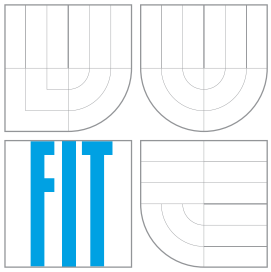
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



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

PAVEL ŽIŽKA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MIROSLAV ŠVUB

BRNO 2008

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2007/2008

Zadání bakalářské práce

Řešitel: **Žižka Pavel**
Obor: Informační technologie
Téma: **Vektorový editor japonských znaků kanji**
Kategorie: Počítačová grafika

Pokyny:

1. Prostudujte základy problematiky vektorového písma.
2. Seznamte se s japonskými kanji znaky a prostudujte způsob jejich dekompozice na základní tahy.
3. Navrhněte způsob dekompozice znaků a editace elementárních tahů.
4. Implementujte vektorový editor základních tahů.
5. Navrhněte notaci pro zápis znaku pomocí tahů a implementujte cca 10 vybraných znaků.
6. Zhodnoťte dosažené výsledky a diskutujte možnosti budoucího vývoje.
7. Vytvořte stručný plakát prezentující vaši bakalářskou práci, její cíle a výsledky.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

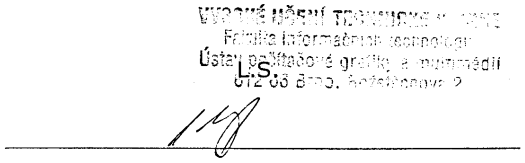
- Splnění prvních třech bodů zadaní.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Švub Miroslav, Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2007
Datum odevzdání: 14. května 2008



doc. Dr. Ing. Pavel Zemčík
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Pavel Žižka**
Id studenta: 79266
Bytem: Sosnová 363, 739 61 Třinec
Narozen: 10. 06. 1986, Třinec
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP: Vektorový editor japonských znaků kanji
Vedoucí/školitel VŠKP: Švub Miroslav, Ing.
Ústav: Ústav počítačové grafiky a multimédií
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:
tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☐ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

Brně dne: .....

.....	.....
Nabyvatel	Autor

Abstrakt

Tato práce se zabývá rozpoznáváním japonských znaků Kanji. Práce zahrnuje problematiku tvorby japonských tahů, znaků a jejich následné rozpoznávání. Vytvořená aplikace slouží k výuce japonských znaků a také k jejich překladu po nakreslení uživatelem. Pro překlad nevyžaduje žádné znalosti japonštiny.

Klíčová slova

Kanji, japonština, rozpoznávání tahů myši

Abstract

This work considers recognizing of Japanese Kanji characters. It discusses the question of creating strokes, characters and their recognition. Created application should serve for Kanji translation and learning. It doesn´t require any knowledge of Japanese.

Keywords

Kanji, Japanese, mouse gestures recognition

Citace

Pavel Žižka: Vektorový editor japonských znaků Kanji, bakalářská práce, Brno, FIT VUT v Brně, 2008

Vektorový editor japonských znaků Kanji

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Miroslava Švuba

.....
Pavel Žižka
14. května 2008

Poděkování

Rád bych poděkoval svému vedoucímu, Ing. Miroslavu Švubovi, za obrovskou ochotu a odbornou pomoc při tvorbě této práce.

Obsah

Kapitola 1

Úvod

1.1 Japonské písmo

Japonské písmo je jedno z nejobtížnějších na světě. Samotná japonština nepatří do žádné jazykové skupiny, má pouze určitou syntaktickou podobnost s korejštinou. Japonské písmo je převzato původně z čínských znaků. Japonci používají dvojího písma pro psaní:

Kana – tyto znaky vyjadřují zvuky, podobně jako naše abeceda. Rozdílem je, že zatímco u nás je základní fonetickou částicí hláska, v japonštině je jí slabika.

Kanji – znak vyjadřuje pojem. Znaky Kanji se často používají ve spojení se znaky Kana, přičemž Kanji vyjadřují podstatná jména, přídavná jména, zájmena, číslovky a slovesa, zatímco Kana se používá pro skloňování a časování.

Každý znak se skládá ze dvou částí: z radikálu a fonetika. Radikál znamená význam slova a fonetikum naznačuje výslovnost. Přesto u většiny znaků není možné odvodit výslovnost pouze z fonetika (také proto, že je velmi obtížné oddělit radikál a fonetikum), proto je nutné znát výslovnost pro každý znak. Znaky mají navíc dvojí čtení:

Onyomi (sinojaponské čtení) – výraz onyomi znamená doslova *čtení podle zvuku*. Jedná se o čtení znaků podobné čínskému čtení, používá se až při složeninách několika znaků, samo o sobě většinou nemá jednoznačný význam.

Kunyomi (japonské čtení) – není možné je odvodit ze znaku, je potřeba je znát.

Samotní Japonci mají se svým písmem řadu starostí. Často se stane, že nejsou schopni si přečíst poezii nebo dokonce noviny, protože se tam vyskytuje sled znaků, které neznají. Proto bylo japonskou vládou určeno 1 850 běžně užívaných znaků (tzv. *Tōyō Kanji*), po jejichž zvládnutí by mělo být možné přečíst všechny běžné texty (např. noviny, technické zprávy, ...)

Přestože se v dnešní době mnoho lidí zajímá o východní země, kulturu a filosofii, není k dispozici mnoho kvalitních programů pro výuku a překlad japonských znaků. Tato bakalářská práce se proto pokouší tento stav změnit.

1.2 Přehled kapitol

- Kapitola 1 by měla seznámit s tématem práce.
- Kapitola 2 pojednává o použitých znalostech a nástrojích.
- Kapitola 3 popisuje dostupné projekty týkající se japonštiny a proč tato práce vlastně vznikla.
- Kapitola 4 se zabývá postupy, které byly využity při návrhu aplikace. Popisuje rozpoznávání jednotlivých tahů, rozpoznávání a ukládání znaků.
- Kapitola 5 stručně popisuje knihovny, které byly při implementaci použity a aplikaci samotnou.
- Kapitola 4 shrnuje výsledky práce, názory uživatelů a naznačuje možná rozšíření.

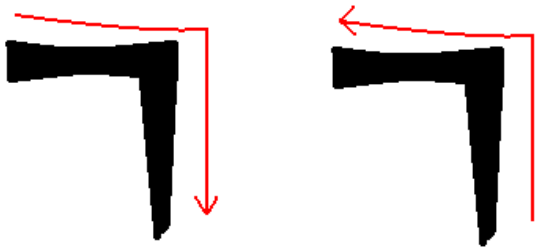
Kapitola 2

Teorie

Před návrhem aplikace bylo potřeba prostudovat způsob tvorby japonských znaků. Kromě toho bylo potřeba zvážit prostředky pro vizualizaci tahů a postupy pro ukládání znaků.

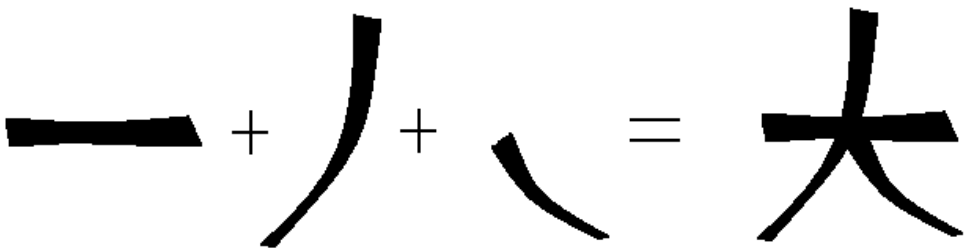
2.1 Teorie japonských znaků Kanji

Kanji se skládají z tahů. Základních tahů je 6, rozšířených (které jsou složeninami základních tahů) je 33 (čísla se mohou různit, jelikož neexistuje přesná specifikace). Kombinací těchto základních a rozšířených tahů vznikají znaky. Tahy mají předepsaný směr kreslení, jak znázorňuje obrázek 2.1.



