`elastique-bin/programmer-guide/`	adresář s programátorskou dokumentací (vstupní bod elastique-bin/programmer-guide/index.html),
`elastique-bin/examples/`	adresář s příklady,
`elastique-src/extern/`	adresář s převzatými zdrojovými soubory,
`elastique-src/src/`	adresář se zdrojovými soubory,
`elastique-src/targets/`	adresář se soubory potřebnými pro kompilaci,
`gtk/`	adresář s instalačním souborem knihovny GTK pro platformu Win32,
`ctimne`	popis struktury CD a programu v češtině,
`readme`	popis struktury CD v programu angličtině.