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

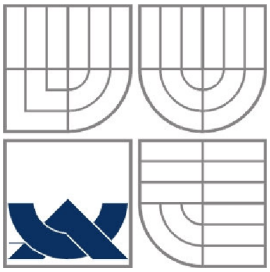
KNIHOVNA PRO PRÁCI S GRAFIKOU SE ZAMĚŘENÍM NA VÝUKU

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

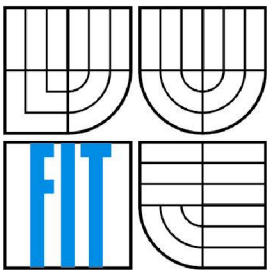
AUTOR PRÁCE
AUTHOR

MARTIN MARUŠIČ

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

KNIHOVNA PRO PRÁCI S GRAFIKOU SE ZAMĚŘENÍM NA VÝUKU

GRAPHICAL LIBRARY FOR EDUCATIONAL PURPOSES

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN MARUŠIČ

VEDOUCÍ PRÁCE
SUPERVISOR

ING. VÍT ŠTANCL

BRNO 2008

ZADANÍ BP

LICENCE A

LICENCE B

Abstrakt

Práce se zabývá návrhem a vývojem grafické knihovny vhodné pro výuku programování na středních školách a ukazuje použití této knihovny pro vizualizaci algoritmu na funkci sinus. Dále popisuje knihovny pro práci s grafickým výstupem a srovnává dobře známé multimediální knihovny.

Klíčová slova

Grafická knihovna, OpenGL, Glut, C++, OOP, výuka.

Abstract

This thesis is focused on projecting and developing graphical library for educational purposes at high schools. And shows usage of this library for visualization of sinus function algorithm. Further describes graphical libraries and compares well known multimedia libraries.

Keywords

Graphic library, OpenGL, Glut, C++, OOP, education.

Citace

Martin Marušič: Knihovna pro práci s grafikou se zaměřením na výuku. Brno, 2008, bakalářská práce, FIT VUT v Brně.

Grafická knihovna pro práci s grafikou se zaměřením na výuku

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Víta Štancla
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jméno Příjmení
Datum

Poděkování

Děkuji Ing. Vítovi Štanclovi za trpělivost a cenné připomínky.

Obsah

Obsah 1
1 Úvod 3
2 Grafický výstup 4
2.1 Grafická knihovna 4
2.1.1 Definice 4
2.1.2 Struktura 4
2.2 OpenGL 4
2.2.1 Jak OpenGL pracuje? 5
2.2.2 Co neobsahuje? 5
2.2.3 Glut 5
3 Návrh a implementace 7
3.1 Požadavky 7
3.2 Návrh knihovny 7
3.2.1 Objektový model knihovny 8
3.3 Vlastnosti grafické knihovny GL2D 8
3.4 Třída GL2D 9
3.4.1 Window 9
3.4.2 Keyboard 10
3.4.3 Mouse 10
3.4.4 MenuManager 10
3.5 Třída Object 10
3.6 Polygon 11
3.6.1 Circle 11
3.7 Image 12
3.7.1 TGA 12
3.8 Texture 12
3.8.1 Textury v OpenGL 13
4 Použití grafické knihovny 14
4.1 Instalace 14
4.2 Náš první program 14
4.3 Vytvoření okna aplikace 14
4.4 Vykreslení grafických objektů 16
4.5 Použití myši a klávesnice 17
4.6 Vyskakovací menu 18

4.7 Použití textur 18
5 Multimediální knihovny 20
5.1 Vybrané multimediální knihovny 20
5.1.1 DirectX 20
5.1.2 SDL 21
5.1.3 SFML 21
5.2 Porovnání 22
5.3 Výhody knihovny GL2D 22
6 Závěr 23
Literatura 24
Seznam příloh 25
Příloha 1 26

1 Úvod

Grafický výstup je pro člověka, který chce začít s programováním, asi to nejatraktivnější. Na středních školách, které nejsou přímo zaměřeny na výuku programovaní nebo grafiky, je to velký problém, protože není mnoho freeware grafických knihoven, které by byly snadné na použití a pochopení. Například grafická knihovna OpenGL obsahuje přes 250 různých funkcí, kterými lze vykreslit složité 3D scény.

Proto bylo cílem mé práce navrhnout a implementovat grafickou knihovnu pro práci s 2D grafikou. Knihovna by měla být přenositelná a vhodná pro výuku programování na středních školách. Díky objektově orientovanému návrhu je rozhraní knihovny snadno použitelné už po krátkém seznámení se s tvorbou objektů a jejich funkcemi.

V následující kapitole se seznámíme s grafickou knihovnou OpenGL a její okenní nadstavbou Glut. Třetí kapitola popíše návrh a implementaci mé grafické knihovny s pracovním názvem GL2D. Ve čtvrté kapitole se naučíme krok po kroku pracovat s knihovnou a využít její funkčnosti. V poslední kapitole bude zmínka o dalších volně dostupných multimediálních knihovnách a o možnostech rozšíření knihovny GL2D.

2 Grafický výstup

Počítačová grafika je dnes velmi rychle se rozvíjející obor. Tento rychlý vývoj je převážně díky moderním počítačovým 3D hrám, počítačem podporovanému projektování (CAD systémy), vizualizací v medicíně, virtuální realitě či využití pro vojenské účely.

2.1 Grafická knihovna

2.1.1 Definice

Knihovna je kolekcí tříd, funkcí a datových typů použitelných pro vývoj programů. Knihovna poskytuje aplikační programátorské rozhraní (API), které umožňuje programu volat funkce poskytované touto knihovnou.

2.1.2 Struktura

Při návrhu programu je vhodné rozdělení funkčnosti do jedné či více knihoven. Nejen, že se zvýší přehlednost programu, ale tyto knihovny jsou lehce znovu použitelné v dalších programech. Knihovna zapouzdřuje svojí komplexní funkčnost, která je dostupná přes jednoduché a snadno pochopitelné rozhraní.

Existuje mnoho typů knihoven, jako např. knihovna pro práci s grafikou, matematickými výpočty, síťovými službami, souborovými službami operačních systémů, atd. V dalších kapitolách se blíže seznámíme s některými grafickými knihovnami.

2.2 OpenGL

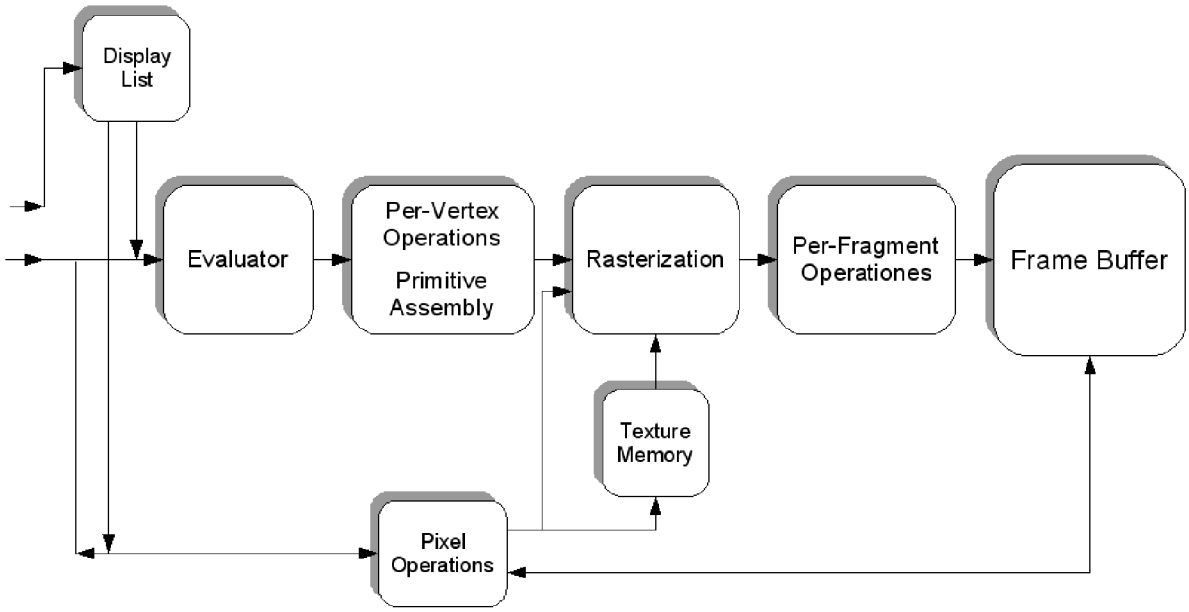
OpenGL je softwarové rozhraní ke grafickému hardwaru. V podstatě je to 3D grafická knihovna, která je velmi dobře přenositelná a velmi rychlá (využívá akcelerovaný grafický hardware). Použitím OpenGL lze vytvořit rozsáhlé 3D scény s výbornou vizuální kvalitou.

Tato knihovna byla vytvořena firmou Silicon Graphics, Inc. (SGI) v roce 1992. Hlavní výhodou je, že dokáže pracovat s mnoha různými grafickými hardwary na různých systémech (např. se systémy Windows od firmy Microsoft, Unix, Linux, Mac OS a dalšími) a dokonce je použitelná i na systémech bez grafického hardwaru, v tom případě se použije softwarová simulace. Více se o OpenGL můžete dozvědět v [3] a [6].

2.2.1 Jak OpenGL pracuje?

OpenGL je procedurální grafické rozhraní. Programátor popisuje kroky k vytvoření požadované scény. Tyto kroky se skládají z mnoha volání příkazů OpenGL, které se používají k vykreslení grafických primitiv, jako jsou body, čáry a mnohoúhelníky v trojrozměrném prostoru.

OpenGL obsahuje i osvětlení, stínování, mapování textur, blending (míchání), průhlednost a mnoho dalších speciální efektů.



Obrázek 1 – Zjednodušený vykreslovací řetězec OpenGL (převzato z [6])

2.2.2 Co neobsahuje?

Knihovna OpenGL neobsahuje žádné funkce pro práci s okny, interakci s uživatelem a ani pro práci se soubory. Tyto funkce nejsou obsaženy kvůli přenositelnosti, protože každý systém má pro tyto účely své vlastní funkce.

Neexistují žádné OpenGL souborové formáty pro popis modelů nebo virtuálních prostředí. Každý programátor si tyto formáty vytváří sám tak, aby vyhovovaly potřebám jeho aplikace, s použitím příkazů OpenGL.

2.2.3 Glut

Knihovna Glut (OpenGL Utility Toolkit) je doplňkem OpenGL, která poskytuje funkce pro práci s okny, vytváření grafického uživatelského rozhraní (GUI) a zpracování událostí. Tyto funkce jsou systémově závislé a před vznikem knihovny Glut se musely programovat pro každý systém, na kterém měla konkrétní aplikace běžet. To vyžaduje podrobnou znalost funkcí daného operačního systému.

Glut vytvořil Mark J. Kilgard, v době, kdy pracoval v SGI. Vydal také knihu o OpenGL a napsal mnoho ukázkových programů. Více o něm viz [5]. Originální knihovna Glut podporovala X Window systém (GLX) a byla upravena i pro systémy Windows (WGL) Natem Robinsem.

Cílem knihovny Glut je vytváření přenositelných aplikací (Glut je multi-platformní) a zjednodušení se učení OpenGL. Použití knihovny Glut zabere jen pár řádku kódu bez potřeby znalosti specifických rozhraní operačních systémů.

3 Návrh a implementace

3.1 Požadavky

Zadáním byly definovány tyto požadavky:

- přenositelnost
- objektově orientovaný návrh
- jednotné a jednoduché rozhraní
- využitelnost při výuce programování na středních školách

3.2 Návrh knihovny

Návrh knihovny se musí řídit definovanými požadavky. Proto je k jejich splnění důležitá volba programovacího jazyku a programového rozhraní pro práci s grafickým hardwarem. Vhodným jazykem, splňující výše uvedené požadavky, je programovací jazyk C++. Pro práci s grafickým hardwarem byla zvolena knihovna OpenGL a její okenní nadstavba Glut (popsáno v kapitole 2.1 této práce).

Jazyk C++ má mnoho výhod. Jednou z nich je především to, že patří k nejrozšířenějším programovacím jazykům. Tato skutečnost je dána tím, že navazuje na jazyk C a rozšiřuje ho při zachování vysokého výkonu a přenositelnosti. Tento jazyk podporuje procedurální programování, datovou abstrakci, objektově orientované programování a generické programování.

Jednotného a jednoduchého rozhraní může být dosaženo použitím objektově orientovaných vlastností (dědičnost, polymorfizmus) jazyku C++. Dědičnost nám dovoluje znovupoužití již hotových tříd. Nová třída zdědí všechny atributy a funkce své rodičovské třídy a doplní své vlastní, tím vznikne upravená kopie původní třídy. Polymorfizmus představuje další nástroj pro oddělení rozhraní a implementace. Dovoluje zlepšit organizaci kódu a zvýšit jeho čitelnost. S polymorfizmem jsou úzce spjaty virtuální funkce, které tvoří jeho základ. Virtuální funkce umožňují vytvoření dynamické vazby a volání odpovídající funkce instance i při volání přes ukazatel bázové třídy, který na ni ukazuje. Více o virtuálních funkcích a C++ viz. [1].

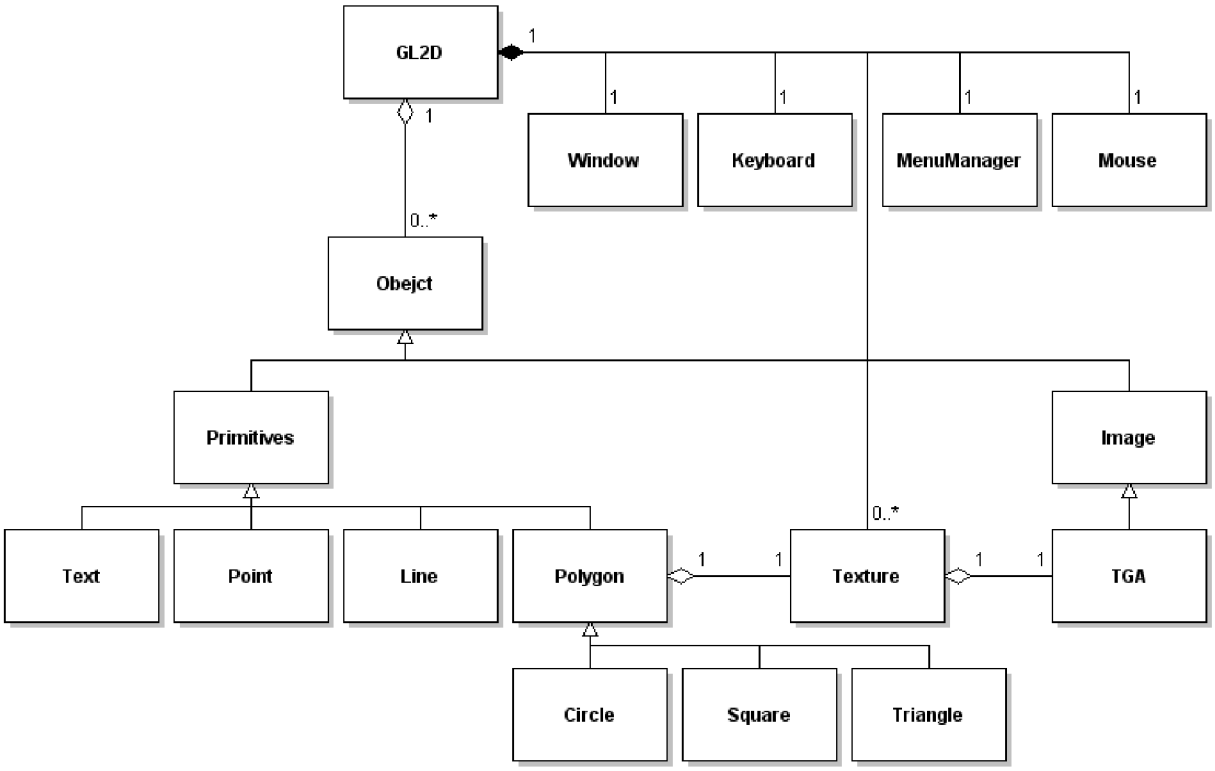
Hlavním požadavkem na tuto knihovnu je využitelnost při výuce programování na středních školách. Knihovna by měla obsahovat funkce pro správu okna, pro uživatelský vstup (myš a klávesnice), případně jednoduché prvky grafického rozhraní. Knihovna by měla být především jednoduchá na pochopení pro studenta, aby se jejím učením neztrácelo mnoho času a aby neodpoutávala studentovu pozornost od probírané látky. Využití grafického výstupu by mělo studentovi pomoci lépe pochopit probíranou látku, např. při výuce algoritmů. Vhodným příkladem je

program pro výpočet a vykreslení nejznámější goniometrické funkce sinus, počítanou pomocí Taylorova polynomu.

Knihovna obsahuje ukázkové programy ke každé své vlastnosti, aby ukázala její použití a tím usnadnila používaní této knihovny.

3.2.1 Objektový model knihovny

Obrázek 2 znázorňuje objektový model knihovny GL2D.



Obrázek 2 – Návrh knihovny

3.3 Vlastnosti grafické knihovny GL2D

Výčet vlastností:

- práce s oknem aplikace
- grafická primitiva – bod, čára
- mnohoúhelníky – trojúhelník, čtverec, kružnice
- barevná výplň mnohoúhelníků
- základní fonty
- textury
- obrázky – TGA

- zobrazení FPS (počet vykreslených snímků za sekundu)

3.4 Třída GL2D

Hlavní třída celé aplikace a zároveň jediná třída, kterou uživatel musí využít pro plnohodnotnou práci s knihovnou. Tato třída má jednoduché rozhraní a spolupracuje s knihovnou Glut. Vytváří callback funkce pro práci s oknem, menu a vstupem z klávesnice a myši.

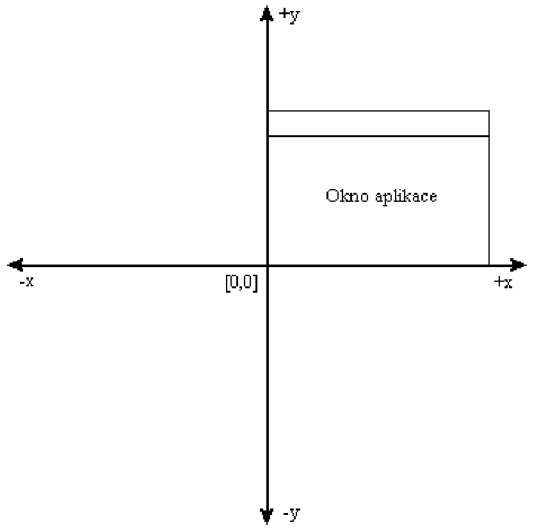
Také obsahuje objekty typu:

- **Window** – třída pro práci s oknem aplikace
- **Keyboard** – třída pro práci s klávesnicí
- **Mouse** – třída pro práci s myší
- **MenuManager** – třída vytvářející jednoduchá vyskakovací menu

3.4.1 Window

Třída *Window* zapouzdřuje okenní funkčnost knihovny Glut. Dovoluje uživateli nastavit základní vlastnosti okna, jako je např. jeho pozice, šířka, výška, barva pozadí a název. Do vytvořeného okna lze pomocí příkazů OpenGL nebo pomocí knihovny GL2D vykreslovat grafická primitiva, obrázky a další objekty. Veškeré změny objektů probíhají v uživatelem definované funkci, kterou si knihovna zavolá pro překreslení obsahu okna.

Souřadný systém okna knihovny začíná bodem [0,0], který je umístěn v levém dolním rohu okna. Pravý horní roh je určen šířkou a výškou okna. Všechny použité rozměry se zadávají v pixelech.



Obrázek 3 – Souřadný systém okna

3.4.2 Keyboard

Pro interaktivitu aplikace s uživatelem je nutné mít možnost získat uživatelův vstup z klávesnice, pomocí jehož se dá program ovládat.

Použití třídy *Keyboard* je velice snadné, uživatel si vystačí s jedinou funkcí. Ke každé klávese může přiřadit svoji funkci, která bude po stisku dané klávesy okamžitě zavolána.

3.4.3 Mouse

Dalším důležitým vstupním zařízením je myš. Knihovna pro zjednodušení práce s touto třídou implementuje mapu klikacích obdélníků. Uživatel může nadefinovat oblast (obdélník) a k němu přiřadit svoji funkci, která bude zavolána kliknutím daného tlačítka myši do této oblasti.

3.4.4 MenuManager

Knihovna také implementuje jednoduchý *MenuManager* pomocí knihovny Glut. Opět bylo dbáno na jednoduchost rozhraní. Uživatel nastaví celé vyskakovací menu s použitím pouze dvou funkcí třídy. Lze vytvořit více vyskakovacích menu a přepínat mezi nimi, ale vždy může být aktivní pouze jedno. Vyskakovací menu je standardně vyvoláno pravým tlačítkem myši, jak je obvyklé z mnoha aplikací, ale toto chování lze změnit.

3.5 Třída Object

Třída *Object* je abstraktní bázovou třídou, od které jsou odvozeny všechny další třídy pro práci s grafikou vytvořené v knihovně GL2D. *Object* je rozhraním pro třídy od něj odvozené. Obsahuje základní funkce pro nastavení pozice objektu v okně a jednu čistou virtuální funkci pro vykreslení objektu.

Třídu *Object* rozšiřuje třída *Primitives* informací o barvě objektu. Od třídy *Primitives* jsou odvozeny základní grafická primitiva:

- *Text* – využívá funkčnosti knihovny Glut pro fonty
- *Point* – třída grafického primitiva bod
- *Line* – třída grafického primitiva úsečka

Více se o rozhraní těchto tříd můžete dozvědět z programové dokumentace umístěné na CD. V následující časti se hlouběji seznámíme se složitějším grafickým objektem třídy Polygon.

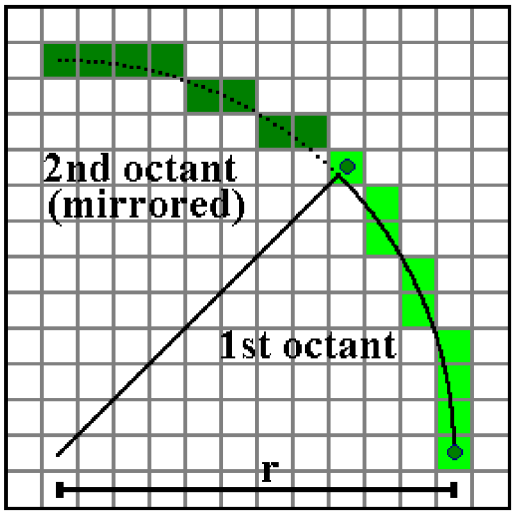
3.6 Polygon

Tato třída dědí od třídy *Primitives* a rozšiřuje rozhraní pro použití 2D objektů (čtverec, trojúhelník a kružnice). Každý 2D objekt má plochu, která může být buď prázdná (vzniká obrys objektu), vyplněná jednou barvou a nebo na ni může být aplikována textura.

3.6.1 Circle

Kružnice je zajímavá hlavně tím, že OpenGL nemá pro její vykreslení žádný příkaz. Proto je potřeba aplikovat nějaký algoritmus pro její vykreslení.

K vykreslení kružnice je použit Midpoint algoritmus, který využívá pouze celočíselnou aritmetiku (sčítání). Hlavní myšlenkou algoritmu je rozhodování, jestli pixel leží na stejné pozici nebo níže podle polohy midpointu vůči kružnici. Využívá osminásobnou symetrii kružnice, takže stačí provést výpočet bodů pro jeden oktant.



Obrázek 4 – Rasterizace kružnice

3.6.1.1 Midpoint algoritmus

Pseudo kód Midpoint algoritmu (více viz [7]):

```
newX = 0
newY = poloměr
tmpX = 3          // pomocné proměnné
tmpY = 2 * poloměr - 2
P = 1 - poloměr   // predikce

Dokud je newX <= newY opakuj:
Vykreslit 8 symetrických bodů s [x,y]
Je-li P > 0 tak
{ P -= tmpY
  tmpY -= 2
  newY -= 1 }
P += tmpX
tmpX += 2
newX +=1
```

3.7 Image

Image je abstraktní třídou poskytující rozhraní pro práci s obrázky. Dovoluje snadno rozšířit knihovnu o další typy grafických formátů.

Každý grafický formát pracuje s obrázkovými daty jinak, proto jsou všechny specifické datové typy uloženy přímo v konkrétní třídě. Také je důležité detekovat a zpracovat případné chyby, které můžou nastat při práci s daty uloženými na pevném disku či jiném médiu.

3.7.1 TGA

Grafický formát TGA (Targa) vytvořila společnost Truevision. Jedná se o formát pro ukládání rastrové grafiky. Data můžou být uložena kromě volné formy i komprimovaně, ale to se v dnešní době skoro nepoužívá.

Typy grafického formátu TGA:

- 1 bit na pixel
- 8 bitů na pixel s barvovou paletou
- 8 bitů na pixel ve stupních šedi
- 16 bitů na pixel (1 bitový alfa kanál)
- 24 bitů na pixel (RGB)
- 32 bitů na pixel (RGB + 8 bitový alfa kanál)

Tento formát je vhodný pro ukládání a načítání textur, protože umožňuje spolu s barevnou informací ukládat i alfa složku. Také je často používán při tvorbě fotorealistické grafiky, většinou se jedná o raytracery (POV-Ray, Vivid) nebo programy pracující s lokálním osvětlovacím modelem (3D Studio). Byl použit v mnoha hrách (Half-life, Sekond Life a dalších). Další informace viz. [4].

Knihovna v současné době podporuje nekomprimované 24bitové a 32bitové formáty TGA.

3.8 Texture

Třída *Texture* zapouzdřuje funkce OpenGL pro texturování. Textury lze vytvářet z knihovnou podporovaných grafických formátů. Rozhraní je opět jednoduché, uživatel jednou definuje jméno k textuře a od té doby pracuje pouze s tímto jménem. Jméno textury může předat objektům, které lze otexturovat a knihovna provede další nezbytné kroky sama. Počet textur v knihovně je omezen (počet textur je také omezen pamětí grafické karty). Více o třídě *Texture* naleznete v programové dokumentaci na CD.

3.8.1 Textury v OpenGL

Mapování textur je metoda přidávání detailu, struktury povrchu nebo barvy počítačem generované grafiky a 3D modelů. OpenGL podporuje pouze rastrové textury, které mohou být jednodimenzionální, dvojdimenzionální nebo třídimenzionální. Textury v OpenGL mají zásadní omezení a to takové, že rozměry textury musejí být mocninou čísla 2 a i jejich maximální velikost je omezená.

Dvojdimenzionální textury jsou běžně používané a podporované na naprosté většině grafických akcelerátorů. Třídimenzionální textury se používají především ve specializovaných aplikacích (medicína, apod.), protože dokáží zobrazit objemová data.

3.8.1.1 Filtrování textur

- GL_NEAREST – nejednoduší a rychlé filtrování, neposkytující dobrou vizuální kvalitu
- GL_LINEAR – používá lineární interpolaci a vytváří realističtější vzhled

3.8.1.2 Mipmapping

Mipmapping je mocná texturovací technika. Nejen, že zvyšuje rychlost vykreslení, ale i vizuální kvalitu scény. Mipmapping vytváří pro jednu texturu sadu dalších textur s různými velikostmi a tím zabrání vytváření nechtěných artefaktů při mapování na malé či velké objekty.

4 Použití grafické knihovny

4.1 Instalace

Požadavky pro použití knihovny GL2D:

- nainstalované knihovny OpenGL a Glut
- knihovny musejí být umístěny tak, aby byly přístupné použitému překladači
- vložení hlavičkového souboru knihovny *gl2d.h* a její zpřístupnění překladači.
- k ukázkovým aplikacím jsou dodány soubory Makefile pro systém Linux

4.2 Náš první program

Po úspěšném nainstalování knihovny GL2D se můžeme pustit do našeho prvního programu. Vzhledem k tomu, že knihovna má být využitelná při výuce programování, tak se v následujícím příkladu budeme zabývat algoritmem funkce sinus a jejího zobrazení. Pro výpočet funkce sinus bude použit Taylorův polynom.

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)!} \qquad (1)$$

Funkci sinus bude možné za běhu aplikace ovládat klávesami a tím měnit její amplitudu, frekvenci, barvu, atd. Nyní Vám to může připadat složité a těžko řešitelné, ale společnými silami, za pomocí knihovny GL2D, bude vytvoření této aplikace hračkou. Při vytváření tohoto programu se seznámíme se základní, ale ne úplnou, funkčností knihovny. Použití dalších konstrukcí je v několika ukázkových programech uložených na přiloženém CD.

4.3 Vytvoření okna aplikace

Při tvorbě každé okenní aplikace je nutné vytvořit její okno. I my tedy tímto začneme programovat naši aplikaci. Seznámíme se třídou *Window* a jejím základním nastavením. Ale již dost teorie, vrhněme se na opravdové programování.

Abychom mohli použít funkce knihovny GL2D musíme do naší aplikace vložit její hlavičkový soubor a vytvořit globální ukazatel na třídu GL2D.

```
#include "gl2d/gl2d.h"

GL2D* g;
```

Dalším důležitým krokem je vytvoření funkce pro vykreslování grafických objektů. Je to jednoduchá funkce bez parametrů a bez návratové hodnoty. Ukazatel na tuto funkci se poté předá knihovně, která se již sama postará o její volání. Prozatím zůstane prázdná, vykreslování objektů si vysvětlíme v následující kapitole.

```
void kresli( void )
{
}
```

Nyní nadefinujeme vstupní funkci do naší aplikace, funkci *main*. Zde přiřadíme námi deklarovanému ukazateli na třídu GL2D její konkrétní instanci a předáme ukazatel na funkci *kresli*. Nyní již můžeme měnit nastavení okna pomocí členských funkcí třídy *Window*. Případně, pro zlepšení čitelnosti kódu, tyto příkazy můžeme přesunout do funkce a odtud ji zavolat.

Nakonec zbývá spustit knihovnu příkazem *Run()* a smazat dynamicky alokované proměnné před ukončením programu. I když pravděpodobně každý moderní operační systém tyto data odstraní z paměti sám, je to programátorskou slušností a dobrým návykem.

```
int main( void )
{
      g = new GL2D(kresli);
      g->window->setClearColor(BLUE);
      g->window->setSize(400,100);
      g->window->setName("Moje 1. okno");
      g->Run();
      delete g;
      return 0;
}
```

Přeložením tohoto programu vznikne naše první okenní aplikace. Ke všem parametrům a funkcím přistupujeme přes námi definovaný ukazatel *g*. Všechny parametry okna jsou uloženy ve třídě *Window* a jen pomocí její funkcí je můžeme měnit. Např. funkce *setSize(int width, int height)* nám umožňuje změnit rozměry okna. Další funkce lze zjistit z programové dokumentace nacházející se na přiloženém CD. Celý ukázkový příklad se nachází v příloze.

Obrázek 5 – Výsledek prvního příkladu, jednoduché okno

4.4 Vykreslení grafických objektů

Nyní již máme okno inicializováno a připraveno pro vykreslení nějakých objektů. Tato část bude věnována vykreslení grafu funkce sinus pomocí grafického objektu *Point.*

Graf funkce sinus se skládá z mnoha bodů, proto se musíme rozhodnout, jak je uložíme v naší aplikaci. Jednoduchým řešením je použití standardních kontejnerů jazyka C++ (viz. [2]). Vytvoříme kontejner typu *vector,* obsahující ukazatele na objekty *Point*, reprezentující body funkce sinus.

```
#include <vector>

std::vector<Point*> body;
```

Po definici vektoru bodů přistoupíme k jejich výpočtu. Výpočet budeme provádět funkcí *double sinus(double cislo, int presnost).* Funkce vypočte pro zadané číslo hodnotu funkce sinus. Parametr *presnost* určuje počet iterací při výpočtu. U této funkce je výhodné převést číslo zadané jako parametr do první periody funkce sinus.

Musíme definovat několik globálních proměnných, které ovlivní zobrazení funkce sinus.

```
float frekvence = 1.0;
float perioda = 1.0;
float amplituda 1.0;
int velikost = 30;
int x = 50;
int y = 50;
Color barva = WHITE;
```

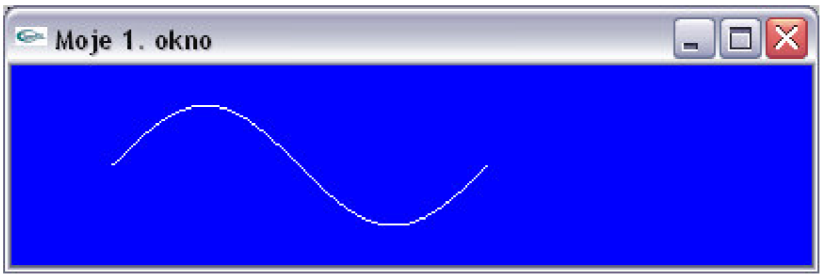
Pro výpočet bodů křivky musíme definovat ještě jednu funkci. Tato funkce slouží k přepočítání bodů pří každé změně parametrů. Důležitým faktorem je krok výpočtu bodů. Krok je počítán automaticky tak, aby dle zvolených parametrů vykreslil souvislou křivku.

```
void vypocetSinu()
{
      if(velikost <= 1) velikost = 1;
      if(perioda < 0) perioda = 0;
      if(frekvence < 0) frekvence = 0;
      if(amplitude < 0 ) amplitude = 0;
      int dx = x; int dy = y;
      float krok = 1.0 / frekvence / velikost / amplitude;
      for(int i=0; i<body.size(); i++)
            delete body[i];
      body.clear();
      for(float x = 0; x <= 2*PI*perioda; x+=krok)
      {
            Point* p = new Point(int(x*velikost + dx),
                  int(amplitude*sinus(x*frekvence,10)*
                        velikost + dy));
            p->setColor(barva);
            body.push_back(p); }}
```

Poslední úpravou v této kapitole je doplnění funkce *kresli()* o kód potřebný k vykreslení bodů křivky.

```
for(int i=0; i<body.size(); i++)
      body[i]->Draw();
```

Přeložením vznikne naše druhá aplikace, která již dokáže něco vykreslit, v tomto případě graf funkce sinus.



Obrázek 6 – Výstup aplikace Sinus

4.5 Použití myši a klávesnice

V tomto okamžiku je náš program připraven na rozšíření o nějakou tu interaktivitu. Použijeme několik kláves ke změnám parametrů vykreslené křivky. Tyto změny jsou: zvětšení či zmenšení frekvence, amplitudy, velikosti a počtu period.

Použití klávesnice je velmi jednoduché, což demonstrují následující řádky. Pomocí funkce *setKeyFunc()* můžeme přiřadit konkrétní klávese funkci, která bude při stisku té klávesy provedena. Pro výčet použitelných kláves nahlédněte do programové dokumentace.

```
void initKeyboard()
{
      g->keyboard->setKeyFunc('+', zvetseni);
      g->keyboard->setKeyFunc(LEFT, frekvencePlus);
      // atd.
}

void frekvencePlus()
{
      frekvence += 0.1;
      vypocetSinu();
}

//...
```

Použití funkce myši je také velmi jednoduché, i když trochu omezené, pro zachování jednoduchosti může uživatel definovat obdélníkovou oblast okna (či více oblastí), které reagují na stisk tlačítka myši a vyvolají nastavenou funkci.

```
void initMouse()
{ g->mouse->addClick(10,5,100,20, prepniNapovedu); }
```

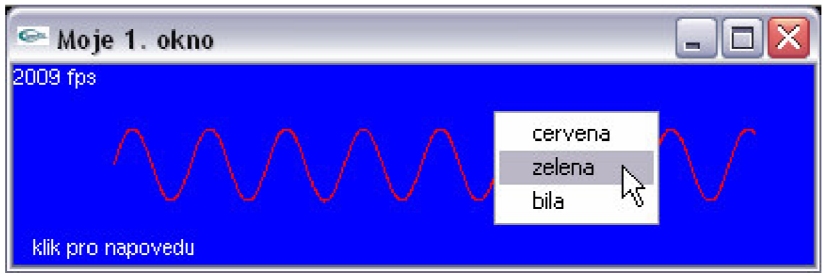
4.6 Vyskakovací menu

Vyskakovací menu, reagující na stisk tlačítka myši, ještě dále zlepšují ovladatelnost programu. V naší aplikaci si ukážeme použití na jednoduchém menu obsahující tří položky. Každá položka reprezentuje barvu pro vykreslení křivky.

Pomocí příkazu *createMenu()* vytvoříme menu se zadaným názvem, který nám bude sloužit k aktivaci menu, v případě, že použijeme více než jedno. Nové menu je automaticky nastaveno jako aktivní. Poté příkazem *addMenuEntry()* přidáme jednotlivé položky do aktivního menu. Položka se skládá z názvu a funkce, která bude vykonána při výběru této položky.

```
void initMenu()
{
      g->menu->createMenu("barvy");
      g->menu->addMenuEntry("cervena", cervenaBarva);
      g->menu->addMenuEntry("zelena", zelenaBarva);
      g->menu->addMenuEntry("bila", bilaBarva);
}

void cervenaBarva() { barva = RED; vypocetSinu();}
```



Obrázek 7 – Výstup hotové aplikace Sinus

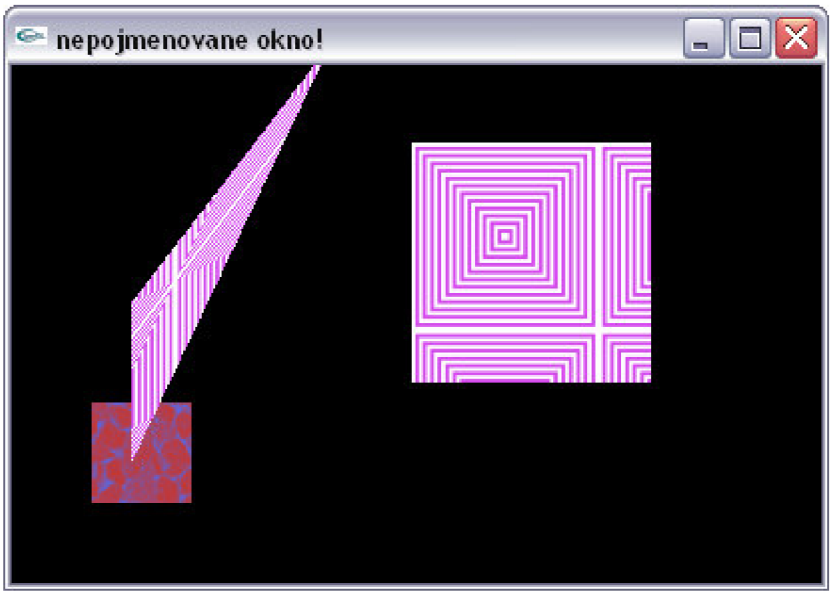
4.7 Použití textur

Zde již nebudeme rozšiřovat program Sinus, ale podíváme se na velmi zajímavou funkci knihovny. Ke každému dvojrozměrnému objektu podporujícího aplikaci textury, lze snad přiřadit texturu pomocí funkce *setTexture()*. Knihovnou omezený počet textur je 32, což je pro účel jejího použití množství dostatečné.

K vytvoření textury musíme použít funkci *createTexture()*, jak je patrno z následující ukázky.

```
Square s1(40,40,50); //ctverec, na který mapujeme texturu
g->texture->createTexture("pokus.tga", "prvni");
s1.setTexture("prvni"); //přiřazení textury
```

Celý zdrojový soubor je umístěn na přiloženém CD.



Obrázek 8 – Ukázka texturování

5 Multimediální knihovny

Multimediální knihovna je kolekcí specializovaných knihoven poskytujících určité služby. Tato knihovna by měla obsahovat knihovny:

- pro práci s grafikou
- pro práci se zvukem
- pro práci s okny
- pro práci se sítí a další…

Není mnoho volně dostupných multimediálních knihoven, které by splňovaly tyto požadavky. A z těchto se zaměříme na knihovny DirectX, SFML a SDL, více či méně splňujících tyto požadavky.

5.1 Vybrané multimediální knihovny

5.1.1 DirectX

V roce 1995, krátce před vydáním systému Windows 95 firmou Microsoft, bylo potřeba rozhraní k přístupu ke grafickému hardwaru, zvukovému hardwaru a dalším periferiím pro programátory. Microsoft tedy přišel s aplikačním programovým rozhraním DirectX.

Hlavním cílem DirectX je podpora vytváření multimediálních aplikací a programování her na platformách firmy Microsoft (systémy Windows a konzole Xbox). DirectX zahrnuje několik komponent.

5.1.1.1 DirectDraw

Komponenta pro kreslení 2D grafiky, v dnešní době je již zavržená. Přesto ji mnoho her stále používá.

5.1.1.2 Direct3D

Komponenta pro práci s 3D grafikou. Poskytuje nízko-úrovňové rozhraní pro práci s funkcemi grafických karet (transformace, osvětlení, textury a další…)

5.1.1.3 DirectInput

Komponenta pro práci se vstupem (myš, klávesnice, joystick a další herní ovladače). Po verzi 8 je zavržena.

5.1.1.4 DirectPlay

Komponenta pro komunikaci na LAN a WAN sítích. Po verzi 8 je také zavržena.

5.1.1.5 DirectSound

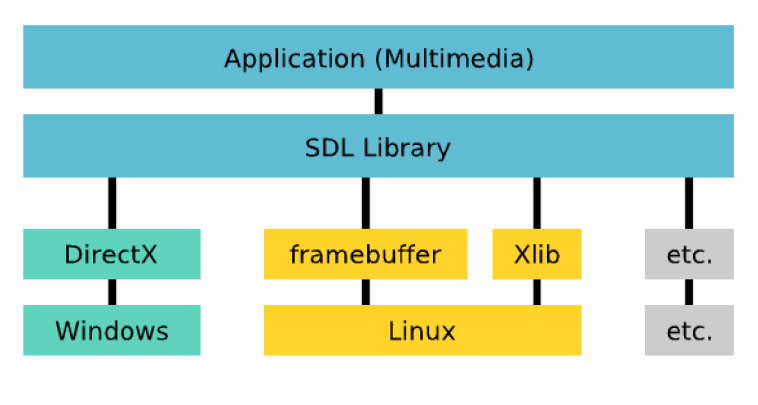
Komponenta pro přehrávání zvuků. Dále existuje DirectSound3D pro přehrávání 3D zvuků pro zvýšení realičnosti.

5.1.1.6 DirectShow

Komponenta pro přehrávání multimediálních formátů.

5.1.2 SDL

SDL (Simple DirectMedia Layer) je přenositelná multimediální knihovna napsaná v jazyku C. Vytváří abstrakci nad různými grafickými platformami, zvukem a vstupem. Dovoluje vývojářovi napsat počítačovou hru nebo jinou multimediální aplikaci jednou a spouštět na různých systémech bez dalších úprav. Knihovna je rozčleněna do podsystémů jako např.: Video, Audio, CD-ROM, Joystick a Timer.



Obrázek 9 – Abstraktní vrstvy několika SDL platforem

5.1.3 SFML

SFML (Simple and Fast Multimedia Library) je přenositelné multimediální aplikační programové rozhraní napsané v C++. Na tuto knihovnu můžeme nahlížet jako na moderní, objektově orientovanou alternativu k SDL. SFML se skládá z několika knihoven. Je možné ji použít jen jako okenní nadstavbu pro OpenGL nebo jako plnohodnotnou multimediální knihovnu pro tvorbu her a interaktivních programů.

5.1.3.1 Window

Dokáže vytvořit několik renderovacích oken. Umí pracovat s myší (až 5 tlačítek), podporuje dva joysticky až se 7 osami a 16 tlačítky. Svými funkcemi může nahradit Glut.

5.1.3.2 Graphics

Komponenta Graphics podporuje moderní efekty a hardwarovou akceleraci. Efektivně spravuje paměť, takže není zapotřebí se starat o uložení a délku života zdrojů. Dokáže načítat i ukládat obrázky typu bmp, jpg, png, tga, dds a psd. Podporuje jednoduchou manipulaci s grafickým textem.

5.1.3.3 Audio

Hardwarová akcelerace je použita kdykoliv je to možné. Pracuje s formáty typu: ogg, wav, aiff, au, raw, atd. Také se specializuje na 3D zvukové efekty a podporuje streamování velkých souborů.

5.1.3.4 Network

Implementuje přenositelnou vrstvu nad TCP a UDP sokety. Jednoduché přenosy dat pomocí proudově založených rozšiřitelných paketů.

5.2 Porovnání

Všechny tyto multimediální knihovny mají více či méně shodné vlastnosti. Jsou to rozsáhlá aplikační programová rozhraní využitelná v mnoha multimediálních aplikacích, využívají výhody akcelerovaného hardwaru, pokud je dostupný. Lze s nimi pracovat na většině dnes používaných platformách, mimo knihovny DirectX, která je pouze pro systémy Windows.

Tyto knihovny poskytují komplexní funkčnost a právě to je činí těžko využitelné při výuce programování na středních školách. K pochopení a použití konstrukcí těchto knihoven by bylo zapotřebí mnoho hodin studia a alespoň základní znalost teorie a práce s grafickým výstupem, což se u začínajících programátorů nedá očekávat.

5.3 Výhody knihovny GL2D

Výše zmíněné nevýhody multimediálních knihoven vedly k vytvoření takové knihovny, která by nezatěžovala studenty programování dalšími nároky zvyšující obtížnost studia.

Sice není multimediální a její funkčnost se nedá srovnávat s těmito multimediálními knihovnami. Obsahuje jen základní grafické objekty a operace. A největší výhodou je jednoduchost jejího použití. A proto si troufám tvrdit, že každý student, po seznámení se s mou knihovnou v kapitole 4, je schopen ji začlenit do svých školních projektů.

6 Závěr

Zadáním byl dán požadavek na vytvoření grafické knihovny se zaměřením na výuku programování na středních školách, která je založena na objektově orientovaném návrhu a je přenositelná. Jedním z hlavních požadavků na knihovnu je jednotnost a jednoduchost jejího rozhraní.

Dle mého názoru je jednoduchost knihovny daná především tím, že zapouzdřuje komplexnost práce s grafickým výstupem a nabízí jednoduché rozhraní pro snadné použití. Obsahuje sadu základních objektů, které lze využít pro vizualizaci řešené tématiky při výuce. Je vhodná pro začínající programátory bez znalosti práce s grafickým výstupem. Díky této knihovně se i zlehka ponoří do problematiky programování grafiky či interaktivních aplikací.

Díky těmto vlastnostem bylo dosaženo uspokojivých výsledků, které byly vytyčeny zadáním. Přesto si myslím, že tato knihovna má určité rezervy, na nichž by se dalo v budoucnu zapracovat. Např. závislost na knihovně Glut, která je v mnoha směrech velmi omezující a nevhodná pro větší projekty.

Možnosti dalšího vývoje knihovny jsou široké. Vývoj by se mohl ubírat několika odlišnými směry. Jedním z nich může být rozšíření funkčnosti, tím pádem konkurenceschopnosti multimediálním knihovnám. Další cestou může být přechod z 2D do 3D a např. implementace funkčnosti pro vytváření jednoduchých herních aplikací.

Literatura

[1] Eckel B. *Myslíme v jazyku C++*. Praha, Grada Publishing, spol. s r. o., 2000. ISBN 80-247-9009-2

[2] Eckel B., Allison Ch. *Myslíme v jazyku C++, 2. díl*. Grada Publishing, a.s., 2006 ISBN 80-247-1015-3

[3] Wright S. R., Jr., Lipchak B., Haemel N. *OpenGL SuperBible*. Boston, Addison-Wesley. 2007 ISBN 0-321-49882-8

[4] Tišnovský P. *Grafický formát TGA* [online] 2006 [cit. 5.5.2008] Dostupné z WWW: <http://www.root.cz/clanky/graficky-format-tga-jednoduchy-oblibeny-pouzivany>

[5] Wikipedia, the free encyclopedia: *Mark Kilgard* [online] [cit. 5.5.2008] Dostupné z WWW: <http://en.wikipedia.org/wiki/Mark_Kilgard>

[6] Wikipedia, the free encyclopedia: *OpenGL* [online] [cit. 5.5.2008] Dostupné z WWW: <http://en.wikipedia.org/wiki/OpenGL>

[7] Kršek P., Španěl M. *Základy počítačové grafiky: Rasterizace objektů ve 2D.* UPGM, FIT, VUT, Brno [online] [cit. 5.5.2008] Dostupné z WWW: <https://www.fit.vutbr.cz/study/courses/IZG/private/lecture/izg_slide_rasterizace_print.pdf>

Seznam příloh

Příloha 1. Obsah přiloženého CD

Příloha 2. CD

Příloha 1

CD obsahuje zdrojové soubory knihovny, ukázkové aplikace, programovou dokumentaci a tuto práci.