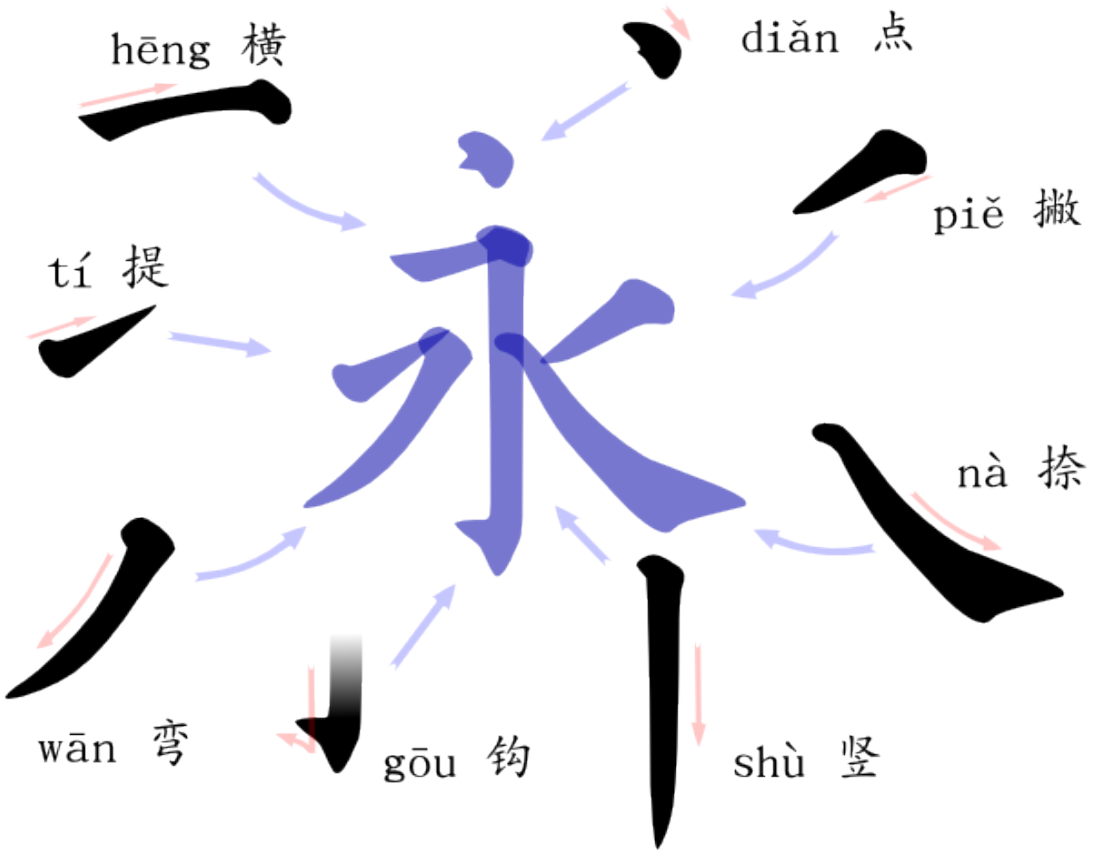
Obrázek 2.1: Na obrázku je vlevo správný směr tahu *Hēng-Zhé*, vpravo chybný směr.

Také pořadí tahů ve znaku je přesně definováno, jak ukazuje obrázek 2.2.



Obrázek 2.2: Skládání znaku *Velký*.

2.1.1 Základní tahy



Obrázek 2.3: Základní tahy.

Znalost základních tahů je zásadní jak pro výuku, tak pro rozpoznávání japonských znaků. Základních tahů je šest, na obrázku 2.3 jsou to *Tí, Hēng, Diǎn, Piě, Nà* a *Shù*. Jsou velice jednoduché a netrvá příliš dlouho si je zapamatovat. Navíc jsou od sebe velmi dobře rozlišitelné (snad pouze *Hēng* a *Tí* by se mohly někdy zaměnit). Zbývající dva tahy na obrázku (*Wān, Gōu*) se používají pouze ve složených tazích [4].

Tí – v překladu znamená *vzestup*. Tah se vede pod mírným sklonem nahoru, zleva doprava.

Hēng – název *vodorovně* napovídá způsob tvorby tahu, vodorovně zleva doprava.

Diǎn – v překladu *tečka*. Velmi krátký tah, vede se směrem dolů, většinou zleva doprava, někdy se však může vést i zprava doleva.

Piě – znamená *odhodit pryč*. Tah je veden šikmo dolů, zprava doleva.

Nà – překládá se jako *tlačit dolů*. Vede se směrem šikmo dolů, zleva doprava, na konci se tah většinou rozšiřuje.

Shù – název *svisle* opět naznačuje směr vedení. Tah je veden svisle shora dolů.

2.2 Zobrazování křivek

Pro zobrazování křivek se používají různé algoritmy. Dají se rozlišit do dvou základních skupin:

Interpolační křivky – křivka musí procházet danými řídícími body. Používají se interpolace Lagrangeovým polynomem a C1 a C2 kubické interpolace.

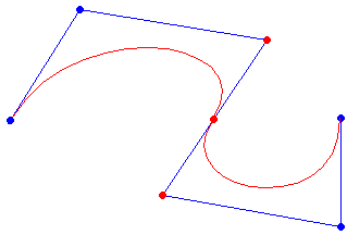
Aproximační křivky – tvar křivky je řízen řídícími body, nemusí však jimi procházet. Aproximačními křivkami jsou například Bézierovy křivky (buďto racionální nebo neracionální), anebo Coonsovy kubiky či jejich zobecnění (B-spline křivky, NURBS...).

2.2.1 Bézierovy křivky

Program využívá pro vykreslování křivek kubické Bézierovy křivky, které jsou implementovány v toolkitu wxWidgets. Tvar křivek je určen řídícími body. Počátečním a koncovým bodem křivka prochází, body mezi nimi určují tvar. Obecně lze Bézierovu křivku stupně n (která je určena $n+1$ řídícími body) popsat pomocí rovnice 2.1 [2].

$$B_{(t)} = \sum_{i=1}^{n} \binom{n}{i} P_{(i)} (1-t)^{n-i} t^i \tag{2.1}$$

Jelikož jsou křivky vyšších stupňů výpočetně náročné, využívají se v praxi většinou křivky čtvrtého stupně (kubické). Pokud je zapotřebí vykreslit složitější křivku, vytvoří se navazováním křivek nižších stupňů. Tímto způsobem může vzniknout křivka, která není hladká. Hladkosti se dosahuje tím, že body v okolí navazování leží v přímce, jak ukazuje obrázek 2.4 [10].



Obrázek 2.4: Obrázek ukazuje navazování dvou Bézierových křivek 3.stupně (jsou zadány čtyřmi řídícími body). Červeně označené body musí ležet v jedné rovině, aby bylo napojení dvou kubických splinů hladké.

Bézierovy křivky se pro své vlastnosti velice často využívají v počítačové grafice (např. v programech *Adobe Illustrator, Inkscape...*).

- Křivka leží v *konvexní obálce* svých řídících bodů.
- Kontrola tvaru křivky je pomocí řídících bodů jednoduchá a intuitivní.

2.3 Formát XML

Formát XML je rozšiřitelný značkovací jazyk, který je navržen jako univerzální formát pro přenos informací mezi různými platformami a programy [5]. Jazyk XML byl zvolen pro uložení jednotlivých znaků pro své nesporné výhody, které skýtá:

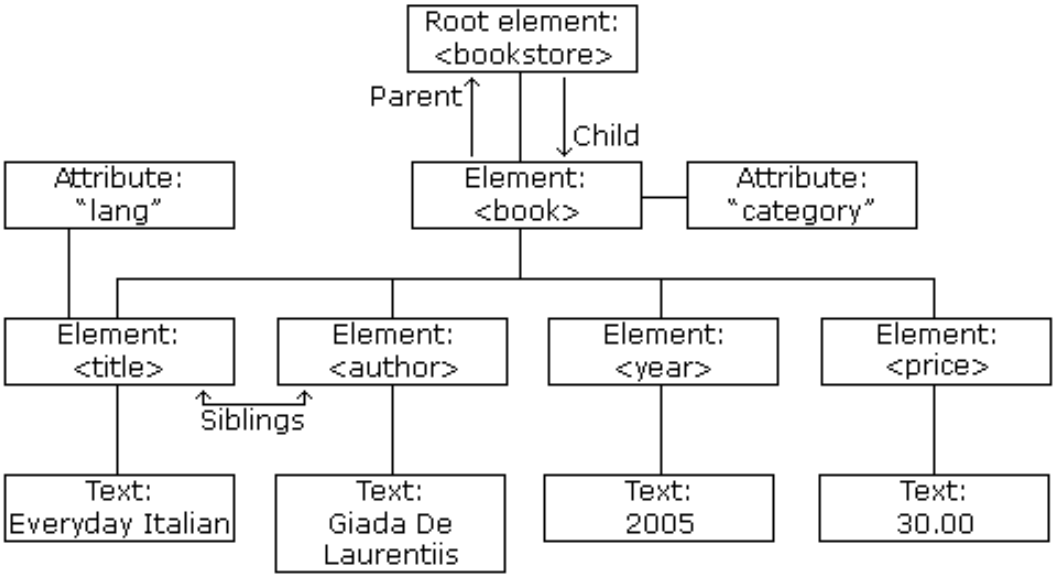
- Informace jsou zaznamenány v textové podobě, což umožňuje nezávislost na použité architektuře.
- Dokáže reprezentovat typické struktury pro počítače (seznamy, stromy...).
- Je založen na mezinárodních standardech.
- Často používaný formát, existuje pro něj spousta parserů pro nepřeberné množství programovacích jazyků.

2.3.1 Definice vlastních tagů

Jazyk XML je navržen pro přenos informací, proto má programátor obrovskou volnost při tvorbě vlastních značek. Sémantiku těmto značkám určuje až to, jak je program interpretuje. Samotné informace se uchovávají buďto přímo ve značkách, jejich atributech nebo textu mezi počáteční a odpovídající koncovou značkou. Vzhledem tomu, že XML dokument (obr. 2.5) vytváří stromovou strukturu (obr. 2.6), je velice vhodný pro uchovávání dat objektového charakteru [1].

```
<?xml version="1.0" encoding="windows-1250"?>
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="CHILDREN">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="WEB">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

Obrázek 2.5: Obrázek znázorňuje ukázkový XML soubor (příklad byl převzat ze stránek *www.w3schools.com/xml/xml_tree.asp*).



Obrázek 2.6: Obrázek znázorňuje strukturu daného XML souboru (2.5) (příklad byl převzat ze stránek *www.w3schools.com/xml/xml_tree.asp*).

Tato technologie má jistě i své nevýhody, mezi které patří jistá redundance dat vzhledem k binárním formátům, ovšem výhody pro účel mé aplikace jasně převažují.

Kapitola 3

Motivace

3.1 Co by měl program umět

Cílem práce bylo vytvořit aplikaci pro rozpoznávání japonských znaků. Využití této aplikace leží v několika oblastech, například pro výuku japonských znaků nebo pro překlad mezi japonštinou a jinými jazyky.

Motivací byl nedostatek kvalitních, volně šiřitelných aplikací pro překlad a výuku japonštiny. Program by měl splňovat dvě hlavní funkce:

Překlad japonských znaků do jiných jazyků — Většinou není problém nalézt odpovídající japonský znak k určitému výrazu v češtině. Ovšem pokud má člověk k dispozici pouze znak, je problém zjistit jeho význam bez znalosti tvorby japonských znaků, odhadu počtu tahů či typu radikálů.

Výuka kresby japonských znaků — Pro tento účel je k dispozici velice málo programů, které navíc nejsou příliš kvalitní. Hlavním problémem je nutnost nakreslit znak na přesně stejné pozici a stejné velikosti, v jaké je znak uložen v databázi, což činí výuku velmi nesnadnou.

3.1.1 Přehled nejvýznamnějších projektů zabývajících se japonštinou

Wakan

Jedná se o freeware překladový slovník pro studenty čínštiny a japonštiny. Poskytuje rozsáhlou databázi znaků, významů, slovních spojení a vazeb. Dále obsahuje propracované prostředí pro vyhledávání znaků, které ovšem předpokládá základní znalosti čínského a japonského písma. Hledání spočívá v nakreslení znaků ze seznamu přibližně 300 radikálů, což činí vyhledávání velmi nesnadným pro člověka, který nedokáže odhadnout, ze kterých radikálů se daný znak skládá. K nevýhodám patří, že je tento program určen pouze pro operační systém Windows, kde je navíc vyžadována instalace podpory pro čínské a japonské znaky [8].

Kanji Alive

Projekt Chicagské Univerzity. Není slovníkem, je zaměřen pouze na výuku. Nabízí více než tisíc znaků k výuce, znázorňuje tvorbu znaku, čtení, psaní a nabízí možnost poslechu

výslovnosti [7]. Zobrazuje také proměny tahů, jak se postupně měnily průběhem času. Kladem je, že není třeba mít nainstalovanou podporu pro čínské a japonské znaky, bohužel tento program funguje pouze pod systémy Mac OS X a Windows.

Yokozuna

Opět projekt více orientovaný na výuku než na překlad a rozpoznávání znaků. Jako jediný v sobě zahrnuje kreslení znaků pomocí myši, avšak pouze pro výuku, nikoliv pro vyhledávání. Dalším problémem je nutnost nakreslit tahy velmi přesně (takřka nulová tolerance), což uživatele od výuky spíše odradí. Implementace je pouze pro operační systémem Windows.

The Kanji Site

Online výukový program, který lze nalézt na *www.kanjisite.com*. Obsahuje rozsáhlou databázi japonských znaků, o který poskytuje vyčerpávající informace. Výhodou je, že znaky jsou uloženy ve formě obrázků, není tedy třeba mít nainstalovanou podporu pro čínské a japonské znaky.

3.1.2 Shrnutí

Kromě výše zmíněných programů je na internetu k dispozici velké množství dalších aplikací pro výuku japonštiny, většinou se jedná o tzv. *Kanji cards*. Tyto kartičky obsahují kresbu znaku, postup jeho kreslení, druhy čtení a další informace, které mohou být žáku užitečné. Při výuce se tyto kartičky ve školách většinou vytisknou a poté slouží pro jednodušší zapamatování znaků.Veškeré dostupné programy pro výuku či překlad japonštiny trpí několika neduhy:

- Zaměření pouze na jednu či dvě platformy.
- Nutnost instalace podpůrných knihoven (*například podpora pro zobrazování znaků*).
- Dosud *žádná* podpora vyhledávání pomocí kreslení znaků.

Zvláště nulová podpora vyhledávání pomocí nakreslení znaků je zarážející. Překladové slovníky jsou sice řazeny logicky podle typu radikálů a počtu tahů, ovšem toto není člověku příliš platné, pokud nemá alespoň základní znalost japonštiny. Proto jsem se rozhodl naprogramovat v rámci bakalářské práce aplikaci, která by byla přeložitelná pod většinou používaných operačních systémů a podporovala kreslení znaků bez znalosti skladby tahů a radikálů.

Kapitola 4

Návrh aplikace

4.1 Struktura aplikace

Program by měl plnit dvě úlohy.

1. Měl by sloužit k výuce japonských znaků.
2. Měl by být schopen rozpoznat znak, který uživatel nakreslí. Jelikož se nedá u uživatele předpokládat znalost japonštiny, měl by být program schopen rozpoznat i znaky, jejichž tahy byly nakresleny v nesprávném pořadí.

Aplikace by tedy měla implementovat dva na sobě nezávislé módy.

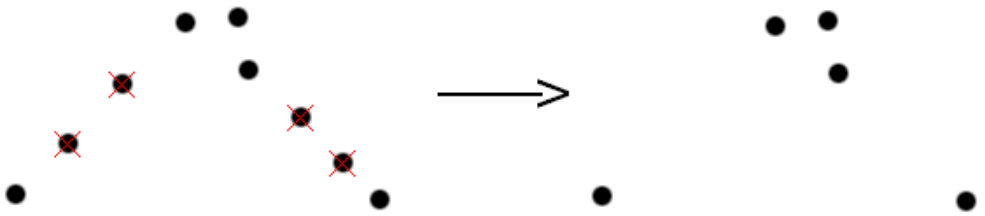
- Mód pro výuku znaků – umožňuje výběr znaku k naučení z několika kategorií. Po vybrání znaku se tento zobrazí na kreslící ploše. Jelikož mají tahy své pevně definované pořadí, je vždy tah, který je *na řadě*, barevně odlišen. Jakmile jej uživatel správně nakreslí, zvýrazní se následující tah, dokud není znak dokončen.
- Mód pro překlad – v tomto módu se čeká, až uživatel nakreslí znak, který by chtěl přeložit. Pro lepší kontrolu nad tvorbou se znaku se po každém nakresleném tahu zobrazí uživateli jak byl tah rozpoznán, aby nedošlo k záměně některých podobných tahů. Znak je poté vyhledán v databázi a uživateli se zobrazí buďto zpráva o nalezení spolu s překladem nebo zpráva o nenalezení znaku. Tento mód slouží zároveň pro ukládání nových tahů. Uživatel může tah buďto přímo uložit, anebo se pokusit znak vyhledat v databázi. Pokud se v databázi znak nenalézá, program sám nabídne možnost uložení.

Ačkoliv jsou oba módy odlišné, používají mnoho společných funkcí a principů. V obou případech je základem kreslení tahu na kreslící plochu aplikace. Ta si pamatuje body, které uživatel při kreslení prošel a poté je předá třídě pro zpracování tahů. Před každým rozpoznáváním tahů je potřeba nejdříve je zpracovat do vhodné podoby, jelikož zaznamenaných bodů je velmi mnoho. S takovým množstvím bodů by bylo velice obtížné pracovat, proto je potřeba je pomocí několika funkcí převést na několik málo bodů, které je možno jednoznačně přiřadit k určitým tahům. Pro vyhlazování křivek, zanedbání nepodstatných bodů a nalezení významných bodů, kde dochází k navazování jednoduchých tahů, využívá aplikace dvou metod, které jsou popsány níže.

4.2 Zpracování tahů

4.2.1 Vyloučení bodů které leží v přímce

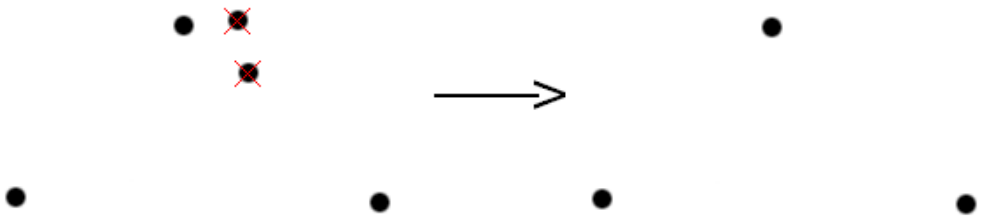
Při průchodu touto funkcí se postupně počítají úhly, které spolu navzájem svírají po sobě jdoucí trojice bodů. Poté jsou odstraňovány vektory které spolu svírají úhel menší než 40°, jak znázorňuje obrázek 4.1. Tím je získána křivka s poměrně malým počtem bodů, přesto v ní stále zůstávají nadbytečné vrcholy, způsobené nedokonalostí použitého periferního zařízení (myš, tablet,...) nebo třesením ruky uživatele. Tyto chybové vrcholy většinou leží blízko sebe, proto je možné je odstranit diskretizací přímky.



Obrázek 4.1: Odstranění bodů ležících v přímce.

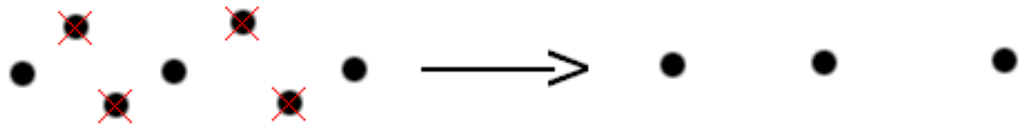
4.2.2 Diskretizace přímky

Diskretizace přímky vymazává šum na křivce, který je způsoben použitým zařízením pro kreslení či třesem ruky zadavatele. Při prvním průchodu si zjistí celkovou délku křivky a podle toho určí práh diskretizace. Tento práh znamená nejmenší povolenou vzdálenost mezi dvěma body. Při druhém průchodu se tedy odstraňují body, které jsou u sebe blíže než je daný práh. Příkladem je obrázek 4.2.



Obrázek 4.2: Diskretizace přímky odstraní šum způsobený nepřesnostmi při tvorbě tahu.

Po průchodu těmito funkcemi získáme relativně malý seznam několika bodů, ve kterých mění křivka zásadně svůj tvar. Je důležité zachovat pořadí těchto funkcí. Pokud by nejdříve proběhla diskretizace, mohlo by se stát, že některý z důležitých bodů by byl vymazán, jak naznačuje obrázek 4.3.



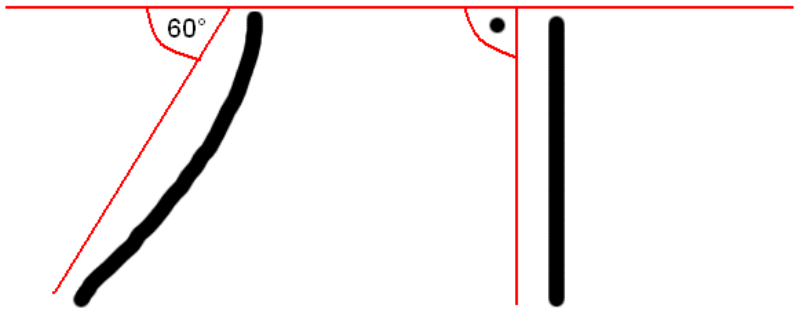
Obrázek 4.3: Odstranění významných bodů vlivem diskretizace před odstraněním bodů ležících v přímce.

4.3 Rozpoznání tahů

Tahů je v Kanji přibližně 40, to však zahrnuje i složené tahy.

4.3.1 Základní tahy

Základních tahů je 6. Jedná se o rovné nebo jen mírně zakřivené křivky. Pokud je nakreslený tah jedním ze základních, měl by být po průchodu výše zmíněnými funkcemi tvořen pouze dvěmi body. Pro jejich rozpoznání stačí znát jejich počáteční a koncový bod a úhel, který svírají s vodorovnou osou (obr. 4.4).

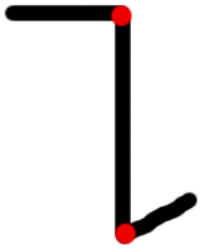


Obrázek 4.4: Obrázek ukazuje způsob rozpoznávání základních tahů podle úhlu, který svírají s vodorovnou osou.

4.3.2 Složené tahy

Tyto tahy jsou složeny ze základních tahů. Při navazování základních tahů (které jsou pouze mírně zakřiveny) vznikají ostré přechody, které je možno využít k rozpoznaní znaků (obr. 4.5). Pro rozpoznání znaků je tedy klíčové najít tyto významné body na křivce.

Před rozpoznáváním je tedy seznam bodů zredukován pouze na význačné body, což významně usnadňuje rozpoznání jednotlivých tahů.



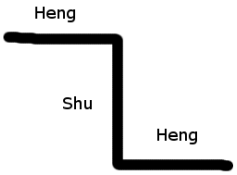
Obrázek 4.5: Složený tah *Hēng-Zhé-Gou*. Červené body vyznačují místa spojení jednotlivých základních tahů.

Při rozpoznávání se postupuje následovně:

1. Pokud se jedná o tah, který se skládá pouze ze dvou bodů, předpokládá se, že je to základní tah a program se jej pokusí přiřadit k některému ze základních tahů.
2. Při rozpoznávání základních tahů se spočítá jejich úhel vzhledem k vodorovné ose a poté se pomocí tabulky 4.1 rozhodne o příslušnosti tahu. Je samozřejmě nutné určité rozmezí platnosti, neboť není možné vést tah přesně pod daným úhlem.
3. Pokud se jedná o složený tah, rozpoznají se tahy mezi dvojicemi bodů a proběhne rozhodnutí o jejich platnosti jak naznačuje obrázek 4.6.
4. Pokud jsou všechny základní tahy ve složeném tahů platné, pokusí se program vyhledat v databázi danou složeninu.

Tah	počáteční úhel	koncový úhel
Shù	85°	95°
Hēng	-5°	5°
Piě	40°	50°
Nà	130°	140°
Diǎn	120°	130°
Tí	10°	20°

Tabulka 4.1: Tabulka rozmezí úhlů pro základní tahy.



Obrázek 4.6: Rozpoznání jednotlivých základních tahů.

4.4 Ukládání znaků

Ukládání znaků by mělo splňovat tyto požadavky:

1. Umožňovat zpětné vyhledávání nezávisle na rozměrech jednotlivých tahů.
2. Rozlišovat kategorie znaků.
3. Ukládat název znaku.
4. Umožňovat rekonstrukci znaku pro jeho vykreslení.

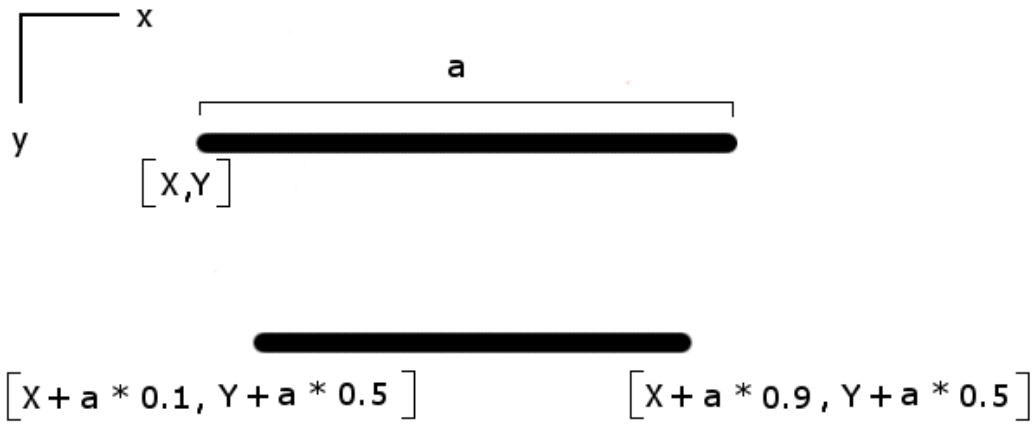
Uživatel nikdy nezadá znak naprosto přesně, dokonce se nelze spolehnout ani na stejnou velikost tahů. Proto nemohou být v databázi uchovány souřadnice vrcholů tak, jak byly zadány při vložení znaku.

Základním předpokladem rozpoznávání je, že se uživatel bude snažit, aby výsledný znak připomínal předlohu co nejlépe. Za toho předpokladu je možno se spolehnout na určité zachování měřítka tahů ve znaku. Proto jsem se rozhodl vyřešit ukládání tahů ve znaku relativně vzhledem k velikosti předcházejícího tahu. Souřadnice řídících bodů tahu je uložena jako násobek velikosti předcházejícího tahu od prvního bodu předcházejícího tahu. Toto samozřejmě neplatí pro první tah, který se nemá k čemu vztahovat a tudíž nezáleží kam jej uživatel umístí. Ukládání znaku tedy spočívá v několika krocích:

1. Uživatel nakreslí znak na kreslící plochu.
2. Požádá aplikaci o uložení.
3. Prohledá se databáze znaků, zda se v ní znak již nenachází. Pokud ano, zahlásí se uživateli, že znak už v ní je obsažen.
4. Uživatel zadá název a kategorii znaku.
5. Dojde vytvoření nového záznamu v příslušné kategorii.
6. Tento záznam obsahuje název znaku a počet jeho tahů. Dále obsahuje seznam tahů i s jejich relativními pozicemi.

Pro první tah platí, že se neukládají relativní souřadnice, nýbrž jeho aktuální souřadnice (není zcela nezbytné ukládat tento údaj, ovšem dá se jej využít při vykreslování znaku ve výukovém módu). Pro ostatní tahy (počínaje druhým) se postupuje podle následujícího algoritmu:

1. Spočítá se velikost přecházejícího tahu.
2. Souřadnice vrcholů se převedou na relativní vzdálenost od prvního bodu předcházejícího tahu vzhledem k velikosti, kterou jsme spočítali v kroku 1 (pro lepší pochopení znázorňuje tento krok obrázek 4.7).



Obrázek 4.7: Relativizace tahů ve znaku.

4.4.1 Formát ukládání

Pro zachování přenositelnosti a jednoduchosti ukládání znaků byl zvolen formát XML. Původním návrhem bylo uchovávat znaky přibližně v takovéto podobě:

```
<nazev_tahu typ="typ_tahu" kategorie="kategorie">
    <tah1 x1="x1" y1="y1" .../>
    <tah2 x1="x1" y1="y1" .../>
</nazev_tahu>
<nazev_tahu typ="typ_tahu" kategorie="kategorie">
    <tah1 x1="x1" y1="y1" .../>
    <tah2 x1="x1" y1="y1" .../>
</nazev_tahu>
```

Obrázek 4.8: Původní návrh uložení znaků.

Toto uložení je člověku přirozené, avšak obsahuje určité problémy. Mezi ty hlavní patří výpočetní náročnost při vyhledávání znaku a malá flexibilita pro rozpoznávání znaků.

Nízká flexibilita — Pokud by byly v databázi uloženy přímo souřadnice vrcholů, tak by to uživatele nutilo nakreslit znak téměř ve stejné velikosti, jako byl uložen do databáze a navíc ještě na stejnou pozici. Tento způsob by učinil aplikaci naprosto nepoužitelnou. Tento problém se podařilo odstranit výše popsaným relativním způsobem ukládání.

Výpočetní náročnost — Pokud by byla databáze kompletní (okolo 6000) a uživatel by vyhledával znak, který je umístěn až na konci souboru, mohlo by prohledávání zabrat nezanedbatelnou dobu. Proto jsem se pokusil o systém ukládání, který by vyhledávání znaku znatelně urychlil.

Výsledkem snahy o odstranění těchto nedostatků byl formát, jenž byl v mnohém podobný stromové struktuře:

