Promítací algoritmy v počítačové grafice

Projection algorithms in computer graphics

Boris Barák

Bakalářská práce
2007

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2006/2007

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Boris BARÁK**
Studijní program: **B 3902 Inženýrská informatika**
Studijní obor: **Informační technologie**

Téma práce: **Promítací algoritmy v počítačové grafice**

Zásady pro vypracování:

1. Vytvořte literární rešerši na zadané téma.
2. Seznamte se s dostupnými druhy promítání, které se v současné době v oblasti 3D grafiky používají. Tato promítání v práci podrobněji charakterizujte.
3. Proveďte návrh 3D grafické aplikace založené na OpenGL, která bude implementovat algoritmy promítání uvedené v teoretické části práce.
4. Aplikaci naprogramujte a vytvořte vhodné uživatelské rozhraní pro základní práce s tímto programem (práci se soubory, volba typu promítání, změna parametrů apod.).
5. Na základě tohoto tématu vytvořte prezentaci v Powerpointu, která by se dala použít při výuce Počítačové grafiky.

Rozsah práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1.Martišek, D.: Matematické principy grafických systémů.Littera, Brno 2002.
2.Žára, J., a kol.: Moderní počítačová grafika. Computer Press, 2005.
3.Shreiner, D. a kol.: OpenGL – průvodce programátora. Computer Press, 2006.
4.Liberty, J. Naučte se C++ za 21 dní, Computer Press, Praha 2002.

Vedoucí bakalářské práce: **Ing. Pavel Pokorný, Ph.D.**
Ústav aplikované informatiky

Datum zadání bakalářské práce: **13. února 2007**

Termín odevzdání bakalářské práce: **24. května 2007**

Ve Zlíně dne 13. února 2007

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky

prof. Ing. Vladimír Vašek, CSc.
děkan

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Cílem této bakalářské práce je podrobně charakterizovat algoritmy, které se v současné době používají v prostorové počítačové grafice a rovněž vytvořit program společně s prezentací těchto algoritmů pro potřeby výuky počítačové grafiky. Program je navržen v jazyce C++ tak, aby umožnil výběr modelu načítaného ze souboru a jeho zobrazování pomocí OpenGL v druzích promítání probraných v teoretické části. Aplikace má jednoduché uživatelské rozhraní pro výběr souboru, nastavení polohy modelu, změnu typu promítání včetně jejich parametrů, které mohou být taktéž zobrazovány.

Klíčová slova: promítání, 3D grafika, OpenGL, SDL, C++

ABSTRACT

It is an objective of the bachelor thesis to characterize algorithms which are used in the three-dimensional graphics nowadays in detail, and also to create a program, along with the presentation of these algorithms, for needs of teaching the computer graphics. The program is designed in the C++ language in order to enable selection of a model of a loaded file and its visual display through the use of OpenGL in the types of projection taught in the theoretical part. The application has simple interface for selecting a file, setting the model position, changing the type of projection including its parameters, which may also be displayed.

Keywords: projection, 3D graphics, OpenGL, SDL, C++

Poděkování:

Mé díky patří Ing. Pavlu Pokornému, Ph.D. za velmi cenné rady, které mi byly vydatnou pomocí při zpracovávání této práce a také za čas věnovaný mé prácí.

Motto:

„Je vždy lehčí přizpůsobit své požadavky programu než naopak.“

Murphyho počítačové zákony - Osmý zákon programování

Prohlašuji, že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků, je-li to uvolněno na základě licenční smlouvy, budu uveden jako spoluautor.

Ve Zlíně …….

Podpis diplomanta

OBSAH

ÚVOD

V současné době je velkým trendem používat zobrazování objektů ve trojrozměrném prostoru v počítačové grafice. Při zobrazování těchto objektů, které jsou většinou vytvořeny pouze jako posloupnost souřadnic v prostoru, je nutné zvolit nějakou metodu, která nám zajistí co nejoptimálnější vidění objektu na ploše (např. monitoru nebo plátně) ať už z hlediska měřitelnosti a porovnávání rozměrů na této ploše nebo z hlediska realistického vnímání objektu. Aplikování této metody se nazývá promítání.

Různé druhy promítání jsou tedy vhodné k jinému účelu a tak je například vhodné použít jiné promítání pro zobrazování strojních součástí než pro počítačové hry. Po promítání zpravidla následují další metody jako nanesení textury, určení viditelnosti objektu a další. Práce se však zabývá především promítáním.

Tato práce vznikla proto, aby pomohla studentům s touto na prostorovou představivost náročnější částí principů počítačové grafiky. Práce je rozčleněna na teoretickou a praktickou část.

V teoretické části je cílem rozebrat všechny dnes používané druhy rovnoběžného a středového promítání. Ke každému jednotlivému druhu je podán co nejvýstižnější popis, případně i konkrétní odvození matematických vztahů charakterizujících dané promítání.

V praktické části jsem navrhl a naprogramoval v jazyce C++ s využitím knihovny OpenGL výukovou aplikaci pro zobrazování různých druhů promítání probraných v teorii s možností použití externích modelů vyexportovaných z modelovacích programů a vlastních textur. Program je také řádně popsán a zdokumentován v praktické části textu. Součástí práce je ještě prezentace vytvořená programem Microsoft PowerPoint.

I. TEORETICKÁ ČÁST

1 DEFINICE PROMÍTÁNÍ

1.1 Promítání obecně

Prostorové objekty bývají zobrazovány na dvourozměrných zobrazovacích zařízeních (monitor, televizor, plátno apod.). Převedení trojrozměrných objektů do podoby dvourozměrné je realizováno promítáním.

Studiem promítacích metod se zabývá vědní obor *deskriptivní geometrie*. Ten rozlišuje celou řadu postupů určených k tomu, abychom i z dvourozměrného obrazu získaného promítáním mohli zpětně odvodit prostorové vztahy, např. vzdálenosti a úhly. V počítačové grafice však vystačíme s omezenou množinou promítacích metod – rovnoběžným a středovým promítáním na jednu průmětnu.

