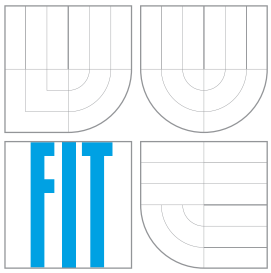
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

# MOST MEZI GLUT A KNIHOVNOU PRO TVORBU UŽIVATELSKÝCH ROZHRANÍ

BRIDGE BETWEEN GLUT AND GRAPHICAL USER INTERFACE LIBRARY

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. JAN FRIESSE
AUTHOR

VEDOUCÍ PRÁCE — Ing. MICHAL VYSKOČIL
SUPERVISOR

BRNO 2008

# Abstrakt

Tato práce se zabývá problematikou tvorby grafických 3D aplikací s použitím knihovny OpenGL. Jsou zde vysvětleny základy architektury knihovny GLUT, její hlavní výhody a nevýhody a popsány alternativní implementace GLUT API. Dále je zde popis nejznámějších grafických knihoven pro tvorbu uživatelského rozhraní, ve kterých je možné tvořit aplikace využívající OpenGL spolu s jejich hlavními výhodami a nevýhodami. Následuje návrh architektury nové implementace knihovny GLUT s použitím vybrané knihovny pro tvorbu uživatelských rozhraní. Nejdůležitější část práce se zabývá popisem vlastní implementace, která se nachází na přiloženém CD. Nakonec jsou popsány možnosti dalšího vývoje implementované knihovny.



# Klíčová slova

OpenGL, GLUT, GTK+, GtkGLArea, GtkGLExt, Motif, QT, Fox Toolkit, FLTK, LessTif, wxWidgets, FreeGLUT, OpenGLUT, GtkGLUT



# Abstract

This document describes a major problems with creating 3D applications in a graphic library named OpenGL. There are some basic informations about a GLUT library including main advantages and disadvantages and a description of alternative implementations of a GLUT API. Further content is focused on the most familiar graphic libraries for creating user interfaces which should be used to develop the OpenGL applications and with their advantages and disadvanteges too. The next part describes a draft of a new GLUT API implementation based on the selected user interface library. Major part of thesis is about implementation of GtkGLUT library. Reader can find this implementation on CD. Last part describes future development plans.



# Keywords

OpenGL, GLUT, GTK+, GtkGLArea, GtkGLExt, Motif, QT, Fox Toolkit, FLTK, LessTif, wxWidgets, FreeGLUT, OpenGLUT, GtkGLUT



# Citace

Jan Friesse: Most mezi GLUT a knihovnou pro tvorbu uživatelských rozhraní, diplomová práce, Brno, FIT VUT v Brně, 2008

# Most mezi GLUT a knihovnou pro tvorbu uživatelských rozhraní

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Michala Vyskočila

.....

Jan Friesse

6. května 2008

# Obsah

# Kapitola 1

# Úvod

Je tomu již 27 let od roku 1981, kdy se na počítačích třídy PC objevila první grafická karta standardu CGA. Jednalo se o jednoduchou kartu s nízkým rozlišením, která byla řízena procesorem. Tento koncept nebyl vhodný a proto se o tři roky později objevily karty se standardem VGA, které již obsahovaly akcelerační funkce přímo na kartě (samočinné vykreslení úsečky, obdelníka a operaci přesunu bitmapy). V jiném segmentu trhu existovaly pracovní stanice, které nejenže uměly 2D, ale také 3D grafiku. Postupem času PC svou nízkou cenou a dostatečným výkonem pracovní stanice vytlačilo. Proto se začaly zhruba kolem roku 1998 na PC objevovat první úspěšnější 3D akcelerátory, jako například S3 Savage3D, NVidia Riva TNT, ATI RAGE 128, ...

K tomu, aby bylo možné programovat 3D grafické aplikace nezávisle na použité grafické kartě, začaly vznikat standardní rozhraní. Z roku 1980 pochází první a jediný oficiální standard jménem PHIGS. Byl založen na kompletním popisu scény (tzv. retained mode) a dnes se již nepoužívá. Mnohem zajímavější je sledovat vývoj standardu OpenGL. Ten pochází z roku 1990 a vytvořila jej firma SGI pro operační systém IRIX, který byl používán jejich pracovními stanicemi. Od standardu PHIGS se lišil hlavně tím, že stavěl na popisu jednotlivých objektů (tzv. immediate mode). OpenGL tvoří velice úspěšnou platformu pro vývoj profesionálních aplikací typu CAD systémy, ale je používáno také vývojáři her.

Firma SGI kromě samotného OpenGL vytvořila také knihovnu GLUT (The OpenGL Utility Toolkit), která slouží pro zajištění platformově nezávislé inicializace OpenGL a interakcí s uživatelem. Knihovna GLUT se díky svému jednoduchému rozhraní, které je modelováno dle rozhraní OpenGL stala velice oblíbená, hlavně pro jednodušší aplikace.

Právě knihovna GLUT je ústředním tématem této práce. Cílem je implementovat knihovnu GLUT v některé ze standardních knihoven pro tvorbu uživatelských rozhraní.

Kapitola 2 na straně 3 je věnována popisu knihovny GLUT, jejích hlavních výhod a nevýhod. Dále jsou zde rozebrány různé implementace této knihovny.

Kapitola 3 na straně 5 pojednává o různých grafických knihovnách a jejich vlastnostech se zvláštním přihlédnutím na podporu OpenGL.

V kapitole 4 na straně 12 je popsána problematika integrace kódu napsaného pomocí knihovny GLUT do projektu, používajícího vybranou knihovnu pro tvorbu uživatelských rozhraní. Právě zde je také uvedeno, která knihovna je použita pro reimplementaci knihovny GLUT a jaké z toho plynou výhody a nevýhody.

Předchozí kapitoly slouží jako úvod do problematiky a byly zpracovány v rámci Semestrálního projektu. Implementace knihovny byla vytvořena v rámci Diplomového projektu a její zevrubný popis se nachází v kapitole 5 na straně 15.

V poslední kapitole 6 na straně 32 jsou rozebrány možnosti dalšího vývoje knihovny.

# Kapitola 2

# GLUT

Jak již bylo zmíněno v úvodu je GLUT (The OpenGL Utility Toolkit), knihovna pro tvorbu přenositelných OpenGL aplikací. Samo OpenGL poskytuje velice dobrou multiplatformní přenositelnost, avšak neřeší podporu vstupu a výstupu. Na programátorovy aplikace tedy zůstávají věci jako například vytvoření okna, podpora klávesnice a podpora myši, kde však neexistovala možnost vytvořit aplikaci opravdu multiplatformně. Proto vznikl GLUT, který kromě dříve popsaných věcí přidává i některé nové, mezi které patří vytvoření modelu kostky, koule, nebo slavné čajové konvice 2.1, která je používána takřka v každé demonstrační OpenGL aplikaci a stala se jeho symbolem. Navíc je k dispozici i tvorba velice jednoduchých menu.

Mezi hlavní výhody GLUTu patří:

- Multiplatformnost. To přináší hlavní výhodu v podobě jednoduché portovatelnosti na jinou softwarovou a hardwarovou platformu a do velké míry také dlouhou životnost programu.
- Jednoduché API, které je modelováno podobně jako OpenGL, přičemž za prefix je zvoleno glut.
- Relativně vysoká úroveň API. Pro vytvoření prvního programu není třeba napsat 100 řádků, jako v případě WinAPI, nebo X-lib. Stačí několik málo řádků.
- Je používáno na FIT.

Obrázek 2.1: Čajová konvice GLUT.

- Stačí pouze standardní C. Tím je zajištěna velice snadná tvorba knihoven pro jiné jazyky, které GLUT používají.

Ovšem jako každá věc má i GLUT svoje stinné stránky, například:

- Již neprobíhá aktivní vývoj.
- Licence GLUTu je relativně restriktivní a nedovoluje distribuovat modifikovanou verzi knihovny.
- Pokud program zavolá funkci `glutMainLoop`, neexistuje žádná možnost, jak se z této funkce dostat zpět.
- Neexistuje žádný callback, který by byl zavolán, pokud je ukončena aplikace. To by bylo vhodné zejména pro vyčištění paměti, zavření souborů, ...
- Neexistence podpory pro některé novější hardware vymoženosti (například kolečko myši).
- Neabstrahuje některé věci, které by mohli být potencionálně přínosné pro programátory (například vlákna).
- Pod Windows používá pro kreslení menu nativní menu WinAPI, avšak pod X kreslí přímo do OpenGL plochy, což může způsobovat problémy.

Protože však existuje relativně početná skupina lidí, kteří GLUT používají, vznikly svobodné implementace GLUTu, které se snaží některé chyby původní verze odstranit. Mezi nejznámější patří FreeGLUT a OpenGLUT.

FreeGLUT se snaží o co nejlepší kompatibilitu s původním GLUTem, avšak přidává některé nové věci. Mezi nejdůležitější patří:

- Funkce `glutMainLoopEvent` provede jednu iteraci zpracování zpráv. Tím je možné vytvořit smyčku zpráv zcela ve vlastní režii.
- Funkce `glutLeaveMainLoop` umožňuje opustit funkci glutMainLoop.
- Přidána možnost určit akci, která bude provedena při zavření okna GLUT aplikace (obvykle kliknutím na tlačítko x).
- Přidáno mnoho nových callbacků, například zpráva o uvolnění klávesy, callback určující stav menu,...
- Možnost vyrendrovat celý řetězec, místo nutnosti renderovat celý text po písmenech ve vlastní režii.

OpenGLUT je odvozen z FreeGLUTu a přidává některé experimentální nové funkce do API.

# Kapitola 3

# Knihovny pro vývoj uživatelských rozhraní

V této kapitole se nachází porovnání několika nejznámějších grafických knihoven pro tvorbu uživatelských rozhraní. Jsou popsány jednotlivé výhody a nevýhody a zvláštní pozornost je věnována jejich schopnosti pracovat s OpenGL. Mezi významná kritéria patří i uživatelská základna. Pokud je knihovna hodně používaná, je takřka jistě zaručen dostatek dokumentace i různých uživatelských fór, kam je možné se v případě problémů obrátit, a kde je zároveň již mnoho problémů vyřešeno. Dalším kritériem je i implementační jazyk. Protože cílem práce je udržet co možná nejlepší kompatibilitu se stávajícími OpenGL aplikacemi, bylo by lépe vybrat knihovnu, která má své API primárně navrženo pro ANSI C. Pokud by však žádná taková vhodná neexistovala, bylo by nutné z tohoto požadavku slevit. Protože by byla vhodná široká podpora různých platforem, měla by i samotná knihovna být pokud možno co nejvíce multiplatformní.

## 3.1 GTK+

GTK+ (The GIMP Toolkit) dostupný na `http://www.gtk.org/` byl vytvořen (jak již název napovídá) kvůli grafickému editoru GIMP (GNU Image Manipulation Program) v roce 1997. GTK+ bylo vytvořeno a primárně je stále určeno pro systémy s X Windows (dnes zejména Linux), avšak existuje i port pro operační systém Windows a Mac OS X. První verze se snažila odstranit nevýhody knihovny Motif (v této knihovně byly také napsány první verze programu GIMP), ale API bylo modelováno velice podobně. S druhou verzí GTK+ se API podstatně změnilo což znamenalo přepis mnoha aplikací, ale od té doby (2002) je zpětně kompatibilní.

Mezi hlavní vlastnosti GTK+ patří:

- Komunitní projekt, který je však podporován mnoha velkými firmami (IBM, Red Hat, Sun, Novell, ...).
- Velice rozšířená a používaná knihovna. Nejen že je na ní postaven GIMP, ale také dvě ze tří nejoblíbenějších pracovních prostředí OS Linux (Gnome, XFCE), vektorový grafický editor Inkscape, a mnoho dalších programů.
- Svobodná licence GNU LGPL, která umožňuje stavět na této knihovně i komerční aplikace, avšak zároveň zaručuje návrat případných vylepšení zpět do knihovny .

- Výborná podpora češtiny a jiných světových jazyků. GTK+ 2.0 je kompletně postaveno na UTF-8.
- Implementačním jazykem je C.
- Existuje množství bindingů do jiných jazyků, mezi nejvýznamnějšími jsou C++ (gtkmm, wxWidgets), Java (java-gnome, nebo SWT), Perl (gtk2-perl), Python (PyGTK), ...

Bohužel však ve standardním GTK+ neexistuje žádná podpora pro OpenGL aplikace. Tento problém ovšem řeší rozšíření GtkGLArea a GtkGLExt, která budou popsána v podkapitolách 3.1.2 na straně 6 a 3.1.1 na straně 6.

Pro bližší popis knihovny GTK+ viz [5].

### 3.1.1 Rozšíření GtkGLArea

Toto rozšíření vzniklo v roce 1998. V dnešní době již není původním autorem vyvíjeno (dokonce již neexistuje ani oficiální stránka, avšak balíček je stálo možné najít například na adrese `http://www.mono-project.com/GtkGLArea`) Protože jej však používá relativně hodně projektů a je k dispozici v repozitářích většiny Linuxových distribucí, je dobré se o něm zmínit.

Hlavní vlastnosti:

- Relativně malá knihovna (kompletní zdrojový kód ˜ 30KB).
- Licence GNU GPL , která je sice svobodná, ale pro komerční vývoj aplikací nevhodná.
- Multiplatformní, funkční v OS Linux a Windows.
- Implementačním jazykem je C.
- Existují bindingy do jiných jazyků, jako například C++ a Mono C#.

### 3.1.2 Rozšíření GtkGLExt

Toto rozšíření je novější než GtkGLArea (2003). Je možné jej získat z domovské stránky na adrese `http://www.k-3d.org/gtkglext/Main_Page`. Narozdíl od předchozího rozšíření je toto aktivně vyvíjeno. Má poněkud jinou filosofii než GtkGLArea, které dávalo vývojáři jednu komponentu, která sloužila pro vykreslování OpenGL. U této knihovny je možné udělat z jakékoli komponenty plochu pro zobrazování OpenGL (takže vytvoření tlačítka s animující se konvicí není problém).

Mezi hlavní vlastnosti patří:

- Podstatně větší knihovna než předchozí.
- Licence GNU LGPL , takže má stejné výhody jako GTK+.
- Multiplatformní, funkční v OS Linux, FreeBSD, IRIX, Mac OS X a Windows.
- Implementačním jazykem je C.
- Existují bindingy do jiných jazyků, jako například C++, Python, Ruby, Perl, Scheme (Gauche Scheme), což jsou jazyky (na rozdíl od C#, pro který binding k dispozici není) v Linuxové a Mac OS X komunitě velice oblíbené.

Pro kompletní dokumentaci ke knihovně viz [6].

## 3.2 QT

Knihovnu je možné nalézt na stránce `http://trolltech.com/products/qt`. První verze této knihovny spatřila světlo světa v roce 1991. O tři roky později založili hlavní vývojáři této knihovny firmu Trolltech, která pokračuje ve vývoji dodnes. První verze byly distribuovány pod licencí FreeQt, která mnoha vývojářům svobodného software nepřipadala dostatečně liberální, proto také došlo k odchodu některých vývojářů a založení vlastního projektu GTK+. Na této knihovně je založeno mnoho projektů, ovšem mezi nejvýznamnější patří uživatelské prostředí KDE, které patří k jednomu ze tří nejpoužívanějších v OS Linux. OpenGL je možné používat velice snadno, a to odvozením nové třídy ze třídy `QGLWidget`, a předefinováním metod `initializeGL`, `resizeGL`, ...

Hlavní vlastnosti knihovny:

- Je vyvíjeno komerční firmou. To sebou nese mnohé výhody, jako například dostupnost komerční podpory, ale také některé nevýhody.
- Velmi kvalitně zpracovaná dokumentace.
- Široká komunita vývojářů, dostupnost mnoha fór.
- Výborná podpora OpenGL.
- Výborná podpora češtiny, je podporováno množství kódových stránek včetně UTF-8.
- Velice rozšířená knihovna, nejen mezi komerčními vývojáři (Opera, Skype, Adobe Photoshop Album,...), ale také vývojáři otevřeného software (KDE a všechny programy v něm obsažené, což dává k dispozici obrovské množství studijního materiálu).
- Implementačním jazykem je C++, avšak je třeba používat moc (Meta object compiler), který implementuje mechanizmus signálů/slotů, introspekcí a jiných vlastností které ve standardním C++ nejsou. To však činí vývoj knihoven na QT postavených značně složitější a prakticky znemožňuje použití čistého C++.
- Knihovna má duální způsob licencování. Pro komerční vývoj je nutné si knihovnu koupit, pro vývoj otevřeného software existuje verze šířená pod GPL, přesto je však některými vývojáři otevřeného software odmítána.
- Knihovna je multiplatformní. Existuje verze pro X Windows Systém (dnes převážně OS Linux a Solaris), MS Windows, Mac OS X.
- Donedávna neexistovala svobodná implementace pro MS Windows. Ve verzi 4 však byla svobodná verze vypuštěna i pro tuto platformu.
- Knihovna prožívá možná až příliš překotný vývoj. Každé vydání nové verze (do dnešního dne 4 hlavní verze) znamená radikální změny v API, které nutí vývojáře přepisovat aplikace. Na zpětnou kompatibilitu se takřka vůbec nehledí.
- Existuje množství bindingů pro jiné jazyky, například Python (PyQT), C# (QT#), PHP (PHP-Qt), Ruby (RubyQT), ...

Pro podrobnější informace o knihovně viz [10].

## 3.3 FLTK

FLTK (Fast Light Toolkit) je možné získat z adresy `http://www.fltk.org/`. Hlavním cílem vývojářů bylo vytvořit malou, jednoduše použitelnou knihovnu pro vývoj grafických uživatelských rozhraní, zejména s podporou 3D, která bude co nejrychlejší. Zajímavé na něm je, že obsahuje vlastní implementaci emulace knihovny GLUT.

Mezi hlavní vlastnosti patří:

- Knihovna je psána v jazyce C++, avšak nepoužívá šablony a výjimky. To do jisté míry zjednodušuje vývoj případných nástaveb.
- API knihovny je kompaktní a srozumitelně navržené.
- Knihovna sama o sobě je velice malá (zabalený zdrojový kód je jen o ˜ 200KB větší než GtkGLExt, staticky slinkovaná aplikace typu Hello World, má asi 100KB).
- Knihovna je multiplatformní. Existuje verze pro X Windows Systém (dnes převážně OS Linux a Solaris) a MS Windows.
- Dobrá podpora OpenGL. Je možné použít buď emulaci knihovny GLUT, nebo vytvořit odvozenou třídu ze třídy `Fl_Gl_Window`, kde se poté předefinuje metoda `draw`, sloužící pro vykreslení obsahu. Zde je již možné použít standardní OpenGL funkce.
- Knihovna je šířena pod GNU LGPL licencí s výjimkou, která umožňuje statické linkování knihovny k aplikaci. Je tedy vhodná i pro komerční vývoj.
- Relativně nedostatečná, roztříštěná dokumentace. Jako nejlepší zdroj informací poslouží samotné zdrojové kódy, jejichž studování je mnohem snadnější, než u jiných knihoven.
- Ve standardní knihovně neexistuje podpora pro UTF-8. Existuje sice neoficiální verze, která tento problém řeší, nebo je také možné použít vývojovou verzi FLTK 2, nicméně pokud je požadována oficiální stabilní verze, není možné je použít.
- Nefunguje zadávání českých znaků (znaků zadávané přes mrtvou klávesu) v systémech s X Windows. Tento problém řeší vývojová verze FLTK 2.
- Málo aktivních vývojářů. Po vydání první verze opadlo prvotní nadšení a nyní vyvíjená verze FLTK 2 je dlouhodobě nedokončena a nic nenasvědčuje tomu, že by se tato nepříjemná situace měla v brzké době změnit. FLTK 2 se navíc vyznačuje značným množstvím chyb, proto je nepoužitelná tam, kde je požadována stabilita.
- Menší komunita uživatelů. Pokud člověk narazí na problém, je mnohdy lepší než se pokoušet ptát na fórech, prostudovat zdrojový kód knihovny.

Pro podrobnější informace o knihovně viz [3].

## 3.4 FOX Toolkit

Tuto knihovnu je možné získat z adresy `http://www.fox-toolkit.org/`. Tato knihovna pro tvorbu grafických uživatelských rozhraní je zvláště oblíbená u tvůrců profesionálních systémů (řídící systémy, informační systémy,. . . ). Jedná se v podstatě o dílo jednoho člověka, avšak vývojářská komunita postupně narůstá. Podpora pro OpenGL je obsažena ve třídě

`FXGLCanvas`. Instanci této třídy je možné zaslat zprávu `makeCurrent`, a potom je jakékoli vykreslování OpenGL směřováno do této instance.

Hlavní vlastnosti knihovny:

- Kompaktní a ucelené API, které je relativně jednoduché na naučení. Protože se jedná o projekt (v podstatě) jednoho člověka, je API velice jednotné.
- Dobrá podpora OpenGL.
- Knihovna je multiplatformní. Měla by být provozovatelná na systémech s X Windows (OS Linux, Solaris, FreeBSD, ...), Mac OS X a Windows.
- Hlavním implementačním jazykem je C++. Existuje však mnoho bindingů do jiných jazyků, namátkou například Ruby (u programátorů v tomto jazyce je FOX Toolkit asi nejoblíbenější knihovna uživatelského rozhraní), Python, Eiffel, ...
- Relativně nedostatečná, roztříštěná dokumentace. Zdrojový kód je ovšem srozumitelný, a slouží jako dobrá dokumentace.
- Knihovna je šířena pod GNU LGPL licencí.
- Menší komunita uživatelů, při problému opět platí, že je nejlepší prostudovat zdrojové kódy.
- V posledních verzích přibyla podpora UTF-8, avšak z neznámých důvodů není možné v systémech s X Windows zadávat české znaky (lépe řečeno obecně veškeré znaky zadávané přes mrtvou klávesu). Ve starších verzích (až do 1.4 včetně) sice nebyla podpora UTF-8, ale zadávání českých znaků fungovalo. Ve Windows tento problém není.

Pro podrobnější informace o knihovně viz [4].

## 3.5 Motif

Tato grafická knihovna je ze všech jmenovaných nejstarší (vznikla kolem roku 1980). Její API je definována standardem IEEE 1295. Motif byl nejrozšířenější knihovna pro tvorbu grafických uživatelských rozhraní pro komerční Unixové systémy. Jedna z posledních verzí 2.1 přinesla dokonce podporu pro Unicode. Na jejím základě stály další úspěšné projekty, jako například grafické uživatelské rozhraní CDE (Solaris OS), nebo 4Dwm (IRIX). Nejznámější implementace byly komerční, avšak existuje i klon jménem LessTif, který je šířen pod licencí GNU LGPL. Tato implementace je obsažena ve většině repozitářů svobodných Unixových systémů, nebo je možné ji získat z adresy `http://www.lesstif.org/`. Implementace této knihovny se již na novější projekty nepoužívají a lze říct, že byla nahrazena knihovnou GTK+.

Hlavní rysy standardu Motif, respektive knihovny LessTif:

- Ucelené propracované API, na kterém je vidět dlouholetý vývoj, a to jak pozitivně, tak negativně.
- Existence velkého množství komponent třetích stran, které je možné stáhnout, případně zakoupit.

- Sama specifikace je multiplatformní, ovšem do operačního systému MS Windows a Mac OS její implementace příliš nepronikly. LessTif je určen pro systémy s X Windows (tedy OS Linux, FreeBSD, v Solaris OS existuje komerční implementace dodávaná rovnou se systémem) a popřípadě Windows, na kterém je nainstalován Cygwin spolu s nějakým X Windows Serverem (například Cygwin X.org).

- Hlavním implementačním jazykem je C. Knihovna ovšem vyžaduje docela dobré porozumění knihovnám XLib a Xt, protože staví na jejich programovacím modelu.

- Existuje velice dobrá dokumentace, které je obrovské množství (včetně tištěných manuálů). Jedná se asi o jedinou knihovnu pro tvorbu grafických uživatelských rozhraní, která má k dispozici nápovědu v podobě manuálových stránek.

- Komunita uživatelů je dnes již velice malá, fóra jsou velice málo aktivní. Většina odpovědí na otázky se však dá najít v archivech.

- Knihovna měla po celou dobu relativně stabilní API, takže starší programy fungují bez problémů.

- Dobrá podpora OpenGL. Základem je komponenta `GLwMDrawingArea`, která představuje kreslící plochu pro OpenGL příkazy. Poté jen stačí tuto komponentu nastavit jako aktivní pomocí funkce `GLwDrawingAreaMakeCurrent` a volat příkazy OpenGL.

Pro podrobnější informace o knihovně viz [7].

## 3.6 wxWidgets

Tato knihovna se od předešlých liší hlavně v tom, že se jedná jen o C++ nástavbu nad jinými knihovnami, konkrétně pod X Windows jde o GTK+ a Motif, pod Windows Win32API a pod Mac OS X rozhraní Cocoa nebo starší Carbon. Zajímavostí je také podpora systému OS/2. Knihovnu je možné získat z adresy `http://wxwidgets.org/`. Knihovna vznikla v roce 1992 pod názvem wxWindows, avšak v roce 2004 musela být přejmenována, kvůli vlastnictví pojmu Windows firmou Microsoft.

Mezi hlavní vlastnosti knihovny patří:

- Implementačním jazykem je C++. Ovšem kvůli stáří knihovny, nejsou použity pokročilejší vlastnosti tohoto jazyka, navíc si mnoho vlastností pozdějších standardů C++ implementuje sama a většinou nekompatibilně (například typ String).

- Existuje relativně hodně bindingů do jiných jazyků, namátkou Python, Haskell, Lua, Smalltalk, ... ovšem ne vždy je kvalita stejná. Zajímavostí je také implementace jazyka Scheme s názvem PLT Scheme, která používá svou vlastní nekompatibilní implementaci wxWindows, která vychází ze starší verze, avšak je velice hluboce integrována do jazyka. Tím se liší od jiných bindingů, které jsou většinou jen tenké a mnohdy příliš nezapadají do konceptu jazyka.

- Dokumentace je kvalitní, existuje dokonce kniha o programování v této knihovně zdarma.

- Komunita uživatelů je početná, fóra jsou velice aktivní. Je relativně hodně používána i vývojáři komerčního software (příkladem je například antivirový systém AVG).

- Vývoj knihovny nedělá velké skoky a API je relativně stabilní. Vývoj je však dostatečně živý.
- Licence této knihovny je velice podobná GNU LGPL s výjimkou, která umožňuje redistribuovat odvozené dílo v binární formě pod jinou licencí
- Podpora pro OpenGL je dobrá. Jejím základem je třída `wxGLCanvas`, od které je nutno odvodit novou třídu, kde je předefinována metoda `OnPaint`. V této metodě je možné používat volání OpenGL.

Pro podrobnější informace o knihovně viz [11].

# Kapitola 4

# Návrh řešení

Tato kapitola se zabývá návrhem vlastního řešení GLUT knihovny nad knihovnou pro tvorbu uživatelských rozhraní. Nejprve bude detailně rozebráno, která knihovna pro tvorbu uživatelských rozhraní byla zvolena a proč. Následně bude uvedena základní architektura, na které by celá knihovna měla stavět, doplněná o výhody a nevýhody takového postupu.

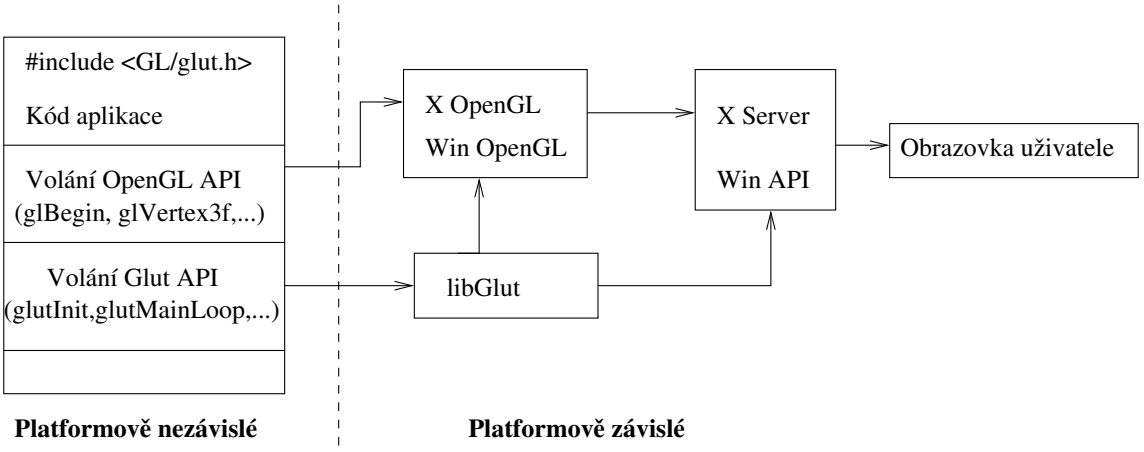
První vlastností, kterou by knihovna měla splňovat je dostatečná podpora OpenGL. Tedy mělo by být možné vytvořit nějakou vykreslovací plochu, ve které bude možno používat příkazy OpenGL API. Rozhodně by se nemělo jednat o nástavbu (například objektovou) bez možnosti přístupu k originálním funkcím OpenGL, tím by se situace stala mnohem složitější. Dále je nutné podpora více oken v aplikaci včetně vykreslovacích ploch a z toho vyplývající možnosti volby aktivní vykreslovací plochy. Tyto vlastnosti splňují všechny popsané knihovny pro tvorbu uživatelského rozhraní.

Další vlastností, která již však není úplně nezbytná, je co největší multiplatformnost knihovny. Lépe řečeno by knihovna měla podporovat alespoň tři nejvýznamnější systémy, tedy X Windows System (reprezentovaný dnes nejvíce OS Linux, FreeBSD a Solaris), Mac OS X a Windows. Tato vlastnost ze seznamu vyřadí knihovnu Motif. Ostatní knihovny tuto vlastnost splňují.

Podstatnou vlastností je také použitá licence knihovny. Protože autor této práce je fanouškem otevřeného software, mělo by se jednat o otevřenou knihovnu, v ideálním případě vyvíjenou komunitou vývojářů svobodného software. Zároveň by však autor byl relativně rád, pokud by se knihovna rozšířila, tudíž se jako nejlepší licence jeví GNU LGPL, nebo BSD/MIT. Tuto vlastnost nesplňuje jediná knihovna a to QT.

Poslední vlastností je použitý implementační jazyk. Ten do velké míry ovlivní podporu starších programů. I když je C++ s C velice kompatibilní, tato kompatibilita se týká jen programů, psaných v normě ANSI. Mnoho starších programů je psáno v normě K&R což bude pro C++ problém. Bohužel ještě větší množství nových programů je psáno tak, že ANSI normu nedodržují, a překlad pod C++ je možný jen s obtížemi. C má navíc podstatnou výhodu v tom, že pro něj existuje větší množství kompilátorů, které většinou normu implementují lépe a celou. Naopak u C++ v podstatě neexistuje překladač, který by normu tohoto jazyka implementoval se všemi jejími vlastnostmi. Proto tedy bude lepší použít knihovnu, psanou v čistém jazyce C. Toto splňuje grafická knihovna GTK+ a Motif. Motif jsme již vyloučili, takže máme na výběr mezi GtkGLArea a GtkGLExt. Protože je knihovna GtkGLExt živější, má větší komunitu a kvalitnější dokumentaci, poslouží jako vhodný základ.

Nyní nastává vhodný čas detailně rozebrat návrh nové knihovny. Zjednodušené schéma knihovny GLUT je zobrazeno na obrázku 4.1.



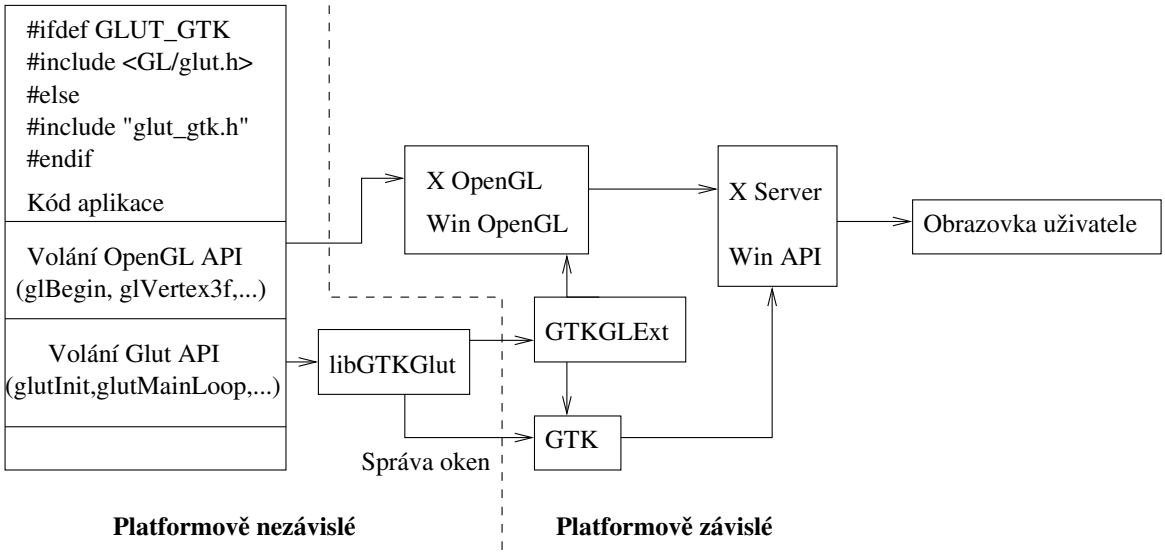

Obrázek 4.1: Schéma aplikace používající knihovnu GLUT

Jak je vidět, aplikace sama o sobě je platformově nezávislá, pokud používá jen volání knihoven OpenGL a GLUT. Tato volání jsou předána na bedra jednotlivých knihoven, které jsou platformově závislé. Volání OpenGL jsou předána v závislosti na platformě buď X Serveru (tato cesta může být nejen v rámci lokálního počítače, ale také po síti počítači jinému), který je může předávat dál například DRI, odkud putují do grafické karty, která provede vykreslení. Na Windows je postup obdobný, jen jsou volání předána Win API, které je posílá ovladači grafické karty. Zajímavostí OpenGL je také to, že pokud náhodou nějaká funkce není podporována, musí být implementována softwarově, která se na této cestě nachází také. Nás však mnohem více zajímá, co se děje v knihovně GLUT. Ta je nucena komunikovat nejen se samotnou implementací OpenGL, ale navíc také X Serverem, nebo Win API za účelem implementace operací pro vytvoření okna, zpracování vstupních událostí, nastavení vlastností okna, ... To celé činí knihovnu GLUT relativně složitou na implementaci, ale také pozdější správu a studium zdrojových kódů.

Díky návrhu nové knihovny bude odstraněno nejen licenční omezení původní knihovny GLUT, ale také technická omezení spojená se složitostí. Pokud bude knihovna dostatečně jednoduchá, bude také jednodušeji spravovatelná a případně lépe upravitelná ke speciálním účelům. Zjednodušené schéma architektury je na obrázku 4.2.

Jak je vidět, v diagramu přibyla vybraná knihovna pro tvorbu grafického uživatelského rozhraní. Co je však mnohem důležitější, nová knihovna GLUT se stala platformově nezávislou. Původní správa oken, kterou knihovna zajišťovala, je nyní ponechána na GTK+. Komunikace s OpenGL je ponechána na knihovně GtkGLExt. Tím byl zajištěný relativně rychlý vývoj knihovny, spojený s celkovou jednoduchostí implementace a čistotou zdrojových kódů (tím je myšleno hlavně minimalizování použití preprocesoru). Další výhodou je také možnost přirozeně odděleného vývoje jednotlivých částí. K případnému studiu knihovny by měla být dostačující znalost GTK+, což je podmínka splnitelná za mnohem kratší časový interval, než studium Xlib, Win API, ...

Snahou také je, aby knihovna pokryla co možná nejvíce funkcí z původní knihovny GLUT a dosáhla maximální kompatibility. Její cíle jsou tedy podobné jako u knihovny FreeGLUT. Za referenční knihovnu lze považovat právě FreeGLUT, který postupně nahrazuje původní GLUT a má svobodnější licenci. Co se týče rozšíření, které knihovna FreeGLUT zavedla, bylo by maximálně vhodné je implementovat také, jelikož se nejedná jen o jakési




Obrázek 4.2: Schéma aplikace používající novou knihovnu GtkGLUT

ad-hoc vlastnosti, ale o implementaci velice žádaných rozšíření.

Knihovna FreeGLUT zároveň slouží jako zdroj některých zdrojových kódů, které nedává smysl psát znovu (například popis GLUT konvice, popis vektorových fontů,...) a zároveň demonstrační příklady dodávané s knihovnou poslouží jako dobrý test kompatibility implementace.

# Kapitola 5

# Implementace

V této kapitole bude podrobně popsána implementace vlastní knihovny. U popisů funkcí bude uveden zejména způsob implementace, hlavní problémy které při implementaci nastaly a jejich řešení. Pokud je implementace triviální, není funkce zmíněna vůbec, nebo je zmíněna nepřímo, například v popisu datových struktur. Dokumentace tedy neslouží jako referenční příručka pro použití knihovny. Jelikož však byla knihovna primárně vyvíjena za účelem dosažení maximální kompatibility s knihovnou GLUT verze 3.7 a rozšířeními implementované v knihovně FreeGLUT 2.4.0, je možné použít jako referenční dokumentaci originální popis GLUT API (ve verzi 3 viz [1]), nebo dokumentaci ke knihovně OpenGLUT (viz [9]), jelikož FreeGLUT bohužel dokumentaci API neobsahuje.

## 5.1 Použité konvence a datové struktury knihovny

Každá exportovaná funkce má, stejně jako v originální knihovně GLUT, za prefix zvolen `glut`, exportovaná makra potom `GLUT_`. Interní funkce, globální proměnné, definice struktur a makra začínají vždy řetězcem `__gtkglut` a `__GTKGLUT`. Takto označené součásti by uživatel knihovny neměl ve svém programu nikdy použít.

Podle pravidla "Ukaž mi svůj kód a skryj datové struktury a nebudu rozumět. Ukaž mi datové struktury a obyčejně nebudu potřebovat tvůj kód, většinou mi bude jasný" je nyní správný čas popsat co nejdůkladněji vnitřní datové struktury knihovny obsažené v souboru `gtkglut_internal.h`.

Interní stav knihovny je uložen v globální proměnné `__gtkglut_context` (dále jen kontext knihovny), která je ukazatelem na strukturu `__gtkglut_context_struct`. Mezi hlavní položky této struktury patří:

- `is_inicialized` - Určuje, zda byla knihovna zinicializována voláním funkce `glutInit`.
- `window_init_geometry` - Struktura `__gtkglut_geometry_struct` obsahující počáteční pozici a velikost nově vytvořených hlavních oken.
- `window_init_iconic` - Proměnná udávající, zda se nová hlavní okna mají vytvářet minimalizovaná.
- `gl_debug` - Určuje, zda byl zadán parametr `-gldebug`, čímž dojde k otestování chyb OpenGL funkcí `glGetError` a případný výpis chyby v čitelné podobě po každém zpracování události.

- `direct_render` - Pokud je tato hodnota nastavena na `TRUE`, je použito přímé renderování OpenGL (tj. hardware). Při `FALSE` je použita software emulace. Existuje také speciální hodnota `__GTKGLUT_DIRECT_RENDER_NOT_SET`, kdy je nejprve proveden pokus o přímé renderování a v případě neúspěchu je použito nepřímé.

- `display_mode` - Určuje výchozí vykreslovací mód, se kterým budou vytvářena nová okna.

- `elapsed_time_timer` - Časovač, který udává délku běhu programu od první inicializace knihovny.

- `callback_idle` - Zde je uložena funkce, která má být zavolána v klidovém stavu knihovny. Registrace se provede voláním funkce `glutIdleFunc`.

- `actual_gl_config` - Objekt knihovny GtkGLExt, který reprezentuje aktuální konfiguraci OpenGL. Je tvořena z proměnné `display_mode`. Pokud obsahuje NULL, nebyl nastaven mód podporovaný grafickou kartou.

- `window_array`, `menu_array` - Dynamické pole všech aktivních oken a menu. Pokud je v poli některá z hodnot `NULL`, došlo ke zrušení položky.

- `current_gl_drawable`, `current_gl_context` - Objekty knihovny GtkGLExt. Určují aktuální GDK kreslící okno a OpenGL kontext. Vnitřně se tyto proměnné nastavují ve funkci `glutSetWindow`. V případě že je zrušeno aktivní okno jsou tyto proměnné nastaveny na NULL.

- `current_window`, `current_menu` - Aktivní okno a menu. Tyto hodnoty vracejí funkce `glutGetWindow`, `glutGetMenu` a jsou nastavovány pomocí funkcí `glutSetWindow` a `glutSetMenu`.

- `cursors_cache` - Vyrovnávací paměť pro kurzory myši. Jedná se o pole, které je při startu aplikace naplněno hodnotami `NULL`. Pokud uživatel požaduje změnu kurzoru voláním funkce `glutSetCursor`, je nejprve nalezena položka v poli která náleží požadovanému kurzoru, a v případě hodnoty `NULL` je vytvořen nový objekt kurzoru, jehož odkaz je uložen do pole. Při příštím volání je použit již vytvořený kurzor. Speciální postavení má prázdný (neviditelný) kurzor. Ten, na rozdíl od ostatních typů kurzorů, není standardně předdefinován v knihovně GTK+ a musí být vytvořen za běhu z pixmapy. Po vytvoření je odkaz na něj uložen do proměnné `cursor_none`.

- `keyboard_modifiers` - Modifikátory klávesnice. Standardně je nastavena na hodnotu `__GTKGLUT_UNDEFINED`. V případě volání uživatelských funkcí pro zpracování vstupu z klávesnice a myši je nastavena na správnou hodnotu a je možné ji získat voláním funkce `glutGetModifiers`. Po dokončení volání uživatelské funkce je opět nastavena na nedefinovanou hodnotu.

- `menu_status_func`, `menu_state_func` - Zde jsou uloženy odkazy na uživatelem definované funkce, které mají být volány v případě změny stavu menu. Nastavení se provede voláním funkce `glutMenuStatusFunc` a `glutMenuStateFunc`.

- `action_on_window_close` - Proměnná zavedená v souvislosti s dosažením kompatibility s knihovnou FreeGLUT. Udává, co se má stát, pokud uživatel v okenním manažeru uzavře okno (obvykle stiskem tlačítka se symbolem křížku). Standardní

GLUT provede ukončení aplikace. FreeGLUT přidává možnosti vyskočení z hlavní smyčky a pokračování v běhu, až do uzavření posledního okna, kdy vždy provede vyskočení z hlavní smyčky. Proměnná se v uživatelském programu nastavuje voláním funkce `glutSetOption`.

- `no_active_toplevel_windows` - Položka sloužící zejména ke zvýšení výkonu. Udává počet aktivních oken nejvyšší úrovně. Pokud hodnota klesne na nulu, je provedena akce definovaná v `action_on_window_close`. Položku by bylo možné vypustit a provést výpočet počtu oken po každém zavření okna, avšak v aplikacích pracujících s mnoha okny by se jednalo o velmi neefektivní operaci.
- `current_active_menu_window_id` - Okno, ze kterého bylo vyvoláno otevření menu. Používá se k volání uživatelské funkce definované při zvolení položky menu k nastavení aktivního okna. Tato proměnná může být globální, jelikož je aktivní vždy maximálně jedno menu.

Veškeré informace uložené o oknech jsou, jak již bylo dříve řečeno, k dispozici v proměnné `window_array`. Jedná se o dynamicky rostoucí pole struktur `__gtkglut_window_struct`. Ta obsahuje následující položky:

- `window`, `drawing_area`, `event_box`, `fixed` - GTK+ komponenty které vytváří vlastní okno a kreslící plochu. Více o způsobu uspořádání a vnoření komponent bude popsáno v části 5.3 na straně 21 zabývající se okny.
- `parent_window` - Číslo rodičovského okna. Pokud se jedná o okno nejvyšší úrovně, je tato proměnná nastavena na hodnotu -1.
- `first_displayed` - Určuje, zda bylo okno již někdy zobrazeno. Pokud je okno zobrazeno úplně poprvé, je dle definice nutné zavolat uživatelskou funkci pro obsluhu změny velikosti okna. Před prvním zobrazením okna je proměnná nastavena na `FALSE`, pak už vždy na `TRUE`.
- `fullscreen` - Proměnná použitelná pouze u oken nejvyšší úrovně. Pokud je volána funkce `glutFullScreen`, je tato proměnná nastavena na `TRUE`. Po změně velikosti je zpět nastavena na standardní hodnotu `FALSE`.
- `child_list` - Spojitě vázaný seznam potomků okna. Standardně je `NULL`, což značí nepřítomnost podoken. Pokud je proměnná neprázdná, udávají jednotlivé položky seznamu čísla oken (nejsou to tedy přímé odkazy na strukturu).
- `window_visible` - Určuje zda je okno viditelné. Kromě očekávaných hodnot `TRUE` a `FALSE`, může také nabývat hodnoty `__GTKGLUT_UNDEFINED`. Ta znamená, že buď současný stav okna ještě není znám, nebo uživatel právě provedl registraci funkce `glutVisibilityFunc`. Tehdy musí být zaručeno doručení zprávy o změně viditelnosti okna uživatelské funkci ihned, jakmile se tak stane. Jelikož však uvnitř knihovny dochází k odstraňování duplicitních zpráv o viditelnosti okna, je nutné nastavit právě tuto hodnotu, která není ani `TRUE`, ani `FALSE` a způsobí zavolání uživatelské funkce.
- `current_cursor` - Identifikátor kurzoru, který je oknu přiřazen. Uživatelem je nepřímo nastavován funkcí `glutSetCursor`, avšak hodnota této proměnné se využívá pouze ve funkci `glutGet`.

- `damaged` - Určuje, zda je okno poškozeno a potřebuje překreslit. Nastavuje se při změně velikosti okna a při vytvoření okna.
- `iconified` - Pouze pro okna nejvyšší úrovně. Pokud je okno ikonifikováno (minimalizované), je proměnná nastavena na `TRUE`, jinak `FALSE`.
- `callback_display` - Uživatelská funkce, která se volá při požadavku na překreslení okna (expose). Tato hodnota je při vytvoření nastavena na `NULL`, ale jako jediná musí být uživatelem nastavena na jinou hodnotu pomocí funkce `glutDisplayFunc`, jinak dojde k ukončení programu.
- `callback_reshape` - Uživatelská funkce, která se volá při změně velikosti okna. Uživatel hodnotu nastavuje pomocí funkce glutReshapeFunc. Pokud není nastavena, tedy její hodnota je NULL, dochází k volání standardního kódu který provede zavolání OpenGL funkce glViewport s parametry 0, 0, nová šířka okna, nová výška okna.
- `signal_*` - Množina proměnných, kterými se nastavují jednotlivé uživatelské funkce. Struktura bude podrobněji rozebrána dále.
- `callback_close` - Rozšíření zavedené knihovnou FreeGLUT. Tato funkce je volána při uzavření okna, a to jak v případě uživatelského, tedy v okenním manažeru, tak i voláním `glutDestroyWindow`.
- `mouse_button_menu` - Pole menu asociovaných k jednotlivým tlačítkům myši. Každá položka pole reprezentuje jedno tlačítko (v knihovně GtkGLUT má uživatel možnost využít až 5 tlačítek) a jeho hodnota je identifikátor menu, které má být zobrazeno.
- `user_data` - Libovolná, uživatelsky definovatelná hodnota typu odkaz. Tato položka je rozšířením zavedeným v knihovně FreeGLUT. Pro nastavení a čtení jsou určeny funkce `glutSetWindowData` a `glutGetWindowData`. Standardní hodnota je NULL.

Jednotlivé položky typu `signal_*`, jsou struktury typu `__gtkglut_signal_struct`, jejíž položky jsou následující:

- `gtk_event_handler` - Určuje GTK+ handler, se kterým má být asociována určená GTK+ událost. Ve své podstatě se jedná o nadbytečnou položku. Ve funkcích, které provádí vlastní registraci by mohla být zadána přímo hodnota funkce, nicméně takto jsou všechny položky nastavovány na jednom místě ve zdrojovém kódu, konkrétně ve funkci pro vytvoření okna, a případná změna je tedy snadnější.
- `glut_callback_handler` - Uživatelem definovaná funkce, která má být provedena.
- `signal_id` - GTK+ identifikační číslo signálu. Nastavuje se při zaregistrování události. Používá se pro odregistrování události.
- `event_added` - Určuje, zda již byla GTK+ událost přidána do masky událostí a je tedy možnost provádět příjem.

Jak již bylo řečeno dříve, v kontextu knihovny se také nachází dynamické pole jednotlivých menu. Ta jsou tvořena strukturou `__gtkglut_menu_struct`, která má následující položky:

- `cached_gtk_menu` - Položka ve které je uloženo dříve vytvořené GTK+ menu. Pokud je položka naplněna a nedošlo k žádné změně položek menu, nebo podmenu, použije se tato proměnná rovnou pro zobrazení na obrazovce. Jinak dochází k novému vytvoření GTK+ menu, které je uloženo právě do popisované proměnné.
- `func` - Uživatelská funkce, která má být volána při aktivaci položky menu.
- `child_items` - Seznam položek menu.
- `in_use` - Určuje, zda je menu aktuálně zobrazeno. Pokud je zobrazeno, nemělo by být s menu žádným způsobem manipulováno.
- `popuped_from_window_id` - Identifikátor okna ze kterého došlo k zobrazení menu.
- `need_refresh` - Položka je nastavena, pokud došlo k jakékoli změně v menu, nebo jeho podmenu.
- `parent_menus` - Seznam odkazů na menu, které mají aktuální menu zaregistrováno jako svoje podmenu.
- `user_data` - Stejná položka jako v případě okna. Pro nastavení a čtení jsou určeny funkce `glutSetMenuData` a `glutGetMenuData`.

Jednotlivé položky menu jsou struktury typu `__gtkglut_menu_item_struct` a obsahují:

- `label` - Textový řetězec, který slouží jako popiska položky.
- `parent_menu_id` - Rodičovské menu, kterému daná položka náleží. Položka je index, nikoli přímý odkaz.
- `submenu_id` - Pokud je položka typu podmenu, je hodnota rozdílná od 0. Pro standardní položky je hodnota nastavena právě na 0.
- `value` - Hodnota, která je nastavena uživatelem při vytváření položky menu. Tato je předávána do uživatelem definované funkce při aktivaci položky.

## 5.2 Inicializační funkce

Inicializační funkce jsou volány zejména na začátku programu. Jednotlivé definice se nachází v souboru `gtkglut_init.c`.

Ve většině programů bude jako jedna z prvních funkcí použita `glutInit`. Ta provádí inicializaci knihovny. Jejími argumenty jsou `argcp`, což je ukazatel na počet parametrů příkazového řádku a `argv`, který je pole řetězců příkazového řádku. Obě hodnoty mohou být po dokončení volání této funkce změněny.

Parametry příkazového řádku jsou stejné jako u knihovny GLUT, tedy:

- `-display DISPLAY` - Určuje display, na kterém má být prováděno vykreslování. Tento parametr je převeden na `--display` a předán na bedra funkce `gtk_init`. Hodnota má efekt pouze u systémů s X Windows a pokud není zpracována, je na konci volání funkce z parametrů odstraněna.
- `-geometry WxH+X+Y` - Určuje počáteční pozici a velikost oken hlavních oken. Interně je řetězec zpracován funkcí `sscanf` a získané hodnoty jsou uloženy do kontextu knihovny.

- `-iconic` - Nově vytvářená hlavní okna budou na počátku minimalizovaná.
- `-indirect` - Vynutí použití software renderingu OpenGL.
- `-direct` - Vynutí použití hardware renderingu OpenGL. Pokud není uvedena ani jedna ze dvou předchozích hodnot, pokusí se knihovna použít rychlejší hardware rendering a v případě neúspěchu použije pomalejší software rendering. Tento i předchozí parametr nastavují položku `direct_render` v kontextu knihovny.
- `-gldebug` - Nastaví položku `gl_debug` v kontextu knihovny.
- `-sync` - Vynutí synchronizovaný režim zobrazení. Interně se převádí na parametr `--sync` a předává se knihovně GTK.

Funkce nejprve inicializuje kontext knihovny (pokud již nebyl zinicializován) a poté provede zpracování parametrů funkcí `__gtkglut_parse_params`. Nakonec jsou zavolány inicializační funkce GTK (`gtk_init`) a GtkGLExt (`gtk_gl_init`).

Zajímavá je funkce pro zpracování parametrů. Ta některé parametry zpracuje ve své režii, ovšem dva parametry předává dál, a proto dojde k uložení těchto parametrů do listu, který je na úplném konci funkce odstraněn a jednotlivé parametry jsou ze seznamu odstraněny. O odstranění parametrů se stará funkce `__gtkglut_remove_null_params`, která mění hodnotu počtu parametrů a položky NULL (což je identifikátor toho, že položka byla odstraněna) nahrazuje neprázdnými položkami (dojde k defragmentaci pole).

Za povšimnutí také stojí zpracování geometrie. V řetězci s geometrií mohou být zadána také záporná čísla počáteční pozice, což znamená posun okna od pravého dolního rohu. Ke správnému výpočtu pozice je potřeba zjistit šířku a výšku obrazovky. To je zajištěno pomocí objektu `GdkScreen`, který ovšem existuje teprve po zavolání funkce `gtk_init`. Proto je zpracování tohoto parametru prováděno nadvakrát. Nejprve je zpracována příkazová řádka a hodnoty jsou uloženy v pomocné proměnné `geometry` a teprve po zavolání `gtk_init` dojde k jejich vyhodnocení a aplikování.

Funkce `glutInitWindowPosition` a `glutInitWindowSize` nastavují počáteční pozici a rozměry nově vytvářených oken nejvyšší úrovně. Obě funkce pouze nastavují proměnnou `window_init_geometry` v kontextu knihovny. Zvláštní úlohu hraje při nastavení pozice hodnota -1. Jedná se o standardní hodnotu a znamená, že umístění okna bude ponecháno na okenním systému.

Relativně zajímavou funkcí je `glutInitDisplayMode`. Ta nastavuje zobrazovací mód, který bude použit pro nově vytvořená okna. Má jediný parametr `mode`. Ten je v uživatelském programu možné vytvořit použitím bitové operace OR na hodnoty `GLUT_RGBA`, `GLUT_RGB`, `GLUT_INDEX`,... Hodnoty jsou stejné jako v GLUT API 3. GtkGLExt používá podobné pojmenování hodnot jen nahrazuje prefix `GLUT_` za `GDK_GL_MODE_` (například `GDK_GL_MODE_RGB`, `GDK_GL_MODE_RGBA`, `GDK_GL_MODE_INDEX`,...). Největším problémem bylo, že některá makra mají definovány jiné číselné hodnoty (například `GDK_GL_MODE_STEREO` má hodnotu 4, ale `GLUT_STEREO` má hodnotu 256). Řešení tohoto problému jsou v podstatě tři:

- Makra pro knihovnu definovat stejně jako v originální GLUT knihovně a vytvořit kód, který se soustavou if testů postará o správné nastavení při vytváření OpenGL kontextu. V konečné verzi je zvoleno toto řešení. Funkce, která se stará o popsaný převod se jmenuje `__gtkglut_convert_glut_mode_to_gtkglext`.

- Makra pro knihovnu opět definovat stejně jako GLUT a vytvořit funkci která vhodnou manipulací bitů převede jeden formát na druhý. Toto je asi nejhorší řešení, protože v případě změny hodnot v GtkGLExt dojde ke ztrátě kompatibility.

- Makra pro knihovnu definovat stejně jako v GtkGLExt. Toto řešení trpí možným problémem nižší kompatibility s programy, které nepoužívají symbolická jména, ale rovnou hodnoty a samozřejmě znamená ztrátu binární kompatibility. Takřka do konce vývoje bylo zvoleno právě toto řešení. Jelikož se na konci jednalo o jedinou překážku v dosažení binární kompatibility s originální knihovnou GLUT, byla tato varianta opuštěna na úkor první.

Hlavní nekompatibilitou je nedostupnost módu luminance (`GLUT_LUMINANCE`), protože v GtkGLExt není podporován.

Interně tato funkce nastavuje proměnnou `display_mode` v kontextu knihovny. Ta je standardně nastavena na kombinaci `GLUT_RGB | GLUT_SINGLE | GLUT_DEPTH`. V mnohé literatuře se lze dočíst, že hodnota `GLUT_DEPTH` není nastavena. Není to ovšem pravda. Podle zdrojových kódů mají knihovny GLUT 3.7 i FreeGLUT 2.4.0 standardně hodnotu `GLUT_DEPTH` nastavenu.

Knihovna GLUT verze 3.4 přinesla zajímavou alternativu k předešlé metodě s podobnou funkcionalitou v podobě funkce `glutInitDisplayString`. V knihovně je použita implementace z FreeGLUT, která nabízí menší možnosti, než originální GLUT, avšak pro valnou většinu problémů je dostatečně použitelná. Hlavní nekompatibilitou je absence zpracování operátorů. Po zpracování řetězce volá předchozí funkci a její možnosti jsou tedy stejné.

Nakonec je vhodné zmínit rozšíření knihovny FreeGLUT a to funkci `glutSetOption`. V implementaci GtkGLUT je implementována jako obal pro dříve zmíněnou skupinu funkcí `glutInitDisplayMode`, `glutInitWindowSize`, `glutInitWindowPosition`. Navíc poskytuje stejnou funkcionalitu jako `glutSetCursor`. Nakonec se také jedná o jedinou možnost, jak nastavit akci, která má být provedena při uzavření okna. Tato funkce by v budoucnu mohla sloužit jako náhrada dříve zmíněných funkcí.

## 5.3 Funkce pro správu oken

Správa oken byla základní a také zřejmě nejsložitější část knihovny. Veškerá funkčnost je obsažena v souboru `gtkglut_window.c`.

Knihovna GLUT rozeznává dva typy oken a to okna nejvyšší úrovně a podokna (subwindow). Okna nejvyšší úrovně jsou spravována okenním manažerem daného grafického prostředí, mezitím co podokna jsou spravována přímo knihovnou GLUT. GTK+ přistupuje k oknům podobně, avšak bohužel ne stejně. Také zde jsou okna nejvyšší úrovně spravována okenním manažerem a podokna jsou spravována v rámci knihovny. U GTK+ platí, že ne každé podokno musí být fyzicky mapovaným oknem a takové okno nemá svůj vlastní kontext. Dalším problémem je, že GTK+ bylo navrhováno zejména pro tvorbu uživatelských rozhraní, kde se takřka vůbec neobjevují překrývající se podokna, čímž se značně liší od knihovny GLUT. Neposledním problémem byl správce rozložení. Pro usnadnění tvorby GUI poskytuje GTK+, podobně jako většina moderních knihoven, právě tento nástroj, který podokna sám rozmísťuje a mění jejich velikost dle velikosti rodičovského okna. Základní filozofie práce s GTK+ je taková, že hlavní okno se chová jako kontejner, do kterého lze vložit právě jednu komponentu. To většinou bývá právě správce rozložení, do kterého se umisťují další komponenty.

Pokud by GLUT nepodporoval podokna, práce by byla jednoduchá. Nejprve by se vytvořilo hlavní okno typu `GtkWindow`, a do něj by se vložila právě jedna komponenta, která by sloužila pro vykreslování OpenGL. Vhodnou komponentou je `GtkDrawingArea`. Tím práce byla hotová.

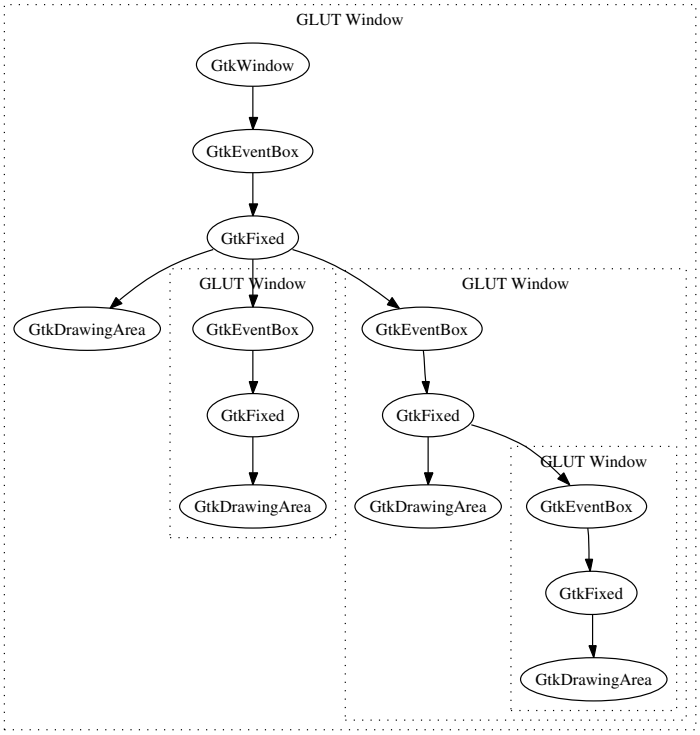
Jelikož však GLUT podokna podporuje, bylo třeba hledat jinou cestu. Nejprve bylo třeba najít vhodného správce rozložení, který by umožňoval, tak jak je požadováno GLUT funkcemi, fixní umístění podoken. Takovým správcem je `GtkFixed`. Tento je možné vložit do hlavního okna a do něj vkládat komponenty `GtkDrawingArea`. Bohužel však GLUT se chová transparentně a do podokna lze vložit další podokno, což je navrženým řešením nerealizovatelné a je nutné minulý návrh přehodnotit. Nechť je tedy každé podokno svázáno se svým fixním správcem rozložení. Toto řešení je možné a bude i víceméně funkční až na jeden detail. `GtkFixed` patří do skupiny komponent, které nemají své vlastní okno. Takže bohužel neexistuje možnost, jak nastavit výšku a šířku této komponenty. To by nemusel být problém, avšak představme si následující situaci. Vytvoříme hlavní okno, do něj vložíme podokno a do něj opět vložíme podokno, které bude větší než jeho předek. Standardní chování GLUT je takové, jak je očekáváno. Tedy poslední podokno bude oříznuto svým rodičovským oknem. Avšak v případě našeho předchozího návrhu toto fungovat nebude a okno bude oříznuto pouze hlavním oknem. Je tedy třeba najít komponentu, která bude mít své vlastní fyzicky mapované okno a do něj vkládat předchozí dvě komponenty. Tuto vlastnost takřka ideálně splňuje `GtkEventBox`. Závěrečné řešení je plně funkční a právě popsaným způsobem je správa oken zajištěna v knihovně GtkGLUT.

Pro objasnění je nejlépe uvést příklad. Cílem je vytvořit okno, v němž budou dvě podokna a v jednom z podoken bude další podokno. Celkové rozložení komponent je vidět na obrázku 5.1.

I když popsané rozložení funguje dobře, není dokonalé. Problém nastává s tzv. z-orderingem, tedy pořadím komponent. Pokud bychom se rozhodli podokno umístit nejhlouběji, dojde k tomu že by bylo překryto komponentou `GtkDrawingArea` jeho předka. Takto se GLUT aplikace nechovají, a proto je v knihovně implementováno jednoduché řešení. Pokud si uživatel vyžádá změnu pořadí oken pomocí funkce `glutPushWindow`, je vždy do pozadí přesunuta i komponenta `GtkDrawingArea` rodičovského okna. U opačné funkce `glutPopWindow` tento problém logicky nenastává.

Jedním z posledních větších problémů byla automatická změna velikosti hlavního okna dle velikosti do něj vložených komponent. Tato vlastnost je pro tvorbu většiny uživatelských rozhraní velice vhodná, jelikož stačí na začátku aplikace nastavit hlavní okno dostatečně malé, přidat do něj komponenty a okno se samo zvětší tak, aby všechny komponenty byly zobrazeny viditelné a schopné zobrazit svůj obsah. Hlavní okno také neumožní zmenšení pod hranici programátorem nastavené velikosti. Obvykle se tedy po vygenerování hlavního okna nastaví jeho velikost na aktuální velikost, čímž je zaručena minimální velikost okna taková, že jsou všechny komponenty zobrazeny. Toto chování se ovšem vůbec nehodí pro emulaci knihovny GLUT, která umožňuje tvořit podokna, která jsou vetší než hlavní okno, ale nedojde ke změně velikosti hlavního okna. Zároveň umožňuje programově nastavit velikost hlavního okna, ale okno je stále možno uživatelem zmenšit. Naštěstí GTK+ obsahuje funkci `gtk_window_set_policy`, která předchozí vlastnosti umožňuje vypnout. V knihovně je tedy touto funkcí nastavena možnost uživatelem libovolně zvětšit i zmenšit okno a zároveň je zakázána automatická změna velikosti.

Po vyřešení předchozích problému byla implementace vlastních funkcí relativně snadná a ve většině případů se jedná pouze o vhodná volání ekvivalentních GTK+ funkcí.




Obrázek 5.1: Příklad rozložení GTK+ komponent tvořící GLUT okno s podokny

## 5.4 Funkce pro správu menu

Menu je v GLUT aplikacích prakticky jediný vysokoúrovňový prvek uživatelského rozhraní. Kompletní implementace se nachází v souboru `gtkglut_menu.c`.

Ačkoli je v GTK+ podpora menu na velice vysoké úrovni, přece jen nebyla implementace právě jednoduchá. Samotný kód byl během vývoje třikrát takřka kompletně přepsán, aby splňoval požadavky na něj kladené.

První implementace byla naivní a jednalo se v podstatě pouze o transformaci GLUT API do GTK+ API. Každé menu tedy bylo vytvořeno jako objekt `GtkMenu`, do kterého byly přidávány položky. Pokud mělo být do menu přidáno podmenu, byl přidán přímo odkaz na komponentu `GtkMenu`. Vše fungovalo relativně dobře, až na dva problémy. První problém nastal, pokud některé menu bylo zároveň samostatným menu a podmenu jiného menu. Pokud došlo ke zrušení rodičovského menu ve kterém bylo vloženo podmenu, došlo zároveň i ke zrušení onoho podmenu.

Pro pochopení příčiny tohoto chování je nutno se ponořit hlouběji do mechanismů fungování knihovny GTK+. Ta obsahuje implementaci mechanismu reference counting. Ten je vhodný zejména kvůli tomu, aby uživatel nemusel uvolňovat všechny komponenty které v aplikaci vytvoří. Pokud je tedy některá komponenta přidána do jiné, například tlačítko do okna, stává se okno zodpovědné za ono tlačítko. To je dosaženo právě použitím referencí. Reference v GTK+ jsou dvojího druhu. První druh, nazývaný fluid, je určen pouze pro prvotní vytvoření komponenty, kdy je hodnota nastavena na 1. Tato reference existuje proto, aby nedošlo ke zrušení komponenty před tím, než je přidána do komponenty jiné. Druhý typ referencí jsou regulérní reference a většina komponent má při vytvoření nastavenu tuto

hodnotu na 0. Pokud však dojde k přidání jedné komponenty do jiné, je přidané komponentě zvýšen počet regulérních referencí o jedničku a počet fluid referencí je nastaven na 0. V této chvíli by již původní ukazatel na komponentu neměl být v uživatelském programu nikdy použit. Jak je vidět, v GTK+ aplikaci tedy stačí zrušit všechna hlavní okna a tím dojde ke kompletnímu uvolnění paměti.

Konečně jsme se dostali k jádru problému. Jelikož menu je také kontejner, je tedy také zodpovědný za svoje podřízené komponenty a při svém zrušení ruší i je. Náprava této chyby je relativně jednoduchá. Každé menu referencovat ve funkci pro vytvoření menu a ve funkci pro zrušení menu provést odreferencování.

Druhý problém byl mnohem závažnější a nedal se již vyřešit. GLUT podporuje vkládání jednoho menu jako podmenu do jiného menu vícekrát a také vložení jednoho menu do více jiných menu. GTK+ však vícenásobné vložení menu nepodporuje a myšlenka naivní implementace tedy musela být zavržena.

Bylo jasné, že funkce pro práci s menu musejí být realizovány ve vlastní režii pouze vhodnou manipulací datových struktur a GTK+ reprezentace menu musí být vytvořena až před samotným zobrazením. Hloupá metoda vytváření menu pokaždé při jeho vyvolání byla značně neefektivní, a proto bylo třeba zavést optimalizaci.

Ta spočívá v následující myšlence. Pokud menu od jeho posledního zobrazení nebylo změněno, není třeba jej generovat znovu a je možné použít dříve vygenerovaný objekt `GtkMenu`. Stará verze menu se ukládá v proměnné `cached_gtk_menu`. Pokud uživatel mění menu, které nemá žádné podmenu, není s implementací problém. Jednoduše se nastaví proměnná `need_refresh` udávající, že menu bylo změněno a je třeba jej přegenerovat. Pokud ovšem uživatel mění menu, které je součástí jiného menu, je třeba aby se nadřazené menu při svém vyvolání také přegenerovalo. Právě proto má každé menu položku `parent_menus`, ve které jsou uložena nadřazená menu. Pokud jsou tedy prováděny změny v menu, nedojde k nastavení proměnné `need_refresh` jen u měněného menu, ale také rekurzivně u všech nadřazených menu. K dosažení tohoto stavu je však třeba vyřešit několik věcí. Při přidávání položky menu, která je podmenu, je třeba v daného podmenu rozšířit položku `parent_menus`. Při rušení menu je naopak třeba položku z proměnné `parent_menus` odstranit. Zároveň je třeba myslet i na GLUT funkce měnící položku menu na podmenu a naopak, tedy `glutChangeToSubMenu` a `glutChangeToMenuEntry`. Zde je nutno také nakládat s proměnnou `parent_menus` obezřetně, aby nedocházelo k tomu, že nadřazené menu bude informováno o změně menu, které již není jeho podmenu, nebo naopak nebude informováno o změně svého podmenu.

Na proměnnou `parent_menus` je vhodné se podívat ještě z jiného úhlu a to na volbu datové struktury. Je jasné, že musí podporovat přidání, odebírání a průchod položkami. Navíc však musí ošetřit případ vícenásobného vložení menu do jiného menu. V tomto případě není možné uložit hodnotu jen jednou. To by mohlo vést k situaci, kdy by při odebrání jednoho podmenu přestal mechanizmus fungovat. Jiné, nepříliš efektivní řešení by bylo uložit jednu hodnotu vícekrát. V takovém případě by buď docházelo k vícenásobnému upozorňování jednoho menu, nebo by se značně ztížil kód vykonávající upozorňování nadřazených menu. Řešení, které bylo nakonec zvoleno a implementováno vychází z metody reference counting.

O celou záležitost se stará datová struktura, autorem pojmenovaná refset. Název je složen ze slov *reference* a *set*, tedy množina. Opravdu se jedná o množinu, kde každá položka obsahuje hodnotu a počet referencí na danou hodnotu. Implementace se nachází v souboru `gtkglut_refset.c`. Základní operace této datové struktury je ref, která provádí zvýšení počtu referencí a unref, která naopak provádí snížení počtu referencí. Struktura je postavena na datové struktuře seznamu. O vhodnosti této datové struktury by bylo možno diskutovat,

přesto má své opodstatnění. Co se týče rychlosti, bylo by vhodnější použít hash tabulku. Ta však zabírá relativně hodně paměti i když je velice málo zaplněná. Jelikož většina menu není z praktických důvodů příliš rozsáhlých, je zvolena pomalejší, ale paměťově efektivnější varianta seznamu.

Při provedení operace ref dojde nejprve k vyhledání položky s definovanou hodnotou. Pokud je taková položka nalezena, je zvýšen počet referencí a je vrácen ukazatel na nepozměněný počátek seznamu. Pokud položka není nalezena, je vytvořena nová položka seznamu s počtem referencí nastaveným na 1 a je opět vrácen ukazatel na počátek nového, tentokrát již změněného seznamu. Operace unref má opačnou funkci. Také ona přijímá jako parametr hodnotu, která má být ordreferencována. Tuto položku se pokouší nalézt v seznamu. Pokud není položka nalezena, je pouze vrácen nezměněný odkaz na počátek seznamu. Zajímavější případ nastává, pokud je položka nalezena. Takové položce je snížen počet referencí o jedničku. Pokud počet referencí dosáhne čísla 0, je položka úplně odebrána ze seznamu. Vrácen je vždy ukazatel na počátek seznamu.

Tím je vyřešen základní problém a zbývají dva menší problémy. Prvním je, jak se vypořádat s oficiálně v knihovně GLUT nedokumentovanou funkcí a to klávesovými akcelerátory. Ty jsou podporovány pouze ve Windows verzi knihovny. Implementace je provedena tak, že pokud se v popisu položky menu objeví znak ampersandu (`&`), je písmeno bezprostředně za ním následující zobrazeno podtržené a použito jako klávesový akcelerátor. Pro aktivaci dané položky menu tedy stačí zmáčknout klávesu s daným znakem. Pokud je program využívající této vlastnosti spuštěn v X Window, je znak ampersand zobrazen jako každý jiný znak a to příliš nepřispívá k přehlednosti. Bohužel se použití této vlastnosti hodně rozšířilo, proto je nakonec tato funkce implementována i v knihovně GtkGLUT. GTK+ používá podobný způsob jako Windows verze GLUT, avšak není použit znak ampersand, ale podtržítko (_). V GtkGLUT tedy existuje funkce `__gtkglut_convert_name_with_mnemonic`, která převádí každý výskyt znaku ampersand na podtržítko a každý výskyt podtržítka na dva znaky podtržítka, což je nutné, aby uživatel mohl zadat a zobrazit i tento znak.

Druhým menším problémem byla možnost, že se nevhodným přidáním podmenu vytvoří cyklus. To by vedlo k zacyklení funkce, starající se o vygenerování menu. Možných řešení bylo více, od inteligentních detekující tento stav grafovými algoritmy, až po prosté ignorování tohoto problému. Implementované řešení patří do skupiny méně inteligentních, zato velice jednoduchých na implementaci i pochopení. Ve funkci pro generování menu, která je volána rekurzivně, je testována hloubka zanoření. Pokud hloubka přesáhne hodnotu danou makrem `__GTKGLUT_MAX_MENU_DEPTH` (v současné verzi s hodnotou 64), nedochází k dalšímu rekurzivnímu zanoření. Hodnota 64 byla vybrána jako hodně předimenzovaná, jelikož v takto hlubokém menu se stejně zřejmě nikdo nezorientuje.

Na závěr této části je vhodné explicitně popsat, v textu několikrát zmíněnou, funkci pro generování menu. Jmenuje se `__gtkglut_regenerate_menu`, jejím parametrem je menu, které má vygenerovat a aktuální hloubka. Po vyřešení všech předchozích problémů byla její implementace relativně snadná a jedná se v podstatě jen o vhodné vytváření GTK+ objektů.

## 5.5 Funkce pro registraci a obsluhu událostí

Funkce pro registraci a zpracování událostí jsou podstatnou částí knihovny GLUT. Veškerý kód pro registraci se nachází v souboru `gtkglut_callback_reg.c` a funkce pro zpracování jsou vždy v souboru s implementací části, které se daná událost týká, ve většině případů tedy `gtkglut_window.c`.

Valná většina událostí je v GLUT i GTK+ podobná, ovšem nikoli stejná. Začněme událostmi, které se mapují identicky. Těmi jsou `glutIdleFunc`, která je vnitřně přidána pomocí funkce `g_idle_add`. Podobně snadná je implementace časovače, který je vnitřně namapován na funkci `g_timeout_add`. Obslužná funkce která volá asociovanou uživatelskou funkci je nazvána `__gtkglut_callback_timer_handler`, a vrací vždy hodnotu `FALSE`. Tím je zajištěno, že je tento časovač odstraněn z vnitřních struktur knihovny GTK+. Pokud by vrácená hodnota byla `TRUE`, byl by časovač volán kontinuálně. Registrace zobrazovací funkce patří do podobné skupiny. Tato jako jediná musí být pro každé okno asociována, jinak dojde k ukončení běhu knihovny a tím také aplikace. Vnitřně se mapuje na GTK+ událost `expose-event` objektu `GtkDrawingArea`. Funkce hlídající změnu velikosti okna je mapována na GTK+ událost `size_allocate` objektu `GtkDrawingArea`. Tato událost je mapována také pro objekt `GtkEventBox`, kde je jejím jediným účelem změnit velikost podřízené `GtkDrawingArea`. Tím je vyvolána změna velikosti tohoto objektu a zde již může být volán uživatelský kód. Jedná se o jedinou funkci, kde existuje standardní kód který je proveden, pokud není uživatelská funkce registrována.

Nyní je načase vysvětlit události, jejíchž implementace skrývala problémy. Avšak ještě předtím je třeba zmínit jedno z nejdůležitějších rozhodnutí, které přineslo zefektivnění zpracování událostí, avšak zároveň značně znepřehlednilo kód a učinilo jej složitějším na pochopení. Předchozí události GTK+ byly buď namapovány vždy, nebo byly mapovány v rámci celé knihovny. Problémem některých událostí je ovšem to, že značně zatěžují systém, jelikož mohou být generovány v obrovském množství. Příkladem takové události je pasivní pohyb myši, tedy pohyb bez stisknutých tlačítek. To není problém při lokálním použití knihovny, avšak v případě síťového provozu by mohlo docházet ke zbytečnému snížení výkonu. K tomu přesně by došlo, pokud by `GtkDrawingArea` měla zaregistrovány všechny události již při svém vytvoření a užitečnou úlohou funkcí určených ke zpracování by ve většině případů bylo jen zjistit, že uživatel nezaregistroval žádnou obslužnou funkci a ukončit se. Naštěstí GTK+ má zpracování většiny událostí standardně zakázáno a je třeba je explicitně povolit funkcí `gtk_widget_add_events`. Teprve potom začne fungovat navázání události na konkrétní funkci. Bohužel nic není dokonalé a jednou přidané události již není možné odebrat. Je však možné odebrat vazbu události na funkci. To jinými slovy znamená, že událost je sice doručena GTK+, ale dále není zpracována.

Výše popsaný mechanismus je reálně implementován. Tedy, dokud není poprvé zaregistrována událost GLUT, která vede k zaregistrování události v GTK+ a navázání události na funkci, není událost doručována a není zatěžován okenní systém. Pokud si uživatel přeje událost GLUT odregistrovat, je zrušena vazba události na funkci. Nová registrace události již vede pouze na vytvoření vazby. Aby tento mechanismus v knihovně fungoval, musí být vytvořena vhodná datová struktura, která obsáhne všechny požadavky. Takovou strukturou je již dříve popsaná `__gtkglut_signal_struct`.

Bohužel tímto není všem problémům konec. Některé GLUT události, které se v GTK+ mapují na událost jedinou, jsou v knihovně rozdělené. Typickým příkladem je zpracování vstupu z klávesnice. Zde je možnost registrovat událost pro stisknutí a puštění kláves standardních, ale také speciálních. Tuto funkcionalitu řeší v GTK+ jediná událost s názvem `key-press-event`. Řešením bylo zavedení čtyř samostatných `__gtkglut_signal_struct`, které mají společnou jedinou GTK+ funkci pro zpracování. To znamená nejen vhodně ošetřit to, aby mapování GTK+ události na funkci nebylo provedeno vícekrát než jednou, ale také aby případné odmapování bylo provedeno teprve až nebude existovat žádná GLUT událost, která by toto mapování potřebovala. Výsledkem je nakonec nepříliš přehledný kód `__gtkglut_keyboard_func`. Podobný problém se týká i událostí myši při registraci funkce

pro zpracování pohybu. Na druhou stranu, při zpracování stisknutí a uvolnění tlačítek myši nastává opačný problém. GTK+ má pro tento účel dvě odlišné události, ale GLUT jen jednu. S obsluhou tlačítek myši souvisí ještě jeden problém. Obsluhu těchto událostí si uživatel může vyžádat nejen explicitně, ale dochází k ní i implicitně při zaregistrování menu na některé z tlačítek. Všechny případy jsou řešeny společně, čímž se kód stal opět relativně nepřehledným. Odregistrování události je podobné a je tedy potřeba nejen otestovat uživatelem registrované funkce, ale také menu asociované s oknem a teprve pokud již nemůže událost nastat, je provedeno odregistrování. GTK+ obsluha události je již relativně jednodušší, jelikož stačí zjistit, které z tlačítek bylo stisknuto a zda je k tomuto tlačítku asociováno menu. Pokud ano, je menu vygenerováno a zobrazeno, jinak je volána uživatelská funkce, pokud je taková zaregistrována.

Vraťme se však zpět ke zpracování událostí generovaných klávesnicí. Zde se skrývá asi poslední vážnější problém a to focus. V okenních systémech platí, že vstup klávesnice je posílán právě oknu, které má focus. U oken nejvyšší úrovně se o tuto vlastnost stará okenní manažer a knihovna ji nemůže prakticky vůbec nijak ovlivnit. Okenní manažer se bohužel nestará o focus podoken a událost klávesnice je stále posílána oknu nejvyššímu. To však většinou není to, co je od knihovny pro tvorbu uživatelských rozhraní vyžadováno a taková knihovna si focus mezi jednotlivými komponentami musí řešit ve svojí režii. Nejinak je tomu i u GTK+. Rozhodování, která komponenta má mít focus řeší více přístupů, ale popsány budou dva nejčastější. První přístup klade největší důraz na myš. Focus má ta komponenta, pod kterou se nachází ukazatel. Tato metoda je historicky starší a je použita například u grafické knihovny Athena. Pro valnou většinu uživatelů je toto chování velice nezvyklé a dnes se již takřka nevyskytuje. Druhá metoda je obvyklejší a jejím základem je možnost určit aktivní komponentu kliknutím myši, nebo speciální klávesou typu tabulátor. Přesně toto chování je implementováno například v uživatelském rozhraní Windows, ale také v GTK+. U obou přístupů existuje možnost nastavit focus také programově. Jelikož je GLUT knihovna již relativně stará a vznikla pro operační systém IRIX s X Windows systémem, implementuje metodu první.

V knihovně GtkGLUT je toto chování emulováno za pomoci interní obsluhy události `enter-notify-event` komponenty `GtkDrawingArea`. Tato událost nastává, pokud uživatel přesunul kurzor myši z jedné komponenty do jiné komponenty. Zpracování je jednoduché. Komponentě které byla tato událost zaslána je programově přidělen focus. Tím je zajištěno chování stejné jako u originální knihovny GLUT.

Za zmínku stojí také implementace funkcí `glutVisibilityFunc` a modernější varianty `glutWindowStatusFunc`. Ty jsou registrovány samostatně, jejich účel je však podobný. Po zaregistrování uživatelské funkce je tato volána vždy při změně viditelnosti okna. První uvedená funkce je starší, má méně možností a v návrhu GLUT 4 API bylo její úplné odstranění. Nyní se však podívejme na vlastní implementaci. Funkce je vnitřně mapována na GTK+ událost `visibility-notify-event` komponenty `GtkDrawingArea`. Bohužel, toto mapování bude fungovat jen částečně. Událost je poslána při skrytí nebo odkrytí okna jiným oknem, ale bohužel není volána při prvotním zobrazení a ikonifikaci okna. Tento problém je řešen na úrovni hlavního okna registrací události `window-state-event`. Ta je využívána zejména v případě ikonifikace okna. Tím by ovšem problém nebyl kompletně vyřešen. Hlavní okno sice dostane událost o svém skrytí, ovšem jeho podokna již ne. Proto je v tomto případě volána obsluha i pro všechna podokna s parametrem udávajícím, že došlo ke skrytí. U opačné operace, tedy deikonifikace tento postup potřeba není, jelikož událost `visibility-notify-event` je v takovém případě zaslána jak hlavnímu oknu, tak všem jeho neskrytým podoknům.

Zpracování ostatních událostí již neskrývá žádné větší problémy a tudíž je vhodné tuto oblast uzavřít.

## 5.6 Funkce pro vykreslení textu

Vykreslování znaků a textů je velice používaná funkcionalita, a proto její implementace nemohla být opomenuta. Všechny funkce se nachází v souboru `gtkglut_font.c`.

V originální knihovně GLUT jsou k dispozici dva druhy písem, bitmapová a vektorová. Uživateli je vždy k dispozici 7 písem bitmapových, z čehož je jeden font typu sans-serif k dispozici ve 3 velikostech, písmo typu serif ve 2 velikostech a existují také 2 velikosti písma typu monospace. Dalším typem jsou písma vektorová, která jsou plně škálovatelná a jsou k dispozici ve dvou variantách s fixní a variabilní šířkou. Oba typy písem sdílí společnou slabinu a to neexistenci podpory pro jiné znaky než ASCII, a to ještě ne vždy kompletní. Při vývoji GtkGLUT bylo třeba rozhodnout, jak se k dané oblasti postavit. První, zřejmě nejlepší metodou by bylo vykreslovat fonty pomocí GTK+ knihovny Pango. Ta je postavena na knihovně FreeType a bylo by tedy možné použít mnoho druhů písem včetně TrueType, vše s podporou hintingu a vyhlazování. Největším problémem tohoto řešení je, že nikdy nelze přesně určit, jaké fonty budou u uživatele k dispozici. Tím by se knihovna značně odlišila od GLUT, protože zde je vždy k dispozici výše popsaných 9 písem. Taková implementace by tedy nebyla plně kompatibilní. Samotné použití knihovny Pango navíc není příliš triviální.

Jinou možností by bylo použít knihovnu Pango nepřímo za pomocí GtkGLExt. Ta poskytuje funkci `gdk_gl_font_use_pango_font`, která umí vygenerovat display list se znaky, které jsou funkci předány jako parametr. Největším problémem však je právě ono generování do display listu. V nejhorším případě by bylo třeba vygenerovat kompletní znakovou sadu unicode, což by znamenalo obrovskou spotřebu paměti. Navíc první problém knihovny Pango zde stále zůstává.

Nakonec je tedy zejména z důvodů zachování maximální kompatibility použito řešení převzaté z knihovny FreeGLUT. Samotný výkonný kód je v podstatě totožný, jen jsou upraveny některé definice a pojmenovací konvence dle knihovny GtkGLUT. Implementace byla vybrána také z důvodu relativní jednoduchosti výkonného kódu. Bohužel toto rozhodnutí sebou nese i některé nepříjemnosti. První je, že veškerá data fontů jsou vloženy do zdrojových souborů `gtkglut_font_data.c` pro bitmapová a `gtkglut_stroke_mono_roman.c`, `gtkglut_stroke_roman.c`, pro vektorová písma. Tím dochází k relativně nepříjemnému nárůstu velikosti knihovny. Druhým problémem je stejně jako v GLUT chybějící podpora pro jiné znaky než ASCII.

## 5.7 Neimplementované funkce

Na závěr popisu funkcí je vhodné uvést ty, které zůstali neimplementovány a zejména vysvětlit, proč tomu tak je. Veškerý kód se nachází v souborech `gtkglut_unimplemented.c` a `gtkglut_game_mode.c`. Každá funkce je kvůli zachování binární kompatibility knihovny fyzicky přítomna, avšak jedinou akcí kterou provádí jé výpis varovného hlášení.

První skupinou je podpora OpenGL overlay. Tato vlastnost není přítomna v knihovně GtkGLExt a obvykle není v programech příliš používána. Zajímavou implementaci má knihovna FLTK, která v případě nepřítomnosti této vlastnosti provádí emulaci a to tak, že vytvoří dvě samostatná okna, z nichž jedno představuje standardní kreslící plochu a druhé,

s průhledným pozadím, overlay. Pak se jen dle aktivně nastavené kreslící plochy určí, kam budou posílány OpenGL příkazy. Jelikož však v takovém případě dochází k nepříjemnému přeblikávání a implementace overlay v GtkGLExt má relativně vysokou prioritu, rozhodl se autor raději počkat na novou verzi.

S podporou overlay úzce souvisí i indexovaný barevný mód. Overlay OpenGL kontext totiž může být nastaven pouze na tento barevný mód. GTK+ má podporu pro indexovaný mód, avšak značně odlišný od toho, co požaduje GLUT API. Tím se tato vlastnost stala při zachování platformové nezávislosti neimplementovatelnou.

Vzhledem k době vzniku knihovny GLUT obsahuje podporu pro mnohá vstupní zařízení, která jsou dnes velice neobvyklá. Příkladem budiž spaceball, button box, ... Podpora v knihovně GtkGLUT není obsažena, jelikož tato zařízení nejsou přímo podporována v GTK+. Bohužel GTK+ nepodporuje ani dnes velice rozšířené herní zařízení joystick. Zároveň není zahrnuta ani přímá podpora pro tablet. Toto zařízení se ovšem v systému jeví jako standardní myš, takže je možné jej používat, avšak specifické funkce k dispozici nejsou. Z funkcí pro vstupní zařízení není implementována ani podpora pro nastavení automatického opakování kláves. Tato funkce je sice v GTK+ dostupná, avšak její použití není doporučeno, protože mění nastavení nejen pro vlastní aplikaci, ale celého systému.

Mezi poslední neimplementované funkce patří možnost změny rozlišení a herní mód. Herní mód původně sloužil pro zrychlení vykreslování, avšak v dnešní době již poněkud ztratil smysl. Změna rozlišení je platformově velice specifická věc a v knihovně GTK+ není vůbec podporována.

I když seznam neimplementovaných vlastností může na první pohled vypadat nepříjemně, většina aplikací tyto funkce nepoužívá, nebo je schopna pracovat správně i bez jejich podpory.

## 5.8 Kompatibilita

Jelikož již všechna důležitá rozhodnutí při vývoji knihovny byla popsána, nezbývá než uvést několik slov o kompatibilitě s originální knihovnou GLUT.

Prvotním cílem při vývoji bylo dodržet kompatibilitu s GLUT 3 API a implementovat užitečné funkce, které přineslo FreeGLUT API. V průběhu vývoje se však ukázalo jako reálné implementovat nejméně prototypy všech funkcí z knihoven GLUT 3.7 a FreeGLUT 2.4.0. Tím je zajištěna kompatibilita nejen na úrovni API, ale také ABI. Knihovnu je tedy možné použít jako vhodnou náhradu pro aplikace, které jsou dynamicky linkovány s originální knihovnou GLUT. Avšak ani ABI kompatibilita ještě nutně nemusí znamenat, že chování knihoven je totožné. Jako vhodný test kompatibility posloužily originální testy z distribuce knihovny GLUT 3.7, uložené v adresáři `test/glut`. Stručné lze hlavní rozdíly v chování shrnout do následujících bodů:

- Neexistuje implementace funkcí popsaných v části 5.7 na straně 28.
- GLUT znovu používá ID menu a oken, která byla zrušena. GtkGLUT se takto nechová a přiřazuje vždy nová ID. Tato vlastnost není nikde popsána a aplikace by na ní neměly být závislé, bohužel tomu tak není vždy.
- GTK+ může posílat více žádostí o překreslení. Tato vlastnost je silně závislá na použitém okenním systému a konkrétní verzi GTK+ a v některých případech k ní nedochází.

- Proporce fontů převzatých z FreeGLUT se v některých případech o několik málo pixelů liší.
- Události všech typů mohou přicházet v jiném pořadí, než u originální knihovny GLUT. Tento fakt je obzvláště patrný u události zobrazení nebo skrytí okna.
- GLUT používá seskupování požadavků na změnu velikosti, umístění a zobrazení okna. Pokud je tedy voláno například skrytí okna, zobrazení okna a přesun na jinou pozici, je proveden pouze přesun na jinou pozici. GtkGLUT toto chování nepodporuje a vše provádí okamžitě. To může znamenat příchod událostí, které by v GLUT nikdy nepřišly.

Implementovaná knihovna nesplní 10 testů z 28, vždy z některého z důvodů popsaných výše. Byť toto číslo nevypadá nejlépe, v běžných korektně napsaných aplikacích nepozná programátor a uživatel rozdíl v chování mezi GLUT a GtkGLUT.

## 5.9 Uspořádání a sestavení knihovny

Zdrojový kód knihovny se nachází na CD v souboru `gtkglut-0.3.7.tar.bz2`. Po rozbalení archivu vznikne adresář `gtkglut-0.3.7`. Jak již název vypovídá, knihovna se nachází ve verzi 0.3.7. Toto označení má naznačit kompatibilitu s originální GLUT knihovnou 3.7 a zároveň, že kód zatím není autorem považován za produkčně stabilní.

Veškeré zdrojové kódy funkcí se nachází v souborech `gtkglut_*.c` v podadresáři `src`. V každém souboru jsou implementovány vždy všechny funkce dané třídy. Hlavním důvodem pro toto rozdělení je snažší orientace ve zdrojových kódech a umožnění linkeru odstranit nepotřebný (nikdy nepoužitý) kód v případě statického linkování. Za povšimnutí také stojí soubor `gtkglut_internal.h`, obsahující definice interních proměnných, struktur, maker a funkcí. Tento soubor by v uživatelském programu neměl být nikdy používán.

V podadresáři `include/GL` se nacházejí hlavičkové soubory s prototypy jednotlivých funkcí. Adresář obsahuje 4 soubory, ale pro uživatelské použití knihovny jsou určeny pouze soubory `glut.h` a `gtkglut.h`. Účelem těchto souborů je pouze vhodně vložit obsah souborů `gtkglut_std.h` (GLUT 3.7 API) a `gtkglut_ext.h` (rozšíření zavedená v FreeGLUT 2.4.0). Pro využití všech dostupných funkcí je vhodné používat soubor `gtkglut.h`. Pokud je naopak požadována maximální zpětná kompatibilita je vhodnější použít soubor `glut.h`.

Podadresář `progs` obsahuje testovací programy. V první řadě se jedná o zdrojové kódy z klasické knihy OpenGL Programming Guide, tzv. The Red Book (do češtiny přeloženo jako [2]). Dále jsou obsaženy programy z adresáře `demos` knihoven FreeGLUT a OpenGLUT. Nakonec jsou v podadresáři `tests` obsaženy nepozměněné testovací programy z originální knihovny GLUT verze 3.7. Tyto testy sloužily jako hlavní měřítko kvality implementace knihovny.

V podadresáři `doc` je umístěn konfigurační soubor nástroje doxygen, sloužícího k vytvoření dokumentace ze zdrojových kódů knihovny. Dokumentovány způsobem vhodným pro doxygen jsou pouze funkce veřejného API, interní funkce se tedy ve vygenerované dokumentaci neobjeví. Dokumentace byla převzata a dle potřeb upravena z knihovny OpenGLUT. Po vygenerování má uživatel takřka kompletní náhradu za originální GLUT 3 API dokumentaci.

Posledním podadresářem je `win32`, kde jsou umístěny soubory specifické pro MS Windows verzi knihovny. Jedná se zejména o soubor `gtkglut.nsi`, který obsahuje kompletní

skript pro vytvoření instalačního balíčku za pomocí programu NSIS. Dále jsou zde umístěny šablony projektů pro vývojové prostředí Dev-C++.

K sestavení knihovny je třeba mít na počítači nainstalovány následující programy:

- Kompilátor jazyka C, ideálně ve verzi GNU, tedy gcc.
- Sestavovací program Make, opět v GNU verzi.
- Knihovnu GTK+ včetně podpory pro vývoj, tedy hlavičkové soubory.
- Nástavbu GTK+ pro práci s OpenGL, GtkGLExt.
- Pro vygenerování konfiguračního souboru nástroje pkg-config makroprocesor M4.
- Pokud je požadavkem možnost vytvořit dokumentaci, tak i nástroj Doxygen.

Samotný sestavovací proces je tvořen buď jen předpřipraveným souborem `Makefile` a nebo, jak to bývá u svobodných programů zvykem, je k dispozici skript `configure`. Předpřipravený `Makefile` se v některých situacích může chovat mírně odlišně od toho, který je vygenerován pomocí skriptu `configure`, avšak základní funkčnost je stejná. V průběhu sestavování se v obou případech používá nástroj pkg-config, který umí vrátit vhodné parametry kompilátoru pro úspěšnou kompilaci. Dále bude popisován předpřipravený `Makefile`.

Pokud jsou splněny všechny předpoklady, dojde po spuštění programu `make` ke zpracování cíle `default` a vytvoření knihovny GtkGLUT ve statické i dynamické verzi. Navíc je vygenerován také konfigurační soubor pro program pkg-config. Programu je možné zadat parametr `PREFIX`, který určuje, kam bude knihovna nainstalována a kde se tedy mají hledat jednotlivé soubory. Nejedná se však o určení, kam dojde k faktické instalaci, to je určeno parametrem `DESTDIR`. Standardně je nastavena hodnota `/usr/local`.

Pokud je jako parametr zadán cíl `all`, dojde navíc k sestavení testovacích programů se statickou verzí knihovny. Hlavní výhodou tohoto přístupu je možnost programy vyzkoušet ještě před nainstalováním knihovny. Tohoto výsledku by bylo možné dosáhnout i při použití dynamické knihovny vhodným nastavením proměnných prostředí (například v systému Linux `LD_LIBRARY_PATH`), nicméně předchozí řešení je uživatelsky přívětivější.

Instalaci je v unixově založených systémech možné provést zadáním cíle `install`. Pokud je požadována instalace dokumentace, je možné použít cíl `install-doc`. Pro instalaci knihovny, která slouží jako náhrada za originální GLUT knihovnu, je určen cíl `install-compat`. Tato volba je vhodná zejména v systémech, kde není knihovna GLUT přítomna a zároveň je nutné spouštět již zkompilované binární programy. Všechny instalační cíle přijímají parametr `DESTDIR`. Ten určuje, kam má být knihovna opravdu nainstalována. Standardně je opět nastaven na `/usr/local`. Rozdělení `PREFIX` a `DESTDIR` je vhodné zejména pro tvůrce balíčků.

Instalaci pro systémy Windows je nejlépe provést předpřipraveným instalačním programem, který nejen nakopíruje knihovny na vhodná umístění, vytvoří zástupce v nabídce Start, ale navíc obsahuje i předpřipravené šablony pro nástroj Dev-C++.

Po instalaci je možné knihovnu začít plně využívat k tvorbě nových programů. V systému Windows je nejlépe použít již dříve zmiňovaný nástroj Dev-C++ s nainstalovanou šablonou, která obsahuje všechny potřebné konfigurační volby překladače. V unixových systémech je možné pro zjištění potřebných nastavení překladače použít nástroj pkg-config. Pokud například chceme sestavit soubor `test.c`, je to možné provést takto:

```
gcc ‘pkg-config --libs --cflags gtkglut-0.3‘ test.c
```

# Kapitola 6

# Budoucí vývoj

Jak bylo patrné z předchozího popisu, je v současné době knihovna na velmi dobré cestě stát se kompletní náhradou originální knihovny GLUT. Hlavními výhodami takového kroku by bylo definitivní vyřešení licenčních problémů a také celkově jednodušší kód knihovny. K tomu by však bylo třeba implementovat dosud neimplementovanou funkčnost a zároveň zajistit intenzivní testování. Jako vhodný zdroj neimplementovaného kódu se jeví knihovna FreeGLUT (zejména podpora joysticku a změna rozlišení). Takto by se knihovna dostala do verze 1.0, kde by došlo k zmražení kódu (tedy zákaz přidávání nových vlastností) a už by se pouze opravovaly případné chyby.

Po tomto vydání by bylo vhodné založit novou verzi (například 2), kde by došlo k pročištění, zjednodušení a přidání nové funkčnosti do API. Tím by se knihovna stala zpětně nekompatibilní a právě proto byl kladen důraz na stabilní verzi 1.0. Ta by sloužila stále pro starší programy. Pro nové programy by byla možnost výběru. Jednalo by se tedy o model, který je často používán svobodnými knihovnami, například GTK+, QT, ...

Otázkou zůstává, zda by se knihovna měla vydat cestou pro jednodušší tvorbu her, nebo profesionálních aplikací (typu CAD). Jelikož se autor ve specifických požadavcích kladených na herní knihovny příliš neorientuje, budou dále popsány spíše rozšíření vhodná pro profesionální aplikace.

Je třeba poznamenat, že v oblasti návrhů nového API již bylo mnoho práce uděláno v rámci projektu OpenGLUT (viz [8]). Jako příklad je možné uvést plná podpora kódování unicode. To je již částečně rozpracováno díky tomu, že GTK+ je plně UTF-8 knihovna. Z toho plyne, že všude, kde je prováděno kreslení textů právě skrze GTK+, je vstupní řetězec očekáván v tomto kódování. Nejvíce viditelná je tato možnost při vytváření menu. Pokud se však uživatel rozhodne kreslit přímo do OpenGL plochy, není unicode podporováno. To by mohla vyřešit podpora knihovny Pango, která je v GTK+ standardně obsažena. Tím by byla zaručena nejen dostupnost unicode, ale navíc také podpora většiny typů písma včetně TrueType s možností vyhlazování a hintingu.

Změny by však měly být provedeny nejen v oblasti výstupu, ale také vstupu. Ideálním řešením by bylo odstranit stávající funkce pro vstup z klávesnice a nahradit je jednou funkcí, která by předávala GDK scancode klávesy s příznakem, zda byla klávesa stisknuta, nebo puštěna. Toto řešení by bylo vhodné i pro hry. Nynější stav je takový, že není možné rozpoznat, zda byla například klávesa 1 stisknuta v alfanumerickém bloku klávesnice, nebo v numerickém. GDK však toto rozlišuje. Dále by bylo velice vhodné zavést podporu vstupních metod. Tím by bylo umožněno zadávat například české znaky za pomocí mrtvých kláves. U GTK+ je tato technologie velice dobře podporována použitím objektu `GtkIMContext`.

Předchozí návrhy řeší největší problémy knihovny GLUT. Je však třeba si uvědomit,

že GTK+ nabízí mnohem větší možnosti. Jako malá ukázka je v současné verzi knihovny implementována možnost načítat obrázky ze souboru funkcí `glutReadImageFile`. Uživatel tím získává za relativně nicotnou cenu, kterou je závislost na relativně velké knihovně GTK+, podporu nejrozšířenějších typů obrazových formátů (zejména JPG, PNG, ...), to vše implementováno na necelých padesát řádků kódu. Jako velice vhodné se jeví zavedení podpory standardních dialogů. Dnešní stav je takový, že GLUT aplikace používají k tomuto účelu různé nástavbové knihovny (například GLUI). Tyto řešení mají hlavní nevýhodu v tom, že příliš nerespektují vzhled, ale zejména chování daného uživatelského rozhraní. GTK+ je v tomto ohledu mnohem dále a byť je jeho domovem operační systém Linux s X Window systémem, nemusí při vhodném tématu vzhledu uživatel MS Windows rozpoznat GTK+ aplikaci od nativní Win32 API. Vlastní funkce by neměli být příliš složité na použití. To by zcela jistě znamenalo podporu jen části možností, nicméně pokud bude některá aplikace potřebovat specifické vlastnosti, je možné použít rovnou volání GTK+.

# Kapitola 7

# Závěr

Závěrem je vhodné stručně shrnout úspěchy, kterých bylo během diplomové práce dosaženo.

Nejprve několik málo slov o knihovně GLUT. O jejím úspěchu a kvalitním návrhu nelze pochybovat. Mark J. Kilgard vytvořil dílo, které přetrvalo dodnes a stále je používáno. To vše i přesto, že existuje množství vyšších knihoven pro tvorbu grafického uživatelského rozhraní, které mají mnohem širší spektrum působnosti a jsou považovány za profesionální. Jejich podpora OpenGL je minimálně stejně dobrá jako u GLUTu, jednoduchostí použití samotného OpenGL se příliš neliší, a přesto se jednoduché programy stále píší právě v knihovně GLUT.

V této práci popisovaná a implementovaná knihovna GtkGLUT může posloužit jako kompletní náhrada za knihovnu GLUT. Lze tedy bez nadsázky říci, že zadání práce bylo splněno. Zároveň se však, zejména díky relativně jednoduchému kódu a svobodné licenci MIT, nabízí možnost využít knihovnu jako most mezi jednoduchým GLUT a složitějšími knihovnami. Je tedy možné použít některé její části ve vlastním programu, který bude postaven na GTK+. Vývoj nové aplikace by mohl vypadat například tak, že programátor napíše výkonné jádro aplikace v API GLUT. Potom se některé části knihovny GtkGLUT přepíší a aplikace se rozšíří například o podporu klasických roletových menu, dialogů pro výběr souborů, ... čímž se z původně jednoduché aplikace může stát takřka profesionální produkt, který bude mít znatelně vyšší možnosti.

Vlastní zpráva může poslouží jako vhodný úvod do základních principů knihovny GLUT (viz kapitola 2 na straně 3), relativně rozsáhlý seznam knihoven pro tvorbu grafického uživatelského rozhraní s popisem jednotlivých vlastností (viz kapitola 3 na straně 5) a nezajímavý jistě není ani návrh vlastní knihovny (viz kapitola 4 na straně 12). Kapitola 5 na straně 15 poslouží zejména pro snadnější proniknutí do nitra implementované knihovny.

Co se týče budoucího vývoje, jedná se zejména o pokračování přechodu na kódování unicode a odstranění zastaralých funkcí z API. Podrobněji je tato problematika popsána v kapitole 6 na straně 32.

Pro zájemce o hlubší pochopení programování grafických 3D aplikací je velice vhodné prozkoumat citovanou literaturu, obzvláště o knihovně OpenGL [2], GLUT [1] a v neposlední řadě GTK+ [5].

# Literatura


[1] Mark J. Kilgard. *The OpenGL Utility Toolkit (GLUT) Pragramming Interface*. Silicon Graphics Inc., 1996.

[2] Dave Shreiner, Mason Woo, Jackie Neider, and Tom Davis. *OpenGL Průvodce programátora*. Computer Press, 2006. ISBN 80-251-1275-6.

[3] WWW stránky. FLTK Documentation. `http://www.fltk.org/documentation.php`.

[4] WWW stránky. FOX Toolkit Documentation. `http://www.fox-toolkit.org/doc.html`.

[5] WWW stránky. GTK+ Features. `http://www.gtk.org/features.html/`.

[6] WWW stránky. GtkGLExt Reference Manual. `http://gtkglext.sourceforge.net/reference/gtkglext/`.

[7] WWW stránky. Open Motif Documentation Supplement. `http://www.opengroup.org/openmotif/docs/`.

[8] WWW stránky. OpenGLUT API Proposals. `http://openglut.sourceforge.net/group__proposals.html`.

[9] WWW stránky. OpenGLUT API Reference. `http://openglut.sourceforge.net/group__api.html`.

[10] WWW stránky. Qt in Depth. `http://trolltech.com/products/qt/indepth`.

[11] WWW stránky. wxWidgets: A portable C++ and Python GUI toolkit. `http://docs.wxwidgets.org/stable/`.