```
<heng>
    <ti x1="x1" y1="y1" x2="x2" y2="y2">Nazev znaku</heng>
    <na x1="x1" y1="y1" x2="x2" y2="y2">
        <pie x1="x1" y1="y1" x2="x2" y2="y2">Nazev znaku</pie>
    </na>
</heng>
<shu>
    <heng x1="x1" y1="y1" x2="x2" y2="y2">Nazev znaku</heng>
</shu>
```

Obrázek 4.9: Stromová struktura uložení znaku.

Tento systém má vynikající výsledky při vyhledávání znaků (v ideálním případě je potřeba prohledat tolik uzlů XML dokumentu, kolik obsahuje znak tahů). Ovšem pro účely výuky či vyhledávání podle názvu znaku by trpěl stejnou nevýhodou jako předchozí formát ukládáni (tzn. bylo by zapotřebí prohledávat v nejhorším případě celý soubor). Zde se mi tedy naskytly dvě možnosti:

- Používat pro každý mód (výukový a překladatelský) odlišné formáty pro ukládání znaků (s tím by souvisela redundance uložené informace).
- Pokusit se jeden z formátů vylepšit, aby nebylo zapotřebí v nejhorším případě prohledávat všechny uzly XML dokumentu.

Nakonec jsem zvolil druhou možnost, uzly s názvy znaků byly doplněny o atribut *strokes*. Tento atribut udává, z kolika tahů se daný znak skládá. Pokud tedy vyhledávám znak o sedmi tazích, nemá smysl prohledávat uzly s odlišným počtem tahů. Použitý formát nakonec vypadá takto:

```
<Pes strokes="4">
    <heng x0="200.00" y0="300.00" x1="500.00" y1="300.00">
        <wan-pie x0="0.46" y0="-0.47" x1="0.47" y1="0.00" x2="0.06" y2="0.52">
            <na x0="0.00" y0="0.44" x1="0.51" y1="0.95">
                <na x0="0.49" y0="-0.73" x1="0.66" y1="-0.54" />
            </na>
        </wan-pie>
    </heng>
</Pes>
```

Obrázek 4.10: Finální způsob uložení znaku. Za kořenovým elementem následují elementy, které určují zařazení do určité kategorie znaků (např. zvířata, kovy, apod.). V kategoriích jsou již přímo ukládány znaky pomocí výše popsaného algoritmu.

4.4.2 Ukládání náhledu

Souběžně s ukládáním znaku do XML se ukládají do souborů náhledy jednotlivých znaků pro lepší orientaci při výběru znaků k naučení. Při ukládání se vykreslí znak z databáze bez jakýchkoliv kaligrafických úprav s tlustšími liniemi, zmenší se na rozměry 32 x 32 pixelů a poté je ve formě bitmapového obrázku uložen na disk. Původním záměrem bylo ukládat náhled znaku se všemi kaligrafickými úpravami, ovšem tenké linie se po zmenšení obrázku vytrácejí, proto byla upřednostněna viditelnost znaku před vizuální krásou.

4.5 Rozpoznání znaků pro výukový mód

Když uživatel dokončí tvorbu znaku a požádá o jeho rozpoznání, dostane třída pro rozpoznávání seznam struktur *tStroke*, která v sobě obsahuje vrcholy tahu a rozpoznaný typ tahu. Postupným procházením znaků, pro které to má smysl, se pokouší najít shodu. Porovnávání probíhá v několika krocích:

1. Nejprve se porovná typ aktuálního tahu s odpovídajícím tahem v databázi.
2. Pokud jsou tyto typy shodné, spočítají se z databáze s pomocí znalosti velikosti předcházejícího tahu a jeho počátečního bodu souřadnice tahu, který se porovná s aktuálním tahem a na základě určité tolerance (uživatel není schopen nakreslit tah naprosto přesně) se rozhodne zda jsou tahy shodné. Tato tolerance se odvíjí od několika faktorů, základní tolerance je nastavena na konstantní hodnotu a se vzdáleností dvou následujících tahů se zvětšuje podle vzorce 4.1.

$$tolerance = 50 + 0,3vzdalenost \tag{4.1}$$

Obrázek 4.11 ukazuje vliv mezí na rozpoznávání znaků.

Obrázek 4.11: Obrázek vlevo ukazuje znak, jak je uložen v databázi, uprostřed znak, který aplikace rozpozná správně, vpravo který vyhodnotí už jako chybný.

Tento styl rozpoznávání je vhodný pro výuku Kanji znaků, kde je požadováno správné pořadí všech tahů, anebo pro zkušené uživatele, kteří znak znají a chtějí si jej pouze procvičit. Ovšem nepříliš znalý uživatel, který požaduje překlad znaku do jiného jazyka neví o tomto znaku nic kromě vzhledu a nemůže být tedy schopen nakreslit jej správně. Proto je pro překlad tento styl nevhodný a musel být vyvinut speciální styl pro slovníkové rozpoznávání znaků.

4.6 Rozpoznání znaků pro překladatelský mód

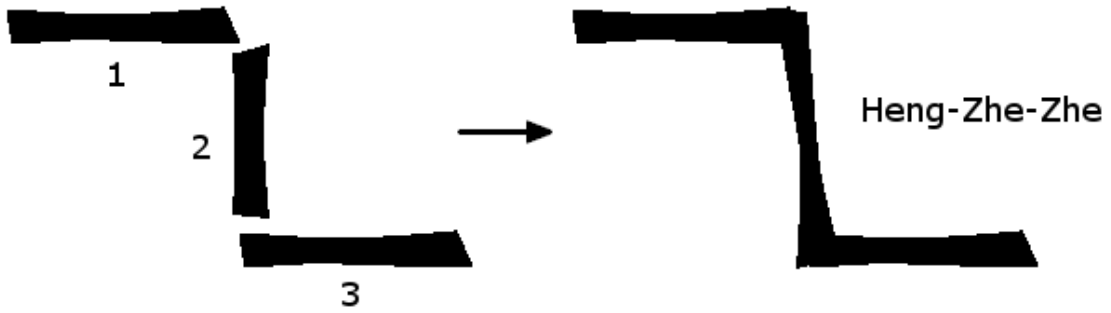
Výhodou tohoto rozpoznávání pro uživatele je, že nemusí nic vědět o stavbě znaku, stačí jej pouze nakreslit a program si již tahy vhodně převede pomocí třídy *Correct*.

4.6.1 Vyhledávání složených tahů

Nesprávné pořadí tahů není příliš velkou překážkou rozpoznávání. Základním problémem je, že uživatel nemusí rozpoznat všechny složené tahy ve znaku. Aplikace by se tedy na tento znak dívala jako na úplně jiný, přestože uživateli by připadaly vizuálně shodné. Proto je potřeba nalézt funkci, které by tyto chyby při tvorbě znaku nevadily. V zásadě připadaly v úvahu dvě možnosti:

- Ukládat do databáze všechny možnosti nakreslení znaku (ukládat postupně znaky se složenými tahy či s jednoduchými tahy, ze kterých se skládají).
- Pokusit se vyhledat složeniny přímo v seznamu tahů, které zadal uživatel.

První možnost jsem okamžitě vyloučil, neboť by došlo k několikanásobnému navýšení velikosti souboru s uloženými znaky (např. pro jeden znak se třemi složenými tahy, které se skládají se dvou jednoduchých tahů by bylo zapotřebí uložit 8 variant). Proto jsem se přiklonil ke druhé variantě. Postupným procházením se vyhledávají pravděpodobné složeniny tahů. Pokud jsou nalezeny dva znaky, z nichž jeden začíná poblíž koncového bodu druhého, zkontrolují se jejich typy. Pokud složenina těchto dvou základních tahů opravdu existuje, spojí se oba tahy do jednoho. Tímto způsobem se postupně prochází celý seznam, dokud dochází ke změnám. Příkladem postupu je obrázek 4.12.



Obrázek 4.12: Uživatel nakreslil dva tahy *Hēng* a jeden tah *Shù*. Při rozpoznávání se nejprve (1) a (2) složí do tahu *Hēng-Zhé* a tento v dalším průchodu spolu s tahem *Hēng* vytvoří tah *Hēng-Zhé-Zhé*.

Po proběhnutí této korekční funkce jsou již v seznamu správné tahy, ovšem v nesprávném pořadí. Toto již však pro vyhledávání stačí. Mnoho znaků není nutno prohledávat, pokud nesedí počet tahů, je znak okamžitě přeskočen. Pokud dojde ke shodě v počtu znaků, postupuje se následovně:

1. Tahy se porovnají s databází nejprve pouze podle typu (zanedbává se pořadí tahů).
2. Pokud je porovnání úspěšné, není ještě jisté, že se jedná o daný znak, existuje mnoho znaků, které obsahují stejné tahy, dokonce občas i ve stejném pořadí, ovšem liší se svými pozicemi. Proto se program pokusí srovnat tahy podle pozic tahů ve znaku:
   (a) Vše záleží na pozici prvního tahu, z níž se poté vygenerují pozice ostatních tahů pro porovnání. Pokud je první tah jednoznačně dán (např. pokud znak začíná

tahem *Shù* a v seznamu tahů je také pouze jeden tah *Shù*), přiřadí se tah na tuto pozici a může začít porovnávání. Pokud seznam tahů obsahuje více tahů shodného typu jako je první tah ve znaku, je třeba postupně vyzkoušet přiřazovat tyto tahy na první pozici. Poté se již porovnává stejným způsobem jako v případě porovnávání ve výukovém módu (4.5).

(b) Jestliže tahy přísluší danému znaku, ukončí se prohledávání, pokud tomu tak není, je potřeba přikročit zpět ke kroku 1.

Po skončení vyhledávání se zobrazí uživateli nalezený znak spolu se svým významem. Pokud nebyl daný znak nalezen, nemusí to ještě nutně znamenat, že se v databázi nenachází, ale pouze že nebyl rozpoznán (může být způsobeno chybným nakreslením).

4.7 Vizualizace tahů

4.7.1 Základy kaligrafie

Kaligrafií se v evropské kultuře většinou rozumí pouze umění krasopisu, tisk či kreslení krásných znaků, často na úkor čitelnosti,. U východních kultur je považováno za cosi duchovního, co vyvěrá z duše umělce, počínaje roztíráním koptu pro výrobu tuše a kreslením znaků konče [3].

V kaligrafii se v největší míře používají Kanji znaky (např.4.13), proto bylo vhodné pro lepší uživatelský dojem potřeba vylepšit vzhled znaků.

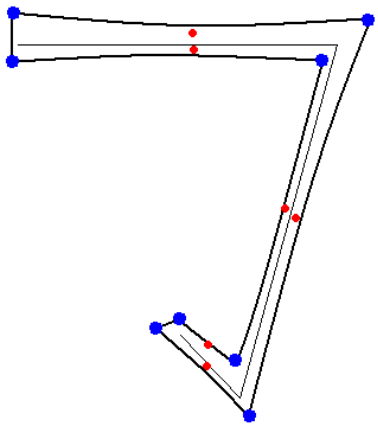
Obrázek 4.13:

4.7.2 Programová vizualizace

Pro vizualizaci tahů jsem zvolil spliny, pro které je ve wxWidgets přímá podpora. Po rozpoznání tahů je třeba vygenerovat řídící body splinu, který *obalí* křivku, kterou nakreslil na obrazovku uživatel. Tato křivka posléze vytvoří na obrazovce grafickou podobu tahu. Při generování křivky se některé body opakují, aby bylo dosaženo ostrých hran tam, kde je třeba. Každý tah má specifický tvar, ani složené tahy nevypadají jako grafické složení svých komponent, není tudíž vhodné generovat řídící body pro každý tah stejným způsobem. Proto jsou posuny řídících bodů od tahu zadaného uživatelem uloženy ve speciálním XML souboru. Soubor obsahuje postupně posuny od jednotlivých řídících bodů, kterých je $2 * pocet\_vrcholu - 1$. Záznam v XML souboru je ve tvaru:

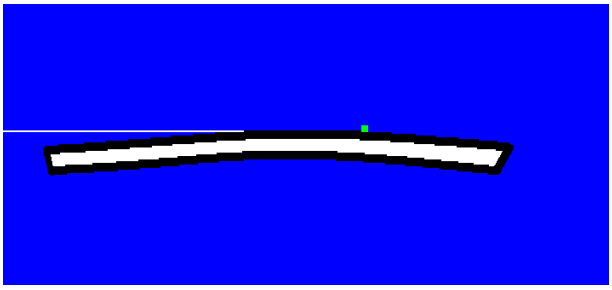
```
<tah x1="4" y1="5" .../>
```

Generování obrysového splinu naznačuje obrázek 4.14

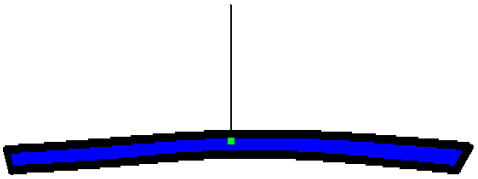


Obrázek 4.14: Je zapotřebí *obalit* základní jednoduchou křivku. Barevně označené body znázorňují řídící body splinu. Modře označené body jsou násobné body, aby bylo dosaženo ostrých přechodů. Červeně označené body slouží k prohýbání křivky.

Po vytvoření obalového splinu je tento vykreslen na obrazovku odlišnou barvou od ostatních tahů. Nyní je potřeba vyplnit jej barvou tahů. Tento bod se ukázal jako poněkud problematický, neboť vzhledem k absenci rovnice křivky není k dispozici mechanismus, který by určil, který bod je uvnitř splinu a který vně. Řešením je vedení přímky od okraje obrazovky dokud není dosaženo okraje splinu. Po překročení této hranice je jistota, že daný bod je uvnitř splinu, proto je možno jej vyplnit pomocí semínkového vyplňování. Poté stačí obarvit barvou tahů obrysový spline. Otázkou zůstává, odkud vést onu přímku, která by zjistila vnitřní bod splinu. Pokud by byla vedena vodorovně, mohlo by při vodorovných tazích dojít ke špatnému rozpoznání vnitřku, jak naznačuje obrázek 4.15. Tento problém jsem vyřešil tím, že přímka je vedena vždy z té osy, kde je větší rozpětí souřadnic bodů $(dx,\ dy)$. Pokud se tedy vykresluje znak, který je rovnoběžný s osou x, vede se přímka rovnoběžně s osou y. Příkladem správně vedeného paprsku je obrázek 4.16.



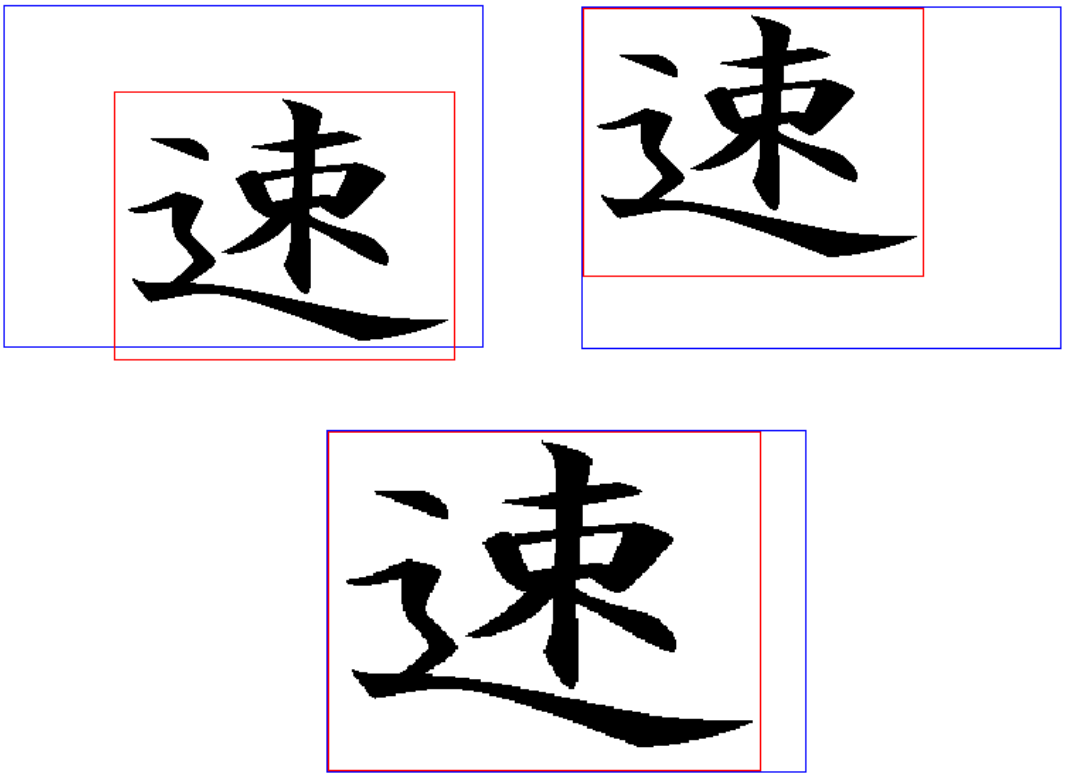
Obrázek 4.15: Přímka je vedena rovnoběžně se směrem tahu. Po protnutí hranic splinu pokračuje přes hranice, dokud přes ně neprojde. Pak předpokládá, že je uvnitř splinu a započne semínkové vyplňování (počáteční semínko je označeno zeleně). Pokud je paprsek veden souběžně s tahem, může se snadno stát, že se tahu pouze dotkne a dojde ke špatnému vyplnění jako v tomto případě.



Obrázek 4.16: Jak je z obrázku patrné, změna mezi koncovými body ve směru osy x je mnohem větší než ve směru osy y. Proto je paprsek veden kolmo na směr osy x. Tímto způsobem je možné s téměř 100% jistotou určit vnitřek splinu a korektně jej vyplnit.

4.7.3 Vystředění znaku na obrazovce

Jak již bylo řečeno, ve výukovém módu je na kreslící plochu vykreslen vyučovaný znak se zvýrazněným tahem, který je na řadě a uživatel pouze obkresluje dané tahy. Pokud by byl znak vykreslen přesně tak, jak je uložen v databázi, mohlo by docházet k potížím s velikostí okna (znak by mohl být příliš malý nebo by se naopak do okna ani celý nevešel). Proto je třeba znak před vykreslením na obrazovku vhodně zmenšit či zvětšit. Při změně měřítka se znak nejdříve zarovná k levému hornímu rohu okna, jak je vidět na obrázku 4.17. Poté se výška a šířka znaku vynásobí poměrem mezi znakem a oknem aplikace. Znaky ovšem nemusí mít rozměry ve stejném poměru jako je okno aplikace, proto se jako referenční poměr použije menší obou poměrů (poměr výšek nebo šířek). Tímto se zamezí tomu, aby část znaku byla mimo kreslitelnou oblast. Tím dostaneme největší možnou velikost znaku vzhledem velikosti okna, aby měl uživatel co nejlepší možnost kreslení.



Obrázek 4.17: Obrázek vlevo nahoře znázorňuje jak by znak vypadal po vykreslení z databáze. Modrý obdélník zastupuje kreslící plochu, červený obdélník obaluje znak. V první fázi se zarovnají levé horní rohy obou čtyřúhelníků, jak je vidět na obrázku vpravo nahoře. Nyní je třeba určit koeficient roztažení. Je vidět, že obalový obdélník znaku a kreslící plocha nejsou ve stejném poměru. Proto není možné určit poměr roztažení jako poměrem mezi výškami nebo šířkami, neboť to by mohlo vést ke roztažení, kdy by část znaku byla mimo kreslící plochu. Je tedy zapotřebí zvolit menší z obou poměrů. Poté se daným koeficientem vynásobí řídící body splinu a dojde k požadovanému roztažení znaku.

Kapitola 5

Implementace

Jelikož bylo cílem vytvořit přenositelný kód, byl pro implementaci zvolen jazyk C++ spolu s knihovnou wxWidgets pro vytvoření grafického uživatelského rozhraní. Z dalších nástrojů byla použita už jen knihovna pro práci s XML soubory TinyXML, která je rovněž naprogramována v jazyce C++. Díky použitým nástrojům by měla být aplikace přeložitelná pod většinou známých operačních systémů. Program byl testován pod operačními systémy Windows a GNU/Linux. Vzhledem k podpoře wxWidgets by neměl být problém s překladem ani pod systémem Mac OS X.

Samotná aplikace byla důsledně oddělena od GUI, proto je použitelná i s jinými toolkity pro tvorbu grafického uživatelského rozhraní.

5.1 Popis tříd

třída Kanji(*Návrhový vzor Singleton*) – Přijímá seznam bodů od grafického uživatelského rozhraní, připravuje křivky pro rozpoznávání, provádí odstraňování bodů ležících v přímce a diskretizaci křivky.

třída Stroke(*Návrhový vzor Singleton*) – Stará se o rozpoznávání tahů, postupně testuje seznam bodů, které dostala od třídy *Kanji* a poté vrací rozpoznaný typ tahu.

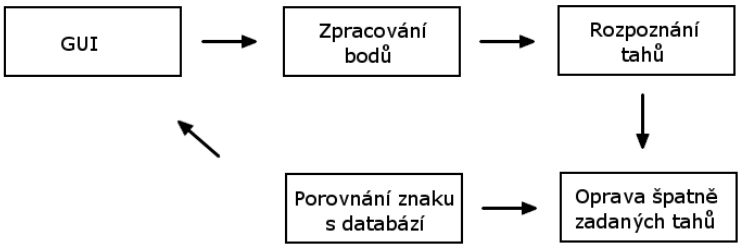
třída Correct(*Návrhový vzor Singleton*) – Třída pro korekci tahů. Jejím vstupem je seznam rozpoznaných tahů, ve kterém vyhledává složené tahy v seznamu jednoduchých či složených tahů.

třída Database(*Návrhový vzor Singleton*) – Třída pro vyhledávání znaků a ukládání znaků. Obsahuje funkce pro načtení znaku z databáze, získávání informací o znacích a jejich porovnávání se seznamem tahů zadaných uživatelem.

5.2 Popis Aplikace

Aplikace obsahuje jedno hlavní okno. V něm se nachází kreslící plocha, napravo od ní je umístěn panel, který plní v různých módech odlišný účel:

- Ve výukovém módu (obrázek 5.2) slouží pro výběr znaku k naučení. Obsahuje ComboBox pro výběr kategorie znaku a pro vybranou kategorii poté zobrazí náhledy znaků.

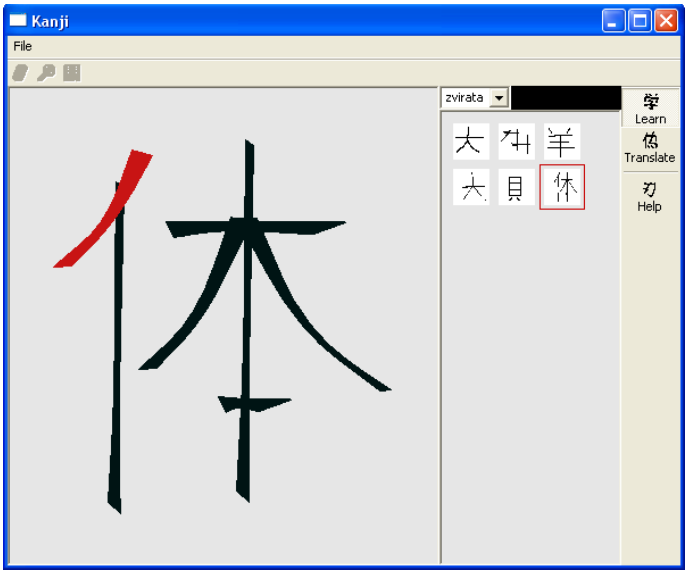


Obrázek 5.1: Obrázek znázorňuje postup aplikace při zpracování tahů.

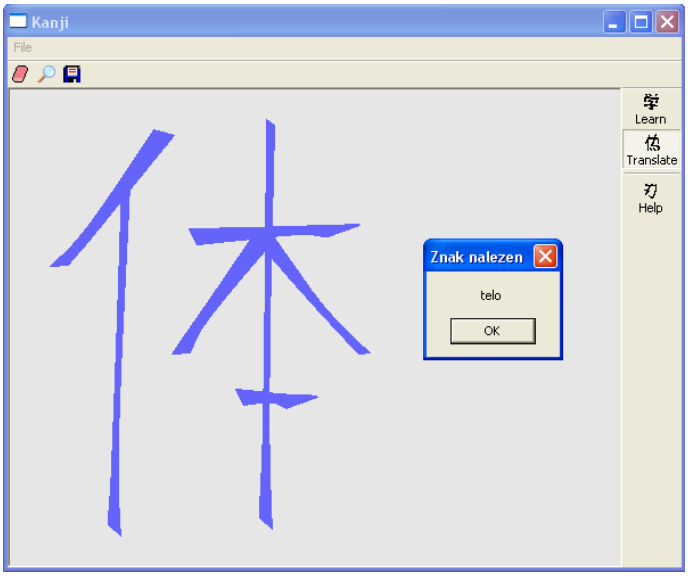
Uživatel si kliknutím na náhled vybere znak, který by se chtěl naučit a ten se mu zobrazí na kreslící ploše.

- Ve překladatelském módu (obrázek 5.3) se panel skryje. Původním záměrem bylo využít jej pro zobrazování znaků, které jsou podobné tomu, které uživatel nakreslil, avšak toto nemělo dle názoru uživatelů valný smysl.

Napravo od panelu se nachází toolbar pro přepínání mezi módem výuky a překladatelským módem. Dále nabízí tlačítko s nápovědou, které pomůže začínajícímu uživateli s orientací v programu. V překladatelském módu jsou navíc na horním toolbaru zpřístupněna ovládací tlačítka pro vyčištění kreslící plochy, rozpoznání nakresleného znaku či jeho uložení. Kreslení tahu je započato stiskem levého tlačítka myši. Po dokončení tahu se tlačítko uvolní. Pokud se uživatel při kreslení tahu splete, může pomocí stisku pravého tlačítka myši vymazat poslední tahy. Rozhraní bylo navrženo tak, aby bylo co nejjednodušší a srozumitelné pro uživatele.



Obrázek 5.2: Výukový mód.



Obrázek 5.3: Překladový mód.

5.3 wxWidgets

Volně šiřitelná multiplatformní knihovna pro tvorbu grafického uživatelského rozhraní [11]. Pod většinou operačních systémů se snaží mít nativní podporu, což zajišťuje vzhled, na který jsou uživatelé zvyklí a orientují se v aplikaci bez obtíží. Je naprogramována v jazyce C++. Knihovna wxWidgets je poměrně robustní knihovna, narozdíl od jiných toolkitů, které poskytují pouze funkčnost pro tvorbu oken a nepřeberného množství tlačítek, poskytuje wxWidgets podporu pro sockety, přehrávání zvuků, parsování XML dokumentů (ačkoliv sami tvůrci uznávají, že tento parser není příliš kvalitní a doporučují používání knihovny TinyXML), speciální okna pro vykreslování OpenGL, ...

5.4 TinyXML

Jak její název napovídá, jedná se o velmi malou knihovnu, naprogramovanou v jazyce C++, která je určena pro načítání, procházení a ukládání dat ve formátu XML [9]. TinyXML nabízí přeložení s podporou Standart Template Library nebo bez ní. Je šířena pod licencí ZLib, která umožňuje využití pro jakýkoliv komerční či open-source projekt. Drobnou nevýhodou této knihovny je absence podpory DTD či XSL, což vylučuje použití pro internetové prohlížeče a podobné projekty, ovšem toto je daň za malou velikost TinyXML. Aplikace vyžaduje podporu pouze pro jednoduchý XML dokument, proto byla knihovna TinyXML plně dostačující.

Kapitola 6

Závěr

6.1 Reakce uživatelů

Aplikace byla poskytnuta ve fázi testování pěti uživatelům, kteří nemají s japonštinou žádné zkušenosti. Všichni byli s výukou znaků velice spokojeni, pochvalovali si přehlednost programu a jednoduchost kreslení. Po naučení některých znaků poté vyzkoušeli i překladový mód a podle jejich slov byli velice příjemně překvapeni kvalitou rozpoznání. Hlavní výtky byl směrovány k malému rozpětí tolerance při kreslení tahů, proto jsem ve spolupráci s nimi upravil toleranci tahů, aby bylo vytváření tahů a znaků uživatelsky co nejpříjemnější.

6.2 Zhodnocení

Aplikace splňuje všechny požadavky, které byly vytyčeny na začátku. Největšími problémy při jejím vývoji byly konstrukce algoritmu pro odstranění nepodstatných bodů na křivce a způsob uložení znaků, který by zajistil možnost nakreslení a rozpoznání znaku v různém měřítku. Těžkosti při rozpoznávání tahů byly patrně způsobeny tím, že se jedná o první aplikaci svého druhu, tudíž nebyly dostupné žádné informační zdroje. V počátku projektu jsem se pokoušel najít inspiraci v *Mouse gestures* internetového prohlížeče *Opera*, jehož zdrojové kódy jsou volně k dispozici. Ovšem tento postup nepřinesl kýžené ovoce, neboť nalezení principů ve zdrojových kódech bylo takřka nemožné. Gesta, která používá *Opera*, jsou navíc ve srovnání s většinou japonských tahů příliš triviální. Proto bylo nutno vydat se při rozpoznávání tahů vlastní cestou.

6.3 Možná rozšíření

6.3.1 Rozšíření databáze

Vzhledem k časové náročnosti bakalářské práce nebylo v mých silách vytvořit rozsáhlejší databázi znaků. Ta v současné době obsahuje okolo 50 znaků. Tvorbu databáze by bylo navíc vhodné svěřit někomu s většími zkušenostmi s tvorbou japonských znaků (na kvalitě uloženého znaku záleží také kvalita rozpoznávání).

6.3.2 Výuka znaku přímo z japonského textu

Zajímavým rozšířením by mohlo napojení aplikace na znakovou sadu pro japonštinu. V databázi by poté u znaků mohl být uložen i jejich kód, tudíž pokud by se uživatel chtěl

naučit psaní znaku, na který narazil v textu, nemusel by jej obkreslovat a poté vyhledávat v databázi. Stačilo by jej kupříkladu zkopírovat do textového pole, které by se přidalo k aplikaci a program by už tento znak podle kódu vyhledal a zobrazil ve výukovém módu.

6.3.3 Rozšíření výuky a slovníkového překladu

Implementovaný mód výuky pouze poskytuje výuku tahů na předem vykresleném znaku. Pro plnohodnotné vyučování by zcela jistě bylo vhodné implementovat určitý systém zkoušení, kde by se náhodně generovaly názvy tahů, které má uživatel nakreslit. Dále by bylo vhodné přidat statistiky úspěšnosti a další pomůcky, které by přispěly ke kvalitnější výuce. Navíc kromě rozlišení Kanji znaků do jednotlivých kategorií (čísla, rostliny, zvířata...) existuje rozlišování na tzv. Jōyō Kanji stupně [6]. Jōyō obsahuje 1 945 znaků, které jsou rozšířením v úvodu (kapitola 1) zmíněného Tōyō Kanji. Japonští žáci by je měli zvládnout během studia základní a nižší střední školy. Jōyō je tvořeno devíti stupni, které jsou odlišené podle náročnosti, výborně se tedy hodí pro výuku a v mnoha výukových programech se používá právě rozdělení na Jōyō stupně.

Slovníkový překlad je zaměřen pouze na překlad japonština-čeština. Opačný směr překladu nebyl naprogramován hlavně z toho důvodu, že pro tento účel existuje velké množství jiných kvalitních programů. Pokud by však v budoucnu došlo k rozšíření této aplikace mezi větší množství uživatelů, bylo by vhodné obousměrný překlad doplnit, aby nebylo zapotřebí používat několik programů najednou.

6.3.4 Rozpoznávání textu v obrázcích

Díky svému oddělení od grafického rozhraní by byla aplikace využitelná například i pro rozpoznávání Kanji znaků v bitmapovém obrázku. Toto by vyžadovalo algoritmus, který by byl schopen v bitmapě vyhledat jednotlivé linie tahů. Poté by byl postup rozpoznávání znaků analogický se současným programem. Ovšem rozpoznávací algoritmus by měl velice složitou úlohu vzhledem k rozličným kaligrafickým úpravám, které často znemožňují rozpoznání znaku i člověku. Stejný znak totiž může z rukou dvou kaligrafů vypadat naprosto odlišně (6.1 a 6.2).

Obrázek 6.1:

Obrázek 6.2:

Literatura

[1] PDF dokument. Xml snadno a rychle. http://www.ics.muni.cz/zpravodaj/clanky_tisk/268.pdf. [dostupné 12.5.2008].

[2] WWW stránky. Bézier curve. http://en.wikipedia.org/wiki/Bézier_curve. [Online, dostupné 12.5.2008].

[3] WWW stránky. Calligraphy. http://en.wikipedia.org/wiki/Calligraphy. [dostupné 12.5.2008].

[4] WWW stránky. Cjk strokes. http://en.wikipedia.org/wiki/Stroke_(Chinese_character). [dostupné 12.5.2008].

[5] WWW stránky. Jazyk xml. http://en.wikipedia.org/wiki/XML. [dostupné 12.5.2008].

[6] WWW stránky. Jōjō kanji. http://en.wikipedia.org/wiki/Joyo_kanji. [dostupné 12.5.2008].

[7] WWW stránky. Kanji alive. http://en.wikipedia.org/wiki/Stroke_(Chinese_character). [dostupné 12.5.2008].

[8] WWW stránky. Projekt wakan. http://wakan.manga.cz/. [dostupné 12.5.2008].

[9] WWW stránky. Tinyxml documentation. http://www.grinninglizard.com/tinyxmldocs/index.html. [dostupné 12.5.2008].

[10] WWW stránky. Výuka počítačové grafiky cestou www. http://lubovo.misto.cz/_MAIL_/curves/. [dostupné 12.5.2008].

[11] WWW stránky. Wxwidgets – cross-platform gui library. http://www.wxwidgets.org/. [dostupné 12.5.2008].