U promítání budeme používat následující pojmy:

- *Promítací paprsek* je přímka vedoucí promítaným bodem, jejíž směr závisí na promítací metodě,
- *Průmětna* je plocha v prostoru, na kterou dopadají promítací paprsky. V místě dopadu vytvářejí průmět, což je výsledný obraz na této ploše.

Pokud je průmětna rovinná, prostorové úsečky se promítají do úseček v rovině, takže je transformace nezakřivuje. Nemusíme tedy promítat všechny body prostorových objektů. V případě úseček, tvořících často hrany těles, stačí podrobit pouze koncové body a ty spojit v průmětně. Úsečka v průmětně bývá zpravidla pasterizovaná.

Při zobrazování prostorových objektů následuje po promítání další zpracování dat: nalezení zakrytých a viditelných částí objektů, vyhodnocení jejich barvy, nanesení textur. Z důvodu snazšího zpracování těchto kroků se stalo zvykem používat jako průmětnu rovinu *xy*, resp. jakoukoliv rovinu kolmou na osu z. Budeme tedy metody promítání uvádět s ohledem na průmětnu rovnoběžnou s rovinou *xy*. Promítání do jiné průmětny můžeme vždy převést pomocí transformací (otočením, posunutím apod.) prostorových objektů na uvedený případ. [1]

1.2 Matematická definice promítání

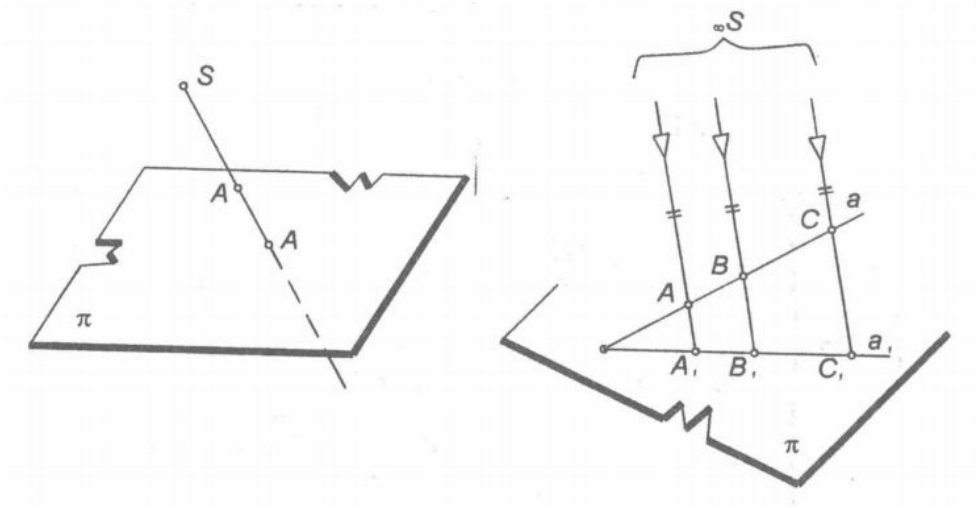
Matematická definice promítání zní: Nechť je dán pevný bod S a pevná rovina π, které nejsou incidentní, tj. $S \notin \pi$. Zobrazení, ve kterém je obrazem bodu $A \neq S$ v prostoru průsečík A′ přímky |SA| s rovinou π, se nazývá promítání (projekce) z bodu S do roviny π.

Bod S se nazývá střed promítání, rovina π průmětna, spojnice |SA| promítací přímka a bod A′ průmět bodu A. Promítání je určeno středem S a průmětnou π.

- Průmětem bodu, který je různý od středu promítání, je bod.
- Průmětem přímky, která neprochází středem promítání, je přímka. Průmětem promítací přímky je bod.
- Průmět bodu je bod ležící na průmětu přímky.
- Průmětem roviny, která neprochází středem promítání, je průmětna. Průmětem promítací roviny (tj. roviny, procházející středem S) je přímka.
- Promítáním se zachovává incidence.

Jestliže střed promítání je vlastní bod, promítání se nazývá *středová* neboli *centrální projekce*. Jestliže střed promítání je nevlastní bod, promítání se nazývá *rovnoběžné* neboli *paralelní projekce*. [3]

Uvedená tvrzení jsou patrná z následujícího obrázku.



Obrázek 1 Definici středové (centrální) projekce a rovnoběžné (paralelní) projekce

1.3 Transformační matice

Maticí se obecně nazývá speciálně zvolená tabulka reálných nebo komplexních čísel.

$$\begin{pmatrix} a(1,1) & a(1,2) & \dots & a(1,n) \\ a(2,1) & a(2,2) & \dots & a(2,n) \\ \dots & \dots & \dots & \dots \\ a(m,1) & a(m,2) & \dots & a(m,n) \end{pmatrix}. \tag{1}$$

Číslo a(i,j) je prvek matice, která obsahuje n sloupců a m řádků, pro které platí, že $1 \leq i \leq m$ a $1 \leq j \leq n$. Rozměrem matice je m x n, což se zapisuje jako $A(m \text{ x } n)$, kde index i ukazuje, na kterém řádku se prvek nachází a index j ukazuje na sloupec matice. [2]

V prostorové počítačové grafice se používají transformační matice o rozměrech 4 x 4. Jako vzor si uvedeme matici

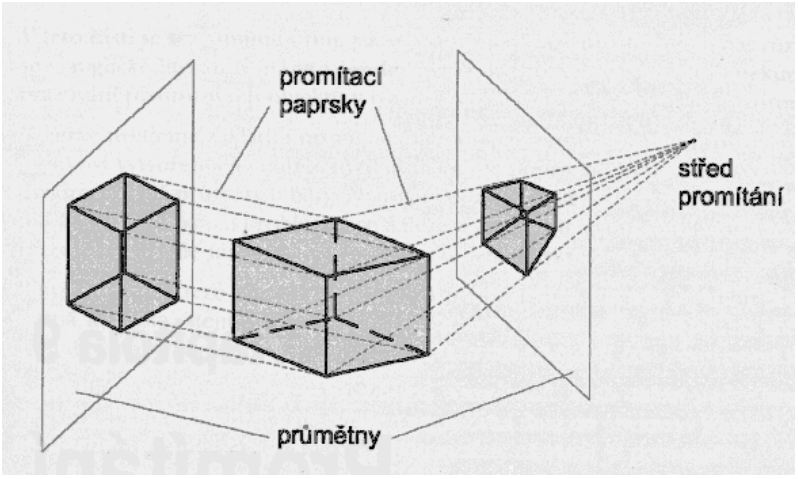
$$\begin{pmatrix} RotXx & RotYx & RotYZx & 0 \\ RotXy & RotYy & RotYZy & 0 \\ RotXz & RotYz & RotYZz & 0 \\ MoveX & MoveY & MoveZ & 1 \end{pmatrix}. \tag{2}$$

Jak je patrno z definice, poslední řádek (mimo poslední sloupec, tj. prvky s názvem obsahujícím *Move*) slouží k určení hodnot posunutí v jednotlivých osách. Složitější je to však s prvky pro rotaci. Prvky v této submatici 3 x 3 (v definici označené částí názvu *Rot*) definují rotací původní osy ve směru osy jiné (např. v prvku označeném *RotXy* určíme hodnotu pootočení osy x na osu y). Poslední sloupec matice je zde nejen proto, aby se daly jednotlivé transformační matice mezi sebou násobit, resp. aby se bylo možné na sebe skládat jednotlivé transformace, ale také pomocí něm můžeme nastavit perspektivu. [8]

Díky transformačním maticím můžeme jednoduchým způsobem definovat a používat nejrůznější geometrické transformace, jako jsou např. posunutí, otočení, změna měřítka, souměrnost a zkosení a to ve všech osách prostoru. Tyto matice stačí v patřičném pořadí vynásobit se stávající projekční maticí a získáme výsledné promítání.

2 DRUHY PROMÍTÁNÍ

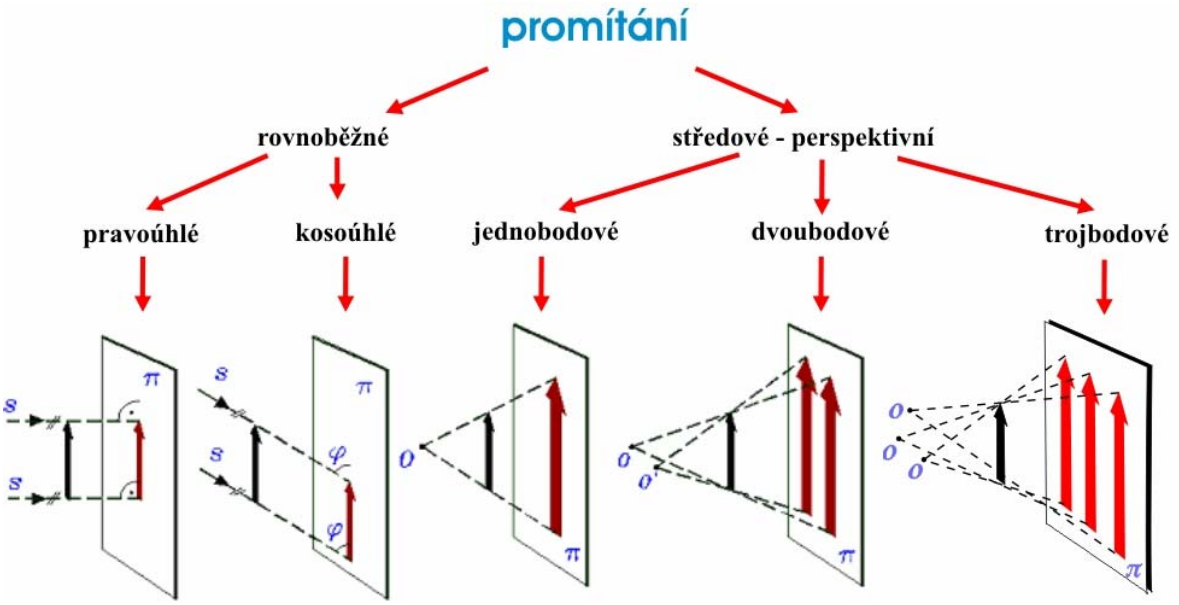
Promítání dělíme, jak již bylo uvedeno, na *rovnoběžné* a *středové* podle rovnoběžnosti, resp. různoběžnosti promítacích paprsků. Na následujícím obrázku jsou vidět názorné ukázky těchto promítání. Z objektu, který máme zobrazit jsou směrem doleva vedeny rovnoběžné promítací paprsky. Jejich průsečíky s průmětnou jsou body výsledného obrazu. K průmětně vpravo vedou promítací paprsky směřující ke středu promítání. V tomto bodě se všechny tyto paprsky stýkají. Na průmětně se opět zobrazí objekt tvořený body, které vznikly protnutím promítacích paprsků s průmětnou. Při tomto druhu promítání tedy závisí tvar a velikost zobrazeného objektu na umístění vzhledem k průmětně.



Obrázek 2 Ukázka rovnoběžného a středového promítání

Pokud je průmětna rovinná, prostorové promítací paprsky se promítají do úseček v rovině (nejsou zakřiveny). U prostorových objektů se nemusí provádět promítání všemi body. V případě úseček (tvořící hrany těles) stačí podrobit promítání pouze koncové body a ty spojit rasterizací v průmětně.

Po promítání následuje další zpracování dat: nalezení viditelných částí objektů, vyhodnocení jejich barvy a nanesení textur. Pro zjednodušení se často jako průmětna používá rovina *xy* (kolmá na osu *z*). [2]



Obrázek 3 Klasifikace základních promítacích metod

2.1 Rovnoběžné promítání

Při tomto způsobu promítání jsou všechny promítací paprsky rovnoběžné. Podle toho, jaký svírají úhel s průmětnou, dělíme rovnoběžné promítání na *pravoúhlé* pro úhel 90° a *kosoúhlé* pro jiné úhly, než je 90°.

Rovnoběžné promítání obecně je typické pro technické aplikace, neboť zachovává rovnoběžnost. Vzdálenost průmětny od promítacích objektů neovlivňuje velikost průmětů. [1]

2.1.1 Pravoúhlé promítání

V počítačové grafice se téměř výhradně používá pravoúhlé promítání. Rovnoběžné promítání do roviny *xy* kolmými paprsky popsanými vektorem (0, 0, -1) představuje jednoduše zanedbání souřadnic bodů. Takovou transformaci popíšeme maticí

$$P_{xy} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \qquad (3)$$

Obdobně by tomu bylo, kdybychom promítali do roviny *xz* nebo *yz*. Transformační matice by byly

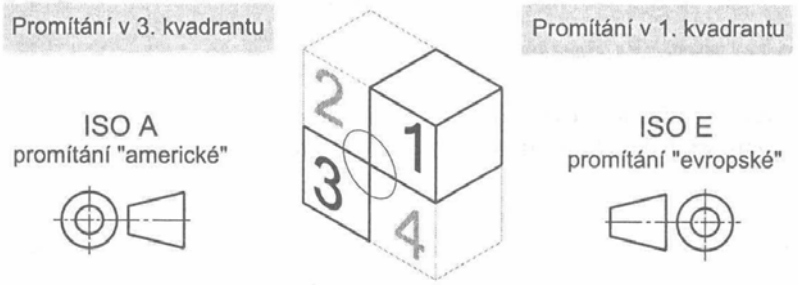
$$P_{xz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4}$$

pro promítání do roviny *xz* (nárys) nebo pro promítání do roviny *yz*

$$P_{yz} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5}$$

Průmět získaný z promítání do roviny *xy* představuje půdorys. Pro získání pohledů z jiných směrů nejprve nalezneme transformaci, která objekt posune a otočí do vhodné promítací polohy nad průmětnou *xy*. Odpovídající transformační matici složíme s maticí rovnoběžného promítání P_{xy} a výslednou transformaci použijeme na všechny prostorové objekty.[1]

Pravoúhlé promítání se v technické praxi dělí podle mezinárodních norem ISO na dvě metody, které se liší umístěním objektu vůči pozorovateli a průmětnám. Jejich název je odvozen podle umístění v soustavě navzájem kolmých rovin. Soustava rovin je rozdělena na čtyři kvadranty. Pro promítání se využívá pravidlo prvního a třetího kvadrantu (viz. obrázek).

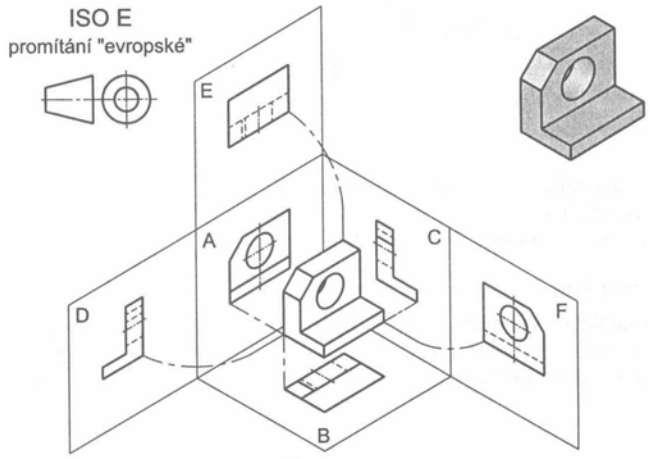


Obrázek 4 Orientace promítání v prvním a třetím kvadrantu.

Každý kvadrant umožňuje promítání celkem na šest rovin. Promítání v prvním kvadrantu se používá v evropských zemích a proto se nazývá *evropské*. Promítání ve třetím kvadrantu se nazývá *americké*. [6]

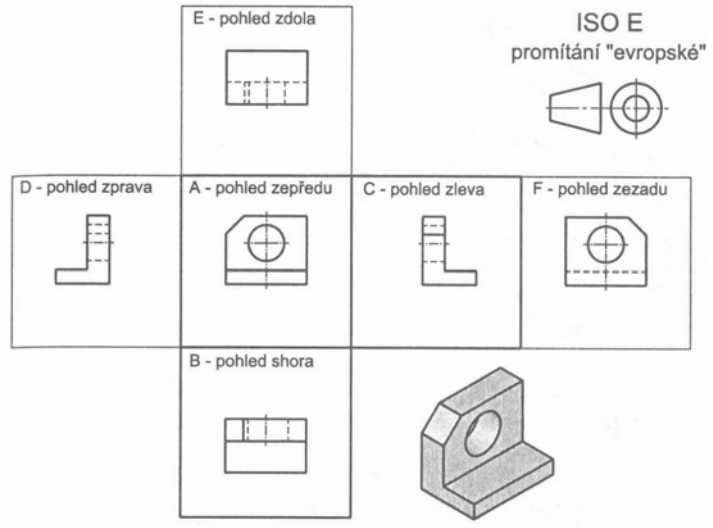
2.1.1.1 Evropské promítání

Promítání v prvním kvadrantu je typem promítání, které se běžně používá v pravidlech technického zobrazování v evropských zemích. Jedná se o způsob pravoúhlého promítání, při němž leží objekt mezi pozorovatelem a průmětnou. Tento typ promítání se značí *ISO E*.



Obrázek 5 Obrázek objektu zobrazovaného v prvním kvadrantu s průmětnami

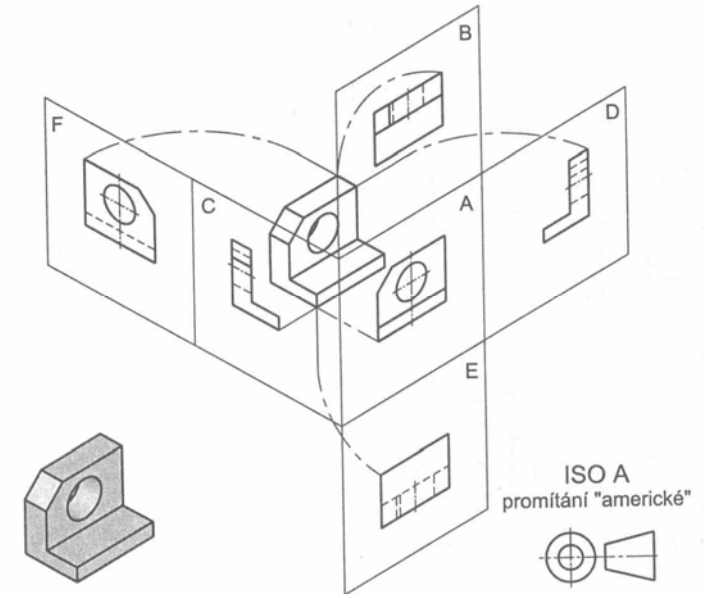
Běžné rozložení sdružených obrazů pro promítání v prvním kvadrantu je zobrazeno na následujícím obrázku. Velmi často používanou kombinací sdružených obrazů ve výkresové dokumentaci je kombinace pohledu zepředu, pohledu shora a pohledu zleva. Pohled zepředu také bývá nazýván nárys, pohled shora půdorys a pohled zleva, resp. zprava se nazývá *levý bokorys*, resp. *pravý bokorys*. [6]



Obrázek 6 Obrázek objektu zobrazovaného v prvním kvadrantu s průmětnami

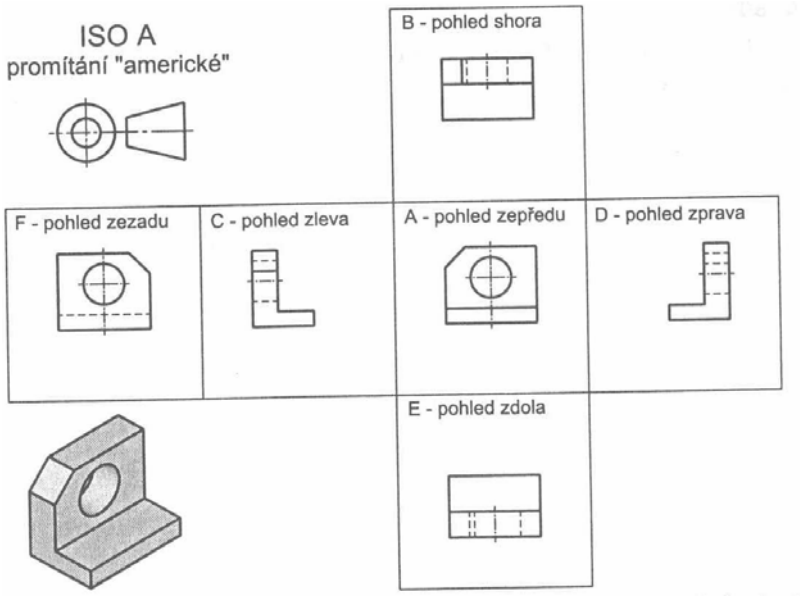
2.1.1.2 Americké promítání

Promítání ve třetím kvadrantu je způsobem pravoúhlého promítání, při němž leží objekt pro pozorovatele za průmětnami. Tento typ promítání se označuje *ISO A*. [6]



Obrázek objektu zobrazovaného v třetím kvadrantu s průmětnami

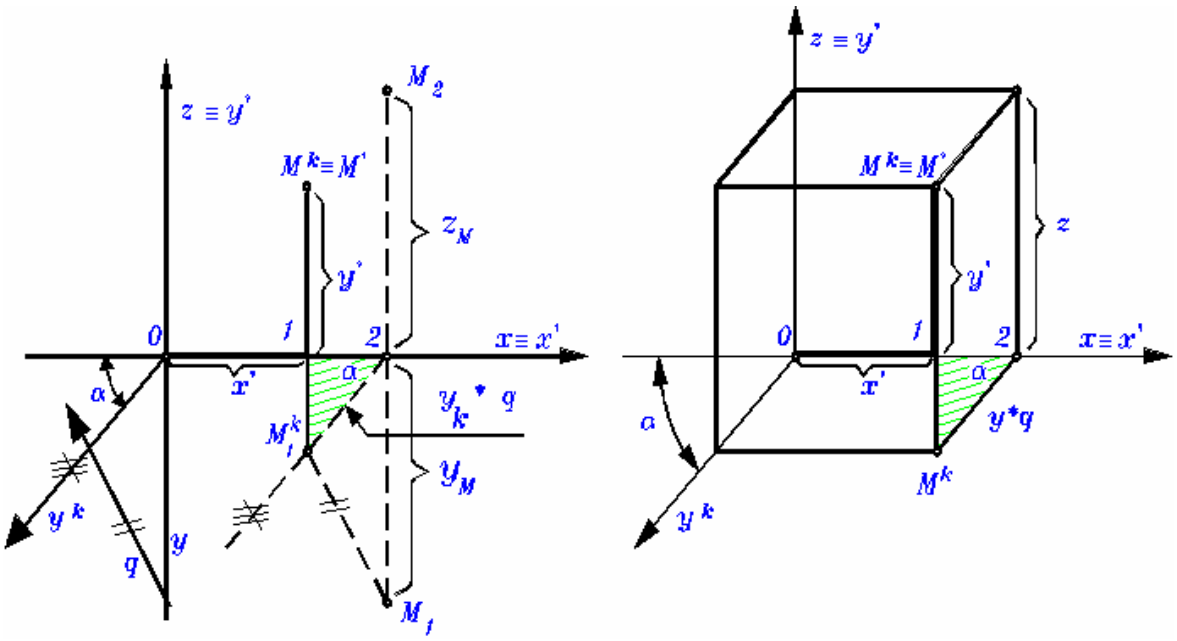
Běžné rozložení sdružených obrazů pro promítání ve třetím kvadrantu je zobrazeno na následujícím obrázku.



Obrázek 7 Obrázek objektu zobrazovaného ve třetím kvadrantu s průmětnami

2.1.2 Kosoúhlé promítání

U tohoto druhu promítání jsou promítací paprsky navzájem rovnoběžné, nejsou však kolmé k průmětně. Průmětna zůstává rovnoběžná s některou z hlavních rovin (rovina *xy*, *yz* nebo *xz*).



Obrázek 8 Ukázka kosoúhlého promítání

Jde o rovnoběžné promítání, kde jedna osa (zpravidla osa y) se promítá pod úhlem, který je kosý vzhledem k ostatním průmětům os a průmětnám. Jednotková délka ve směru této osy je násobena nenulovým číslem (zkrácením) q.

Nákresnu (zobrazovací rovinu $(x'z')$) ztotožníme s rovinou (xz).

Průmět osy z ztotožníme s osou z' zobrazovací roviny, osu x' zobrazovací roviny ztotožníme s osou x souřadnicového systému.[1]

Z toho tedy plyne následující:

$$z \equiv y', x \equiv x' . \tag{6}$$

Potom dle obrázku odvodíme pro x' a y' vztahy:

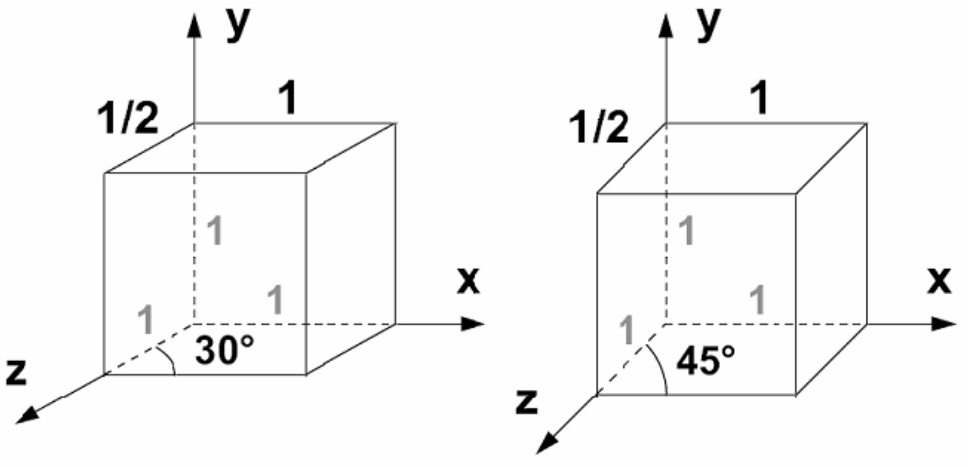
$$x' = x - y.q.\cos(\alpha) , y' = x - y.q.\sin(\alpha) . \tag{7}$$

Pro technické kosoúhlé promítání je $q = \frac{\sqrt{2}}{2}$ a $\alpha = 135°$.

Dva nejčastěji používané druhy kosoúhlého promítání se nazývají *kabinetní* a *kavalírské promítání*.

2.1.2.1 Kabinetní promítání

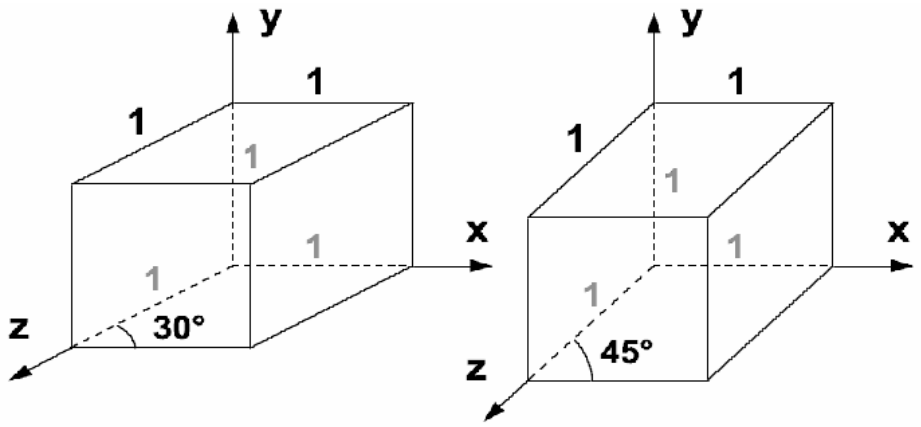
Směr projekce s průmětnou svírá 63,4º (arctan(2)). Úsečky kolmé na průmětnu se zkracují na polovinu.



Obrázek 9 Ukázka kabinetního promítání

2.1.2.2 Kavalírské promítání

Směr projekce s průmětnou svírá 45º. (úsečky rovnoběžné s průmětnou i kolmé na ní mají stejnou délku)



Ukázka kavalírského promítání

2.1.3 Axonometrické promítání

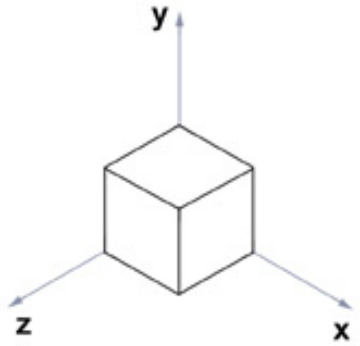
Axonometrické promítání používá projekční roviny, které nejsou rovnoběžné s hlavními osami. Průmětna protíná 2 nebo 3 hlavní osy WCS. Tři úsečky v rovině, které mají společný jeden krajní bod a které neleží v přímce, lze předpokládat za jeden rovnoběžný průmět tří vzájemně kolmých a stejně velkých úseček, které mají jeden krajní bod společný. V souladu s tím je axonometrie často definována pěti hodnotami j_x, j_y, j_z, α, β, kde j_x, j_y, j_z jsou průmětny jednotek na osách x, y, z a α, β jsou úhly, které svírají promítnuté osy x, z s kolmicí na průmět osy y. [2]

Pokud se souřadnicový systém průmětny (osy x, y) zvolí tak, aby průmět j_y ležel na ose y, pak axonometrii popíše matice

$$T_{axon} = \begin{bmatrix} j_x \cos(\alpha) & -j_x \sin(\alpha) & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -j_z \cos(\beta) & -j_z \sin(\beta) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{8}$$

2.1.3.1 Izometrie

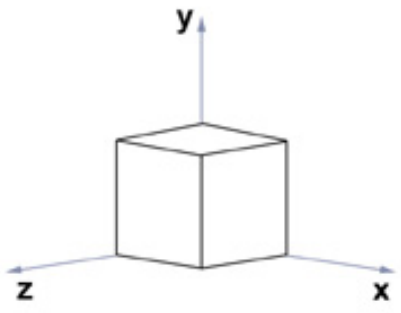
Pokud průmětna protne hlavní osy ve stejné vzdálenosti od počátku WCS (svírá úhel se všemi osami), lze v průmětu měřit a porovnávat vzdálenosti. Zkreslení vzdálenosti je totiž ve všech promítnutých osách stejné. Tato projekce se nazývá *izomerie*. [2]



Obrázek 10 Ukázka izometrie

2.1.3.2 Dimetrie

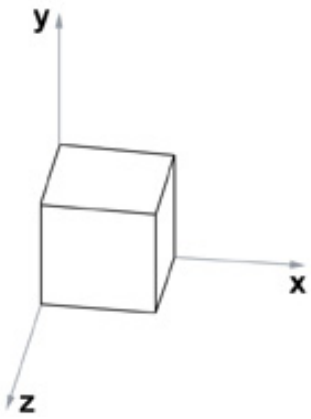
Pokud jsou od počátku ve stejné vzdálenosti pouze 2 průsečíky, lze v průmětu měřit vzdálenosti ve dvou směrech promítnutých os. Ve třetím směru jsou vzdálenosti zkrácené nebo prodloužené. Tento případ se nazývá *dimetrie*. [10]



Obrázek 11 Ukázka dimetrie

2.1.3.3 Trimetrie

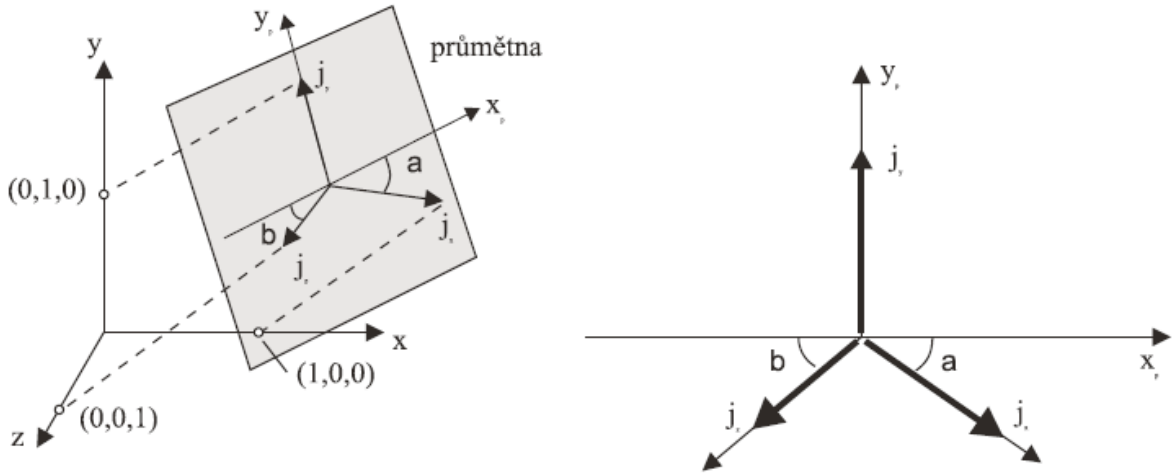
Při obecném sklonu vůči osám hovoříme o *trimetrii*. Zde platí, že v každém směru promítnuté osy je třeba při porovnání uplatnit odlišné měřítko.[10]



Obrázek 12 Ukázka trimetrie

2.1.3.4 Pravoúhlá axonometrie

Axonometrické promítání zachovává rovnoběžnost promítnutých hran a mění úhly mezi hranami. V případě, že je směr promítání kolmý na průmětnu, se jedná o *pravoúhlou axonometrii*. [2]



Obrázek 13 Zadání pravoúhlé axonometrie

Jednotlivé případy axonometrií můžeme dle parametrů rozdělit do následující tabulky.

Tabulka 1 Tabulka druhů axonometrie

izomerie	jx = jy = jz	$\alpha = \beta$
dimetrie	jx = jy	$\alpha = \beta$
trimetre	jx ≠ jy ≠ jz	
kosoúhlé promítání	jy = jz	$\beta = 0$

2.2 Perspektivní promítání

Při tomto způsobu promítání vycházejí všechny promítací paprsky z jednoho bodu, který nazýváme *střed promítání*. Ve výsledném obrazu se sice prostorové úsečky zobrazí opět do úseček v rovině, obecně však není zachována rovnoběžnost. Vzdálenost objektů od středu promítání ovlivňuje velikost jejich průmětů a proto mají vzdálenější objekty menší průměty. [1]

S trochou nadsázky lze říci, že úběžníky reprezentují „nekonečno“, v němž se rovnoběžníky stýkají. Úběžníků můžeme nalézt nekonečné množství, ale pouze ty, které odpovídají hranám osově orientovaných kvádrů, tedy úsečkám rovnoběžným se souřadnicovými osami, se nazývají hlavní. Mají význam v architektuře, protože v ní se používají objekty s hranami a plochami, které jsou rovnoběžné se souřadnicovými osami (podlahy, stěny).

Středové neboli *perspektivní* promítání vytváří obrazy podobné těm, které vidí lidské oko v reálném světě. Vzdálenější objekty jsou menší, původně rovnoběžné úsečky se viditelně sbíhají. Perspektivní promítání tedy respektuje optický model, který vyjadřuje lidské vidění reálného světa. Modeluje proporcionální změnu předmětů při vzrůstající vzdálenosti od pozorovatele a poskytuje dobrý prostorový vjem na rovinné průmětně. Z tohoto důvodu je středové promítání používáno v architektuře a v aplikacích, které kladou důraz na podobnost s reálným světem, jako např. v oblasti virtuální reality. [2]

Nyní odvodíme vztah pro výpočet matice středového promítání se středem v ose *z* a průmětnou v rovině *xy*. Střed promítání [*x, y, z*] leží v takovém případě nad průmětnou ve vzdálenosti d = z a jeho souřadnice zapíšeme zjednodušeně jako [0, 0, d]. Pro daný prostorový bod P_1 o souřadnicích [x, y, z] hledáme souřadnice jeho průmětu P'_1 = [x, y, z].

Promítací paprsek určený středem promítání a bodem P_1 lze zapsat v parametrickém tvaru jako

$$x = x_1 - t.x_1,\ y = y_1 - t.y_1,\ z = z_1 - t.(z_1 - d) \tag{9}$$

Bod P'_1 v průmětně má souřadnici $z'_1 = 0$, takže pro parametr *t* průsečíku promítacího paprsku s průmětnou platí

$$t = \frac{z_1}{z_1 - d}. \tag{10}$$

Po dosazení do rovnic určíme zbylé dvě souřadnice průmětu

$$\left[x'_1, y'_1\right] = \left[x_1 \frac{d}{d - z_1}, y_1 \frac{d}{d - z_1}\right] = \left[x_1 \frac{1}{1 - \frac{z_1}{d}}, y_1 \frac{1}{1 - \frac{z_1}{d}}\right]. \tag{11}$$

Tuto transformaci můžeme zapsat v maticovém tvaru pomocí homogenních souřadnic bodů jako

$$\left[x'_1, y'_1, z'_1, w'\right] = \left[x_1, y_1, z_1\right] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1/d \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{12}$$

Váha w' bodu P'_1 v průmětně není tentokrát rovna jedné, ale nabývá hodnoty

$$\omega' = 1 - \frac{z_1}{d}. \tag{13}$$

Charakteristickým rysem středového promítání je to, že nezachovává rovnoběžnost úseček. Průměty úseček, rovnoběžných v trojrozměrném prostoru, jsou obecně mimoběžné a střetávají se ve středu promítání. Výjimkou jsou prostorové úsečky, ležící v rovině rovnoběžné s průmětnou. [1]

Průmětna může mít libovolnou polohu, ale se rozlišují tři případy, které odpovídají orientaci průmětny vůči osám souřadnicového systému a to *jednobodové*, *dvoubodová* a *trojbodová perspektiva*.

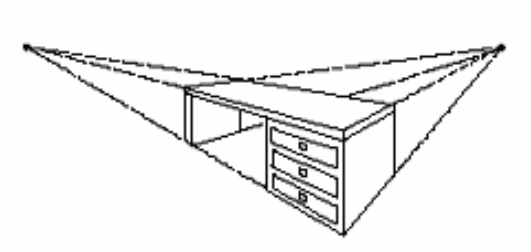
2.2.1 Jednobodová perspektiva

Jednobodová perspektiva vzniká, když průmětna protíná jedinou souřadnicovou osu. Všechny úsečky kolmé na průmětnu míří do jediného bodu, který se nazývá *hlavní úběžník*. Taktéž je tato perspektiva označována jako tzv. *jednoúběžníková perspektiva* či *jednostředová perspektiva*. [2]

Obrázek 14 Jednobodová perspektiva

2.2.2 Dvoubodová perspektiva

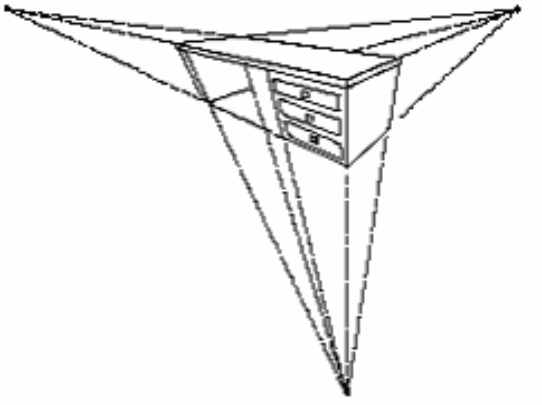
Tato perspektiva vzniká, pokud průmětna protíná dvě ze souřadnicových os. Hrany osově orientovaných kvádrů směrují do dvou hlavních úběžníků. