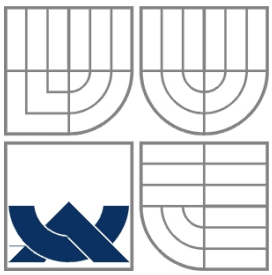
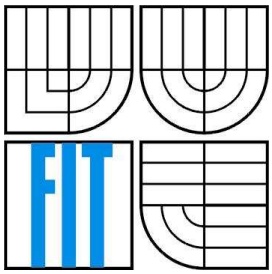


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VEKTOROVÝ EDITOR JAPONSKÝCH ZNAKŮ KANJI

VECTOR EDITOR OF JAPANESE KANJI CHARACTERS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

LINDA PAZDÍRKOVÁ

VEDOUCÍ PRÁCE
SUPERVISOR

ING. MIROSLAV ŠVUB

BRNO 2009

Abstrakt

Práce se zabývá rozborem japonských znaků kanji a tahů, z nichž se skládají. Navrhuje možnost jejich zápisu do struktur pomocí implementovaného vektorového editoru a zkoušení uživatele pomocí překreslování znaku z předlohy.

Abstract

The thesis deals with the analysis of Japanese kanji characters and strokes they consist of. It suggests the possibility of their entry into structures using the implemented vector editor and testing user through drawing characters from the sample.

Klíčová slova

vektorový editor, japonské znaky, kanji, wxWidgets, C++, XML

Keywords

vector editor, Japanese characters, kanji, wxWidgets, C++, XML

Citace

Pazdírková Linda: Vektorový editor japonských znaků kanji, bakalářská práce, Brno, FIT VUT v Brně, 2009

Vektorový editor japonských znaků kanji

Prohlášení

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

……
Linda Pazdírková
20. května 2009

Poděkování

Ráda bych poděkovala Ing. Miroslavu Švubovi za jeho rady, čas a všeobecnou podporu při vytváření této práce.

Obsah

Obsah 1
1 Úvod 2
2 Japonské písmo 3
2.1 Typy japonského písma 3
2.2 Tahy a jejich zápis 3
2.3 Kaligrafie 4
2.3.1 Kaisho 4
2.3.2 Gyousho 4
2.3.3 Sousho 4
3 Návrh implementace 6
3.1 Vektorové písmo 6
3.2 Reprezentace znaku 6
3.3 Reprezentace tahu 6
3.3.1 Vyplnění křivky tahu 7
3.4 Zadávání znaků 7
3.4.1 Snížení počtu řídících bodů 7
3.5 Porovnávání znaků 8
3.5.1 Využití obalového obdélníka 8
4 Použité technologie 10
4.1 Programovací jazyk C++ 10
4.2 wxWidgets 10
4.3 XML (Extensible Markup Language) 10
5 Implementace 11
5.1 Prostředí aplikace 11
5.1.1 Základní okno 11
5.1.2 Okno editačního módu (*editorDialog*) 12
5.1.3 Okno testovacího módu (*testerDialog*) 13
5.2 XML struktura znaku 14
5.3 XML struktura tahu 15
5.4 Určení bodů křivky 16
5.5 Kreslicí okno (třída *SimWindow*) 17
5.6 Struktura *stroke* 18
5.7 Struktura *symbol* 18
6 Závěr 19
Literatura 20
Přílohy 21
Příloha 1.: Tabulka implementovaných tahů 21
Příloha 2.: Obsah CD 23

1 Úvod

Cílem této bakalářské práce bylo vytvořit program, který usnadní učení japonských znaků kanji. Nedílnou součástí tohoto programu je samotný vektorový editor, v němž lze jednotlivé znaky vytvářet. Podobné programy sice jsou k dispozici na internetu, ale převážná většina je řešena rastrově a má značně omezené možnosti.

Součástí této technické zprávy o práci je 6 kapitol. Ve druhé je rozebrána obecná teorie japonského písma. Třetí kapitola se zabývá obecnými řešeními problémů spojených s vytvářením aplikace. Ve čtvrté jsou zmíněny technologie použité pro tvorbu programu. v páté kapitole se pak nachází rozbor konkrétní implementace řešení zmíněných ve třetí kapitole. Závěrečná kapitola shrnuje dosažené výsledky a možnosti dalšího rozšíření.

2 Japonské písmo

2.1 Typy japonského písma

Pro zápis japonštiny se používají tři typy písma - hiragana, katakana a kanji. Hiragana a katakana jsou slabičná písma (jeden znak představuje celou slabiku), která doplňují kanji, a jejich znaky jsou z kanji odvozené. Hiragana se používá pro japonská slova, pro která neexistuje vyjádření v kanji, dále pro předpony, přípony, částice, koncovky a také jako naznačení výslovnosti. Katakanou lze zapsat slova přejatá z cizích jazyků a také typografické zvýraznění slov.

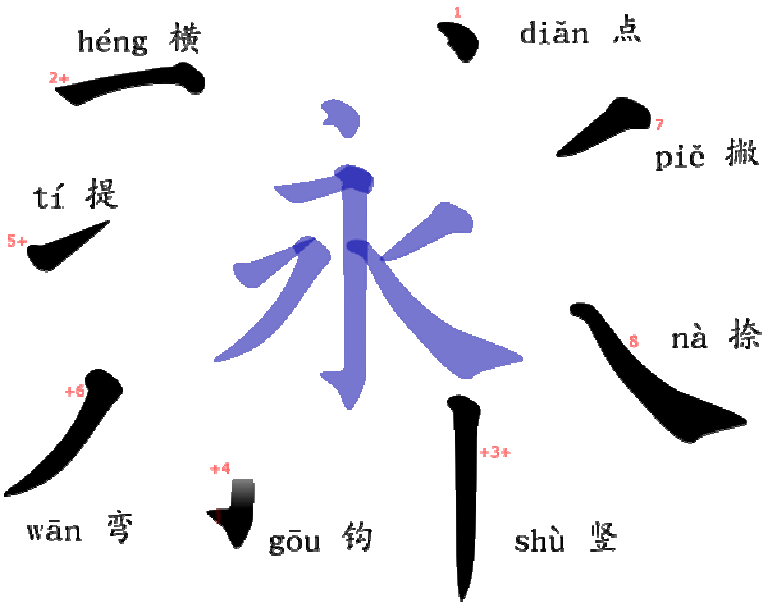
Kandži je odvozené z čínského písma, jednotlivý znak v něm reprezentuje význam, resp. kořen slova. Jak se jednotlivé znaky čtou, je nutné se naučit, není možné výslovnost odvodit od toho, jak znak vypadá. Význam konkrétního znaku však lze určit podle tzv. radikálu, což je část znaku, která celý znak zařazuje do určité významové skupiny. Některé základní znaky jsou samy radikálem a mohou tedy on nich být odvozovány další znaky jejich drobnou změnou nebo přidáním dalších tahů.

Jednotlivé znaky se tradičně píší shora dolů, z nich sestavené "řádky" pak následně zprava doleva. v moderních textech se však používá také stejný směr jako v latince - tedy zleva doprava a řádky shora dolů.

Základní kanji běžné japonštiny obsahuje 1945 znaků, kodifikace pro průmyslové použití (JIS) 6879 znaků. Celkový počet znaků je však neznámý, odhaduje se na několik desítek tisíc. jako otázka, co z ní vypustit. Příliš mnoho podrobností může čtenáře právě tak odradit jako žádné detaily.

2.2 Tahy a jejich zápis

Znaky kanji se skládají z jednotlivých tahů, existuje 37 tahů, z nichž 8 je základních a ostatní jsou složené.



Obr. 2.1: Základní tahy kanji

V každém znaku jsou tahy zapisovány v konkrétním pořadí. To je dáno předem určenými pravidly, které je nutno dodržovat (viz dále). Velikosti jednotlivých tahů jsou pro určítý znak typické. Je také možné, že se v jednom znaku objeví stejný typ tahu v různých velikostech.

Pravidla pro pořadí zápisu jednotlivých tahů jsou následující:

1. Zapisuje se zleva doprava, shora dolů.
2. Horizontální tahy mají přednost před vertikálními.
3. Protínající tahy se zapisují jako poslední.
4. Úhlopříčky zprava doleva mají přednost před úhlopříčkou zleva doprava.
5. Centrální vertikální tah má přednost před vnějšími úhlopříčkami.
6. Vnější tahy mají přednost před vnitřními.
7. Levá úhlopříčka se píše před uzavíracími tahy.
8. Spodní uzavírací tahy se zapisují nakonec.
9. Tečky a drobné znaky se zapisují jako poslední.

2.3 Kaligrafie

Pro zápis znaků kanji existují tři různé styly - Kaisho, Gyousho a Sousho.

2.3.1 Kaisho

Kaisho doslova znamená "správné písmo". Je to styl, kde každý tah je napsán co nejpřesněji tak, aby výsledek byl co nejbližší tištěným znakům. Kaisho je také použitý v tomto editoru.

2.3.2 Gyousho

Gyousho doslova znamená "cestující písmo". Používá se pro běžné ruční psaní, například osobních poznámek. Tahy, které jsou ve stylu Kaisho zapsané zvlášť v Gyousho splývají dohromady.

2.3.3 Sousho

Sousho doslova znamená "travní písmo". Tento styl bývá nejčastější u uměleckých děl, kde štětec zřídka opustí papír a vytváří tak ladné elegatní tvary. v porovnání s tišteným písmem už jsou takto stylizované znaky téměř nerozpoznatelné.

Obrázek 2.2: Znak ve stylu Kaisho

Obrázek 2.3: Znak ve stylu Gyousho

Obrázek 2.4: Znak ve stylu Sousho

Obrázek 2.5: Tištěný znak

3 Návrh implementace

3.1 Vektorové písmo

Jednotlivé znaky vektorových písem jsou (na rozdíl od znaků písem bitových) popsány pomocí matematicky vyjádřených vektorů, které vytvářejí jejich obrysy. Díky tomu nejsou omezeny nějakou konkrétní velikostí a znak o velikosti např. 10 bodů vypadá tvarově úplně stejně jako stejný znak o velikosti 40 bodů. Při vykreslení nebo vytištění znaku však stejně musí být nakonec převedeny na jednotlivé pixely, protože obrazovka i tiskárna pracují s konečným počtem zobrazovacích bodů.

Mezi výhody vektorových písem patří především jejich obecný zápis a z toho plynoucí nezávislost na rozlišení a stálá kvalita. Problémem naopak může být vyplňování obrysů u zařízení s nízkým rozlišením a vyšší časová náročnost vytvoření rastru než u bitmapového písma. Vektorový popis také zahrnuje pouze obrys písma, nikoliv jeho další vlastnosti.

3.2 Reprezentace znaku

Jedním z prvních problémů, nad kterými je třeba se zamyslet je způsob reprezentace jednotlivých znaků v aplikaci. Je potřeba jednak vytvořit pevnou strukturu uloženou v souboru, kde budou uchovávány údaje o konkrétních vzorových znacích použitelných pro zkoušení uživatele, a také dynamickou strukturu, do níž se budou zapisovat znaky, které jsou momentálně vykreslovány.

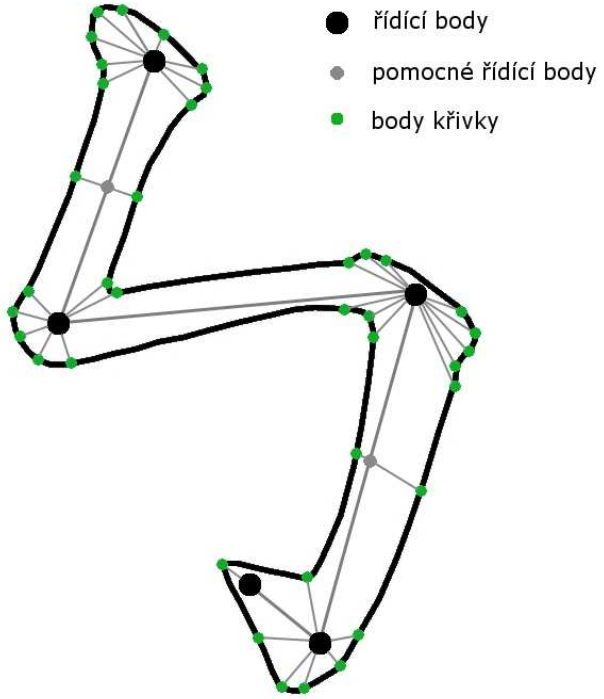
Každý znak se vždy skládá z daného počtu tahů, kde každý je určitého typu, a taktéž záleží na pořadí, v jakém jsou jednotlivé tahy nakresleny. Proto v těchto strukturách musí být zaznamenány i údaje o jednotlivých tazích, kterými je znak tvořen.

3.3 Reprezentace tahu

Také pro tahy, ze kterých se znaky skládají, je nutné vytvořit pevnou strukturu uloženou v souboru, kde budou uchovávány údaje o tom, jak by měl který typ tahu vypadat (vycházející z reálných částí japonských znaků), a také dynamickou strukturu, do níž se budou zapisovat konkrétní vykreslované tahy.

Tyto tahy uživatel zadá vymezením řídících bodů, podle kterých lze zjistit, o jaký typ tahu se jedná. To je možné díky zjištění úhlů, které jednotlivé body svírají a vzdáleností mezi nimi, které jsou právě pro daný typ určující.

Ve chvíli, kdy je podle těchto údajů typ tahu nalezen ve struktuře uložené v souboru, lze z ní načíst informace pro výpočet souřadnic bodů křivky výsledného tahu. Pomocí těchto souřadnic pak může být samotná křivka vykreslena.



Obr. 3.1: Struktura tahu

3.3.1 Vyplnění křivky tahu

Pro zabránění chybám při vyplňování křivky barvou je třeba zvolit barvu odlišnou od vykreslovacích, tou se nakreslí obrys tahu a vyplní se vždy až po zadanou barvu obrysu pomocí semínkového vyplňování. Až poté lze oblast tahu vykreslit finální barvou, pro což je použito vyplňování, které probíhá, dokud se nenarazí na odlišnou barvu.

3.4 Zadávání znaků

Uživatel může řídící body pro jednotlivé tahy zadat dvěma různými způsoby. První možností je na kreslící ploše přímo vybrat konkrétní řídící body a druhou nakreslit křivku, která je přibližným vyjádřením tahu – z ní je potom nutné jednotlivé řídící body určit.

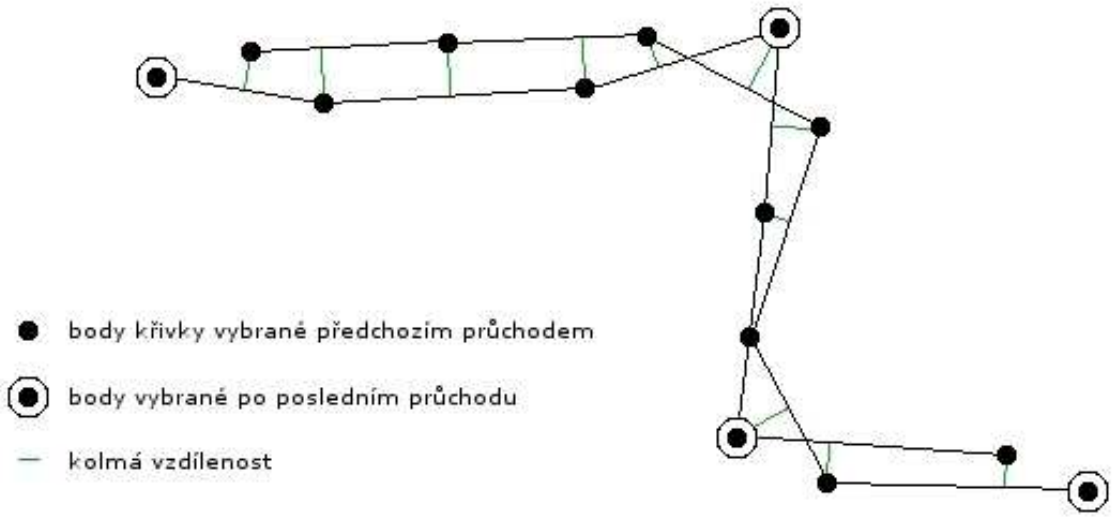
3.4.1 Snížení počtu řídících bodů

Při zadávání tahu uživatelem pomocí křivky narážíme na problém zbytečně vysokého počtu řídících bodů – v podstatě všechny body nakreslené křivky jsou řídícími body. Proto je nutné vyhledat jen ty body, které jsou pro následné určení typu tahu a výpočet bodů výsledné křivky skutečně důležité.

To je možné provést algoritmem, který postupně prochází jednotlivé body každé zadané úsečky a počítá vždy kolmou vzdálenost jednoho bodu nacházejícího se mezi dalšími dvěma. Pokud je tato kolmá vzdálenost nižší než určitá konstanta, lze tento bod vyřadit. Konstantu lze poté postupně zvyšovat a algoritmus opakovat, dokud nejsou odstraněny všechny zbytečné body.

V pseudokódu by tento algoritmus vypadal zhruba takto:

```
násobek = 0
pro i-tý bod ze seznamu zakreslených bodů
    pokud kolmá vzdálenost bodu i+1 od přímky mezi body (i) a (i+2)
       > k * násobek
           přidej bod i+1 do seznamu řídících bodů
pokud počet řídících bodů ≠ počet zakreslených bodů
    zvyš násobek o 1 a vrať seznam řídících bodů od rekurzivně
       zavolaného algoritmu
jinak vrať seznam řídících bodů
```



Obr. 3.2: Výběr řídících bodů

3.5 Porovnávání znaků

Hlavní částí programu má být zkoušení uživatele pomocí překreslování vzorového znaku. Pro zjištění, zda uživatel znak správně zakreslil, je třeba oba znaky porovnat.

Srovnávají se vždy postupně jednotlivé tahy v pořadí, v jakém byly nakresleny, tzn. první vzorový tah s prvním tahem uživatelovým, atd. u každého tahu se zjišťuje se jeho typ (určený podle řídících bodů), poloha a velikost, přičemž pro porovnání polohy a velikosti se využívá výpočtu tzv. obalových obdélníků. Tento obdélník je jednoznačně určen obdélník tvořený maximálními a minimálními souřadnicemi tahu.

3.5.1 Využití obalového obdélníka

Ze souřadnic vzorového i zadaného tahu jsou spočítány obalové obdélníky. Následně se zjistí poměr jejich překryvu ku velikosti plochy n-úhelníka, tvořeného vnějšími hranami obou obalových obdélníků. Pokud je tento poměr vyšší, než zadaná hranice, tahy jsou vyhodnoceny jako shodné.

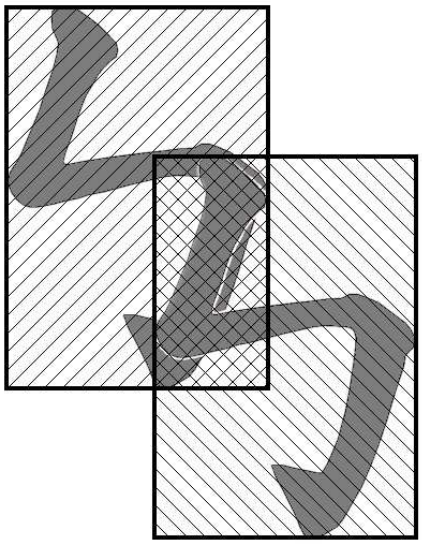
Nastane-li například situace, kdy se porovnávané tahy nacházejí na různých místech, je jejich překryv, a tedy i výsledný poměr, nulový. Kdyby jejich pozice byla shodná, avšak jeden z tahů by byl výrazně větší než druhý, ke shodě opět nedojde, neboť překryv může dosahovat maximálně plochy obdélníka menšího tahu.

Algoritmus porovnávání jednotlivých tahů by pak v pseudokódu vypadal přibližně takto:

```
pokud typ vzorového tahu ≠ typ nakresleného tahu
     vrať false
pokud (souřadnice x nejlevnějšího bodu vzorového tahu
     > souřadnice x nejpravějšího bodu nakresleného tahu) nebo
     (souřadnice x nejpravějšího bodu vzorového tahu
     < souřadnice x nejlevějšího bodu nakresleného tahu)
          vrať false
pokud souřadnice x nejlevnějšího bodu vzorového tahu
     > souřadnice x nejlevnějšího bodu nakresleného tahu
          L = souřadnice x nejlevnějšího bodu vzorového tahu
jinak L = souřadnice x nejlevnějšího bodu nakresleného tahu
pokud souřadnice x nejpravějšího bodu vzorového tahu
     > souřadnice x nejpravějšího bodu nakresleného tahu
          P = souřadnice x nejpravějšího bodu nakresleného tahu
jinak P = souřadnice x nejpravějšího bodu vzorového tahu
...
```

(podobně se pro souřadnice *y* určí proměnné *N* a *D*)

```
overlay = (P - L) * (D - N)
area = (souřadnice x nejpravějšího bodu vzorového tahu
     - souřadnice x nejlevějšího bodu vzorového tahu)
     * (souřadnice y nejspodnějšího bodu vzorového tahu
     - souřadnice y nejhornějšího bodu vzorového tahu)
     + (souřadnice x nejpravějšího bodu nakresleného tahu
     - souřadnice x nejlevějšího bodu nakresleného tahu)
     * (souřadnice y nejspodnějšího bodu nakresleného tahu
     - souřadnice y nejhornějšího bodu nakresleného tahu)
     - overlay
pokud (overlay / area) > k
     vrať true
jinak vrať false
```



Obr. 3.3: Překrytí obalových obdélníků tahů

4 Použité technologie

4.1 Programovací jazyk C++

Při volbě programovacího jazyka pro bakalářskou práci bylo potřeba vzít v úvahu požadavky jako je přenositelnost kódu a efektivita, proto byl vybrán jako nejvhodnější jazyk C++. Tento jazyk podporuje zároveň několik různých paradigmat, včetně objektově orientovaného programování, které lze v této práci ve velké míře využít. Zároveň je dostatečně nízkoúrovňový, aby v něm napsaný kód mohl být dostatečně efektivní, a existuje pro něj řada přenositelných knihoven, které lze při tvorbě editoru využít.

4.2 wxWidgets

Pro implementaci grafického uživatelského rozhraní (GUI) editoru byl vybrán soubor knihoven wxWidgets (dříve nazývaný wxWindows). Tato sada nástrojů pro jazyk C++ umožňuje kód GUI zkompilovat a spustit na několika různých počítačových platformách s minimálními nebo žádnými změnami kódu. Zajišťuje funkčnost pro systémy jako jsou:

- Windows
- Max OS X
- UNIX (X11, GTK+ a Motif)
- OS/2
- AmigaOS a další

Což znamená, že aplikaci napsanou s pomocí wxWidgets bude možné provozovat na všech těchto operačních systémech. Použití těchto nástrojů zároveň také zajišťuje použitelnost programu i na nejnovějších platformách, s podporou nejnovějších technologií. Navíc aplikace vytvořené pomocí wxWidgets také mají přirozený vzhled, zapadajícího do konkrétního operačního systému.

Knihovny wxWidgets poskytuje třídy a funkce pro vytváření pokročilých grafických aplikací, jako jsou například funkce pro práci s okny, menu, nástrojovou lištou, stavovým řádkem, funkce pro kreslení, atd. Obsahuje také základní nástroje pro vytváření platformově nezávislých dialogů.

4.3 XML (Extensible Markup Language)

Jazyk XML byl vybrán jako nejvhodnější pro ukládání perzistentních dat, se kterými editor pracuje a to hned z několika důvodů:

a) XML se stal standartním formátem pro výměnu informací, protože se jedná o jednoduchý otevřený formát nesvázaný s jakoukoliv platformou či technologií a je zpracovatelný libovolným textovým editorem.
b) Jazyk XML má mezinárnodní podporu, text v něm lze psát v různých světových jazycích a znakových sadách a kódování je v něm jednoznačně určeno.
c) Pomocí XML značek lze jednoznačně vyznačit, jaký význam mají jednotlivé části textu, dokumenty tak obsahují více informací než by tomu bylo ve formátech zaměřených na vzhled.

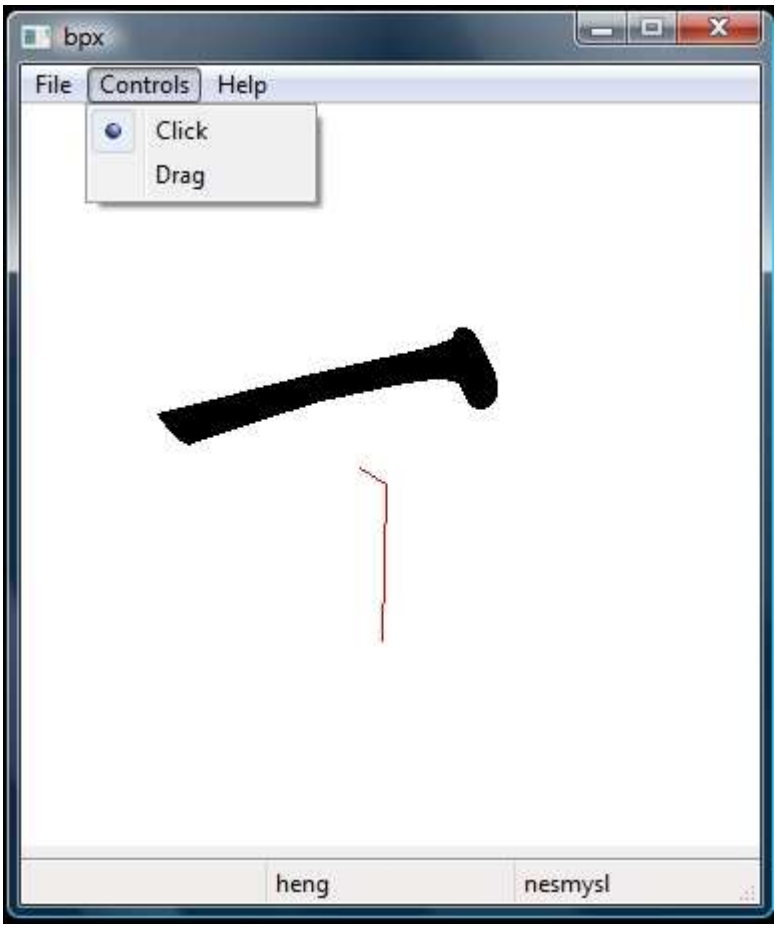
5 Implementace

5.1 Prostředí aplikace

Implementované prostředí editoru japonských znaků kanji se skládá z následujících částí:

5.1.1 Základní okno

Základní okno se zobrazí ihned po spuštění aplikace a obsahuje kreslicí plochu a hlavní nabídku. v okně lze buď tahem myší nebo postupným určením jednotlivých bodů klikem (poslední se určuje klikem pravého tlačítka myši) nakreslit tah. Ten se ihned porovná se záznamem v XML souboru a pokud je nalezen odpovídající vzor, vykreslí se vyplněná křivka daného tahu.



Obr. 5.1.: Základní okno s rozpoznaným a nerozpoznaným tahem

5.1.1.1 Hlavní nabídka

Hlavní nabídka obsahuje tyto položky:

- File
  - Editor – otevře okno editačního módu
  - Tester – otevře okno testovacího módu
  - Clear – vymaže kreslicí plochu základního okna

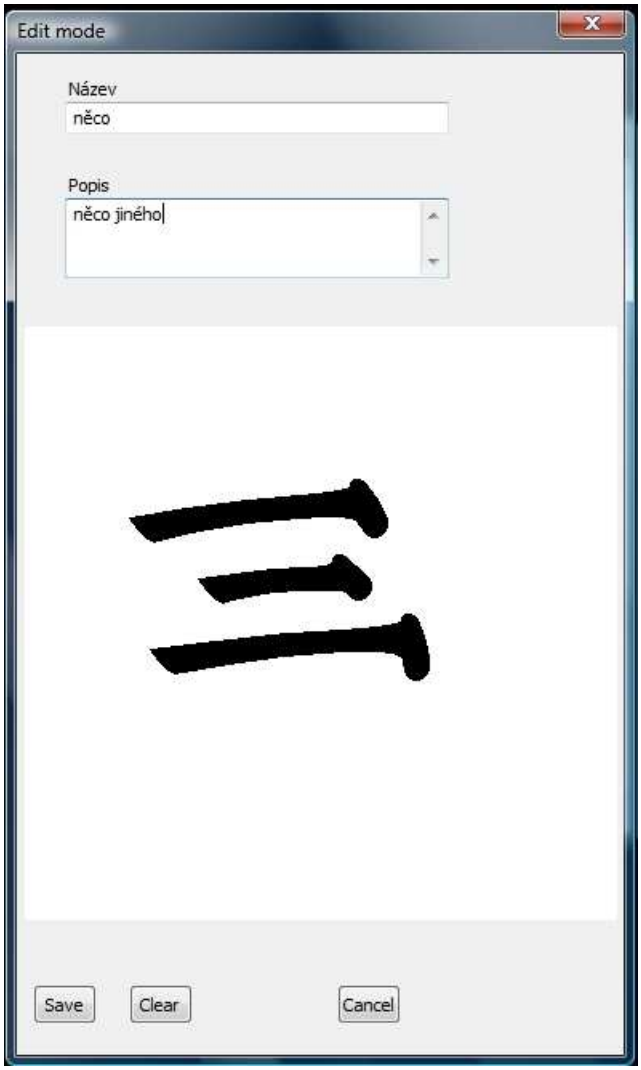
- Exit – ukončí program

- Controls
  - Click – přepne do režimu zadávání řídících bodů klikem myši
  - Drag – přepne do režimu zadávání řídících bodů tahem myši

- Help
  - Help – zobrazí nápovědu k programu
  - About... – zobrazí základní informace o programu

5.1.2 Okno editačního módu (*editorDialog*)

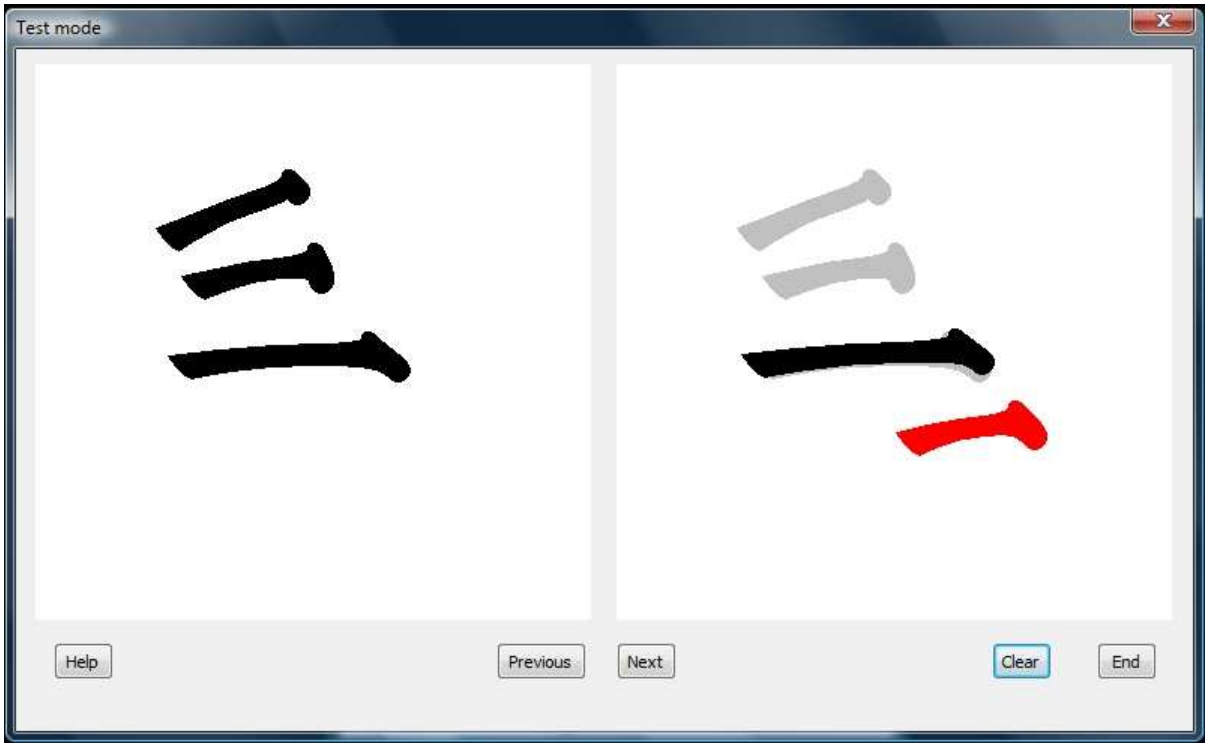
Po vybrání příslušné možnosti v hlavní nabídce se zobrazí okno editačního módu. v něm je možné zapisovat nové znaky, které budou uloženy do XML souboru. u každého nového znaku může uživatel zadat jeho název a popis a následně v kreslicím okně nakreslit samotný znak (podobně jako v základním okně – viz výše).



Obr. 5.2: Okno editačního módu

5.1.3 Okno testovacího módu (*testerDialog*)

Po vybrání příslušné možnosti v hlavní nabídce se zobrazí okno testovacího módu. Zde si uživatel může vyzkoušet kreslení znaků podle vzorů. v levé části okna je vždy zobrazen vzorový znak, do pravé uživatel kreslí. Pomocí tlačítek v dolní části okna lze přecházet na další nebo předchozí vzorové znaky v rámci jedné série (jednoho XML souboru) či si nechat zobrazit nápovědu – vzorový znak v pravé části okna.



Obr. 5.3: Okno testovacího módu (červeně je vyznačený nesprávně zakreslený tah)

5.2 XML struktura znaku

Informace o každém znaku jsou uloženy v XML struktuře tohoto typu:

```
<symbol>
  <name>example</name>
  <description>this is just example</description>
  <strokes>
    <stroke>
      <type>dian</type>
      <numpoints>3</numpoints>
      <points>
        <pt>
          <x>98</x>
          <y>79</y>
        </pt>
        <pt>
          <x>136</x>
          <y>109</y>
        </pt>
        <pt>
          <x>123</x>
          <y>125</y>
        </pt>
      </points>
    </stroke>
    <stroke>
      <type>hg</type>
      <numpoints>3</numpoints>
      <points>
...
```

kde hodnoty jednotlivých uzlů jsou následující:

`name`	jméno znaku
`description`	popis/význam znaku
`strokes`	seznam jednotlivých tahů, kterými je znak tvořen
`type`	typ tahu (odpovídající uzlu `name` v XML struktuře tahu)
`numpoints`	počet řídících bodů tahu
`points`	seznam řídících bodů tahu
x	souřadnice *x* řídícího bodu křivky
`y`	souřadnice *y* řídícího bodu křivky

5.3 XML struktura tahu

Informace o každém tahu jsou uloženy v XML struktuře tohoto typu:

```
<stroke>
  <name>heng</name>
  <numpoints>3</numpoints>
  <main>
    <angle>
      <angp1>
        <bor1>5</bor1>
        <bor2>40</bor2>
      </angp1>
      <angp2>
        <bor1>280</bor1>
        <bor2>345</bor2>
      </angp2>
    </angle>
    <len>
      <len1>
        <b1>2</b1>
        <b2>5</b2>
      </len1>
      <len2>
        <b1>0</b1>
        <b2>0</b2>
      </len2>
    </len>
  </main>
  <curve>
    <c1>
      <pt>1</pt>
      <a>135</a>
      <dist>1</dist>
    </c1>
    <c2>
      <pt>122</pt>
      <a>95</a>
      <dist>0.9</dist>
    </c2>
    ...
```

kde hodnoty jednotlivých uzly jsou následující:

`name`	jméno tahu
`numpoints`	počet řídících bodů
`angle`	seznam rozmezí úhlů mezi jednotlivými řídícími body
`angp`*n*	rozmezí úhlu mezi řídícími body s indexem *n*-1 a *n*
`bor1`	spodní hranice rozmezí úhlu (ve stupních)

`bor2` horní hranice rozmezí úhlu (ve stupních)
`len` seznam rozmezí vzdáleností mezi jednotlivými řídícími body
`len`*n* rozmezí vzdálenosti mezi řídícími body s indexem *n*-1 a *n*
`b1` spodní hranice rozmezi vzdálenosti (viz dále)
`b2` horní hranice rozmezí vzdálenosti (viz dále)
`curve` seznam bodů křivky
`c`*n* bod křivky s indexem *n*-1
`pt` určení bodu, od kterého se daný bod křivky vypočítá
`a` úhel (ve stupních) mezi daným bodem křivky a bodem určeným uzlem `pt`
`dist` vzdálenost (v pixelech, vynásobených konstantou) mezi daným bodem křivky a bodem určeným uzlem `pt`

5.4 Určení bodů křivky

Body, podle nichž se vykresluje výsledná křivka tahu, se počítají buď z jednoho z řídících bodů tahu, nebo z bodu, který se nachází mezi určitými řídícími body. Toto je určeno podle uzlu `pt` XML struktury daného tahu.

Pokud je hodnotou uzlu jednociferné číslo, počítá se bod křivky od řídícího bodu tahu s indexem `pt` – 1. Obsahuje-li však uzel trojciferné číslo, bod křivky se počítá od bodu určeného následovně:

Bod se nachází v 1/*c* vzdálenosti mezi řídícím bodem s indexem *a*-1 a řídícím bodem s indexem *b*-1, kde:

a = 1. cifra hodnoty uzlu `pt`
b = 2. cifra hodnoty uzlu `pt`
c = 3. cifra hodnoty uzlu `pt`

Takže např. z následující XML stuktury:

```
<c2>
  <pt>122</pt>
  <a>95</a>
  <dist>0.9</dist>
</c2>
```

zjistíme, že 2. řídící bod křivky se nachází ve vzdálenosti 0,9k pixelů pod úhlem 95 stupňů od bodu nacházejícího se v ½ vzdálenosti mezi řídícími body s indexy 0 a 1.

5.5 Kreslicí okno (třída *SimWindow*)

Všechna kreslicí okna v aplikaci jsou instance třídy *SimWindow*, definované v souboru *bpxFrm.h*. Právě v této třídě jsou definované hlavní struktury a funkce, nezbytné pro funkčnost celé aplikace. Patří mezi ně tyto:

strField

vektor struktur typu stroke, do kterého se průběžně ukládají všechny nakreslené tahy

rightness

vektor proměnných typu boolean, který určuje, které tahy byly rozpoznány (odpovídají vzoru v XML souboru)

DrawSymbol

funkce, která v testovacím módu vykreslí vzorový znak

TestSymbol

funkce, která v testovacím módu načte vzorový znak z XML souboru, zjistí hranice jeho obalového obdélníka a ke každému tahu, z něhož se daný znak skládá, vypočítá body jeho křivky pomocí bodů načtených z XML souboru (obsahujícího seznam jednotlivých tahů)

FindStroke

funkce, která prohledává seznam tahů v XML souboru a hledá, zda některému z těchto tahů odpovídá nakreslený tah (případně kterému); konkrétně rekurzivně prochází jednotlivé tahy v souboru a ověřuje, zda u daného tahu souhlasí počet řídících bodů a následně zda vzájemná poloha těchto bodů náleží do intervalů úhlů a vzdáleností patřících k tomuto tahu – pokud vše souhlasí, funkce vrátí jméno nalezeného tahu

CheckOverlay

funkce, která ověřuje, zda tah odpovídá příslušné části vzoru – kontroluje typ tahu a dostatečné překrytí obalových odbélníků vzoru a nakresleného tahu

SaveSymbol

funkce, která do XML souboru se seznamem znaků postupně uloží jednotlivé tahy nakresleného znaku

5.6 Struktura *stroke*

Do této struktury se ukládají základní stavební jednotky celé aplikace – jednotlivé tahy. Struktura *stroke* se skládá z následujících prvků:

- typ tahu
- vektor bodů tahu
- vektor řídících bodů křivky tahu
- hraniční body obalového obdélníka tahu

5.7 Struktura *symbol*

Do této struktury se ukládají jednotlivé znaky, které mají být vykresleny jako vzor v testovacím módu. Struktura *symbol* se skládá z následujících prvků:

- jméno znaku
- popis/význam znaku
- vektor tahů, z nichž se znak skládá

6 Závěr

Aplikace vytvořená v průběhu této bakalářské práce umožňuje zápis japonských znaků kanji do efektivní struktury a následné zkoušení uživatele porovnáváním jím nakreslených znaků s vytvořenými vzory.

Při implementaci bylo zjištěno, že zápis tahem není zcela přesný, zdokonalení algoritmu pro výběr řídících bodů by však vyžadovalo značné množství času navíc. Dále je nutné doplnit další tahy do souboru se seznamem tahů.

Program je možné vylepšit rozdělením znaků do tematických kategorií s možností odděleného zkoušení znaků z těchto jednotlivých kategorií. Dále by byla možná integrace slovníku spojená s rozpoznáváním uživatelem zadaného znaku.

Literatura

[1] Smart, J., Roebling, R., Zeitlin, V., Dunn, R.: wxWidgets 2.8.10: a portable C++ and Python GUI toolkit [online]. c1992-2006, aktualizováno 2009-02-01. Dostupné na URL: <http://docs.wxwidgets.org/stable/>

[2] Smart, J., Hock, K.: *Cross-Platform GUI Programming with wxWidgets*. Pearson Education, Inc., 2005. ISBN 0-13-147381-6

[3] Root.cz [online]. c1998-2009, aktualizováno 2009-05-18. Dostupné na URL: <http://www.root.cz/>

[4] Japanese calligraphy, Japanese name translation, Kanji name translation, japanese symbols for names [online]. c2009, aktualizováno 2009-05-18. Dostupné na URL: <http://www.japanese-name-translation.com/>

[5] Murphy, D. and team: Yamasa Online Japanese Dictionary [online]. Dostupné na URL: <http://www.yamasa.org/ocjs/kanjijiten/english/>

[6] Fonty [online]. c2004, aktualizováno 2004. Dostupné na URL: <http://www.magtypo.cz/>

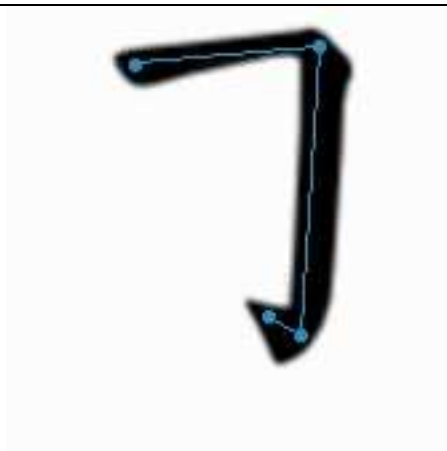
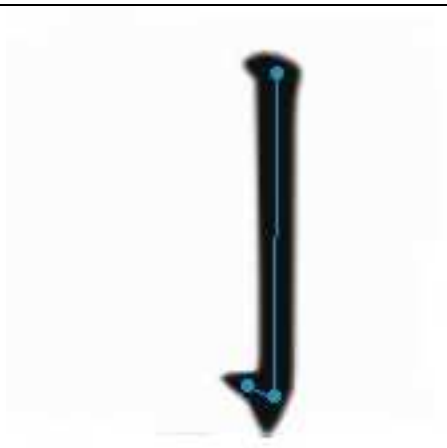
[7] Wikipedia, the free encyclopedia [online]. aktualizováno 2009-05-18. Dostupné na URL: <http://en.wikipedia.org/>

[8] Wikipedie, otevřená encyclopedie [online]. aktualizováno 2009-05-18. Dostupné na URL: <http://cs.wikipedia.org/>

Přílohy

Příloha 1.: Tabulka implementovaných tahů

Příloha 2.: Obsah CD

binary – adresář obsahující spustitelné binární soubory
doc – adresář obsahující tuto technickou zprávu ve formátu DOC (MS Word) a PDF
poster – adresář obsahující plakát prezentující tuto práci
source – adresář obsahující zdrojové kódy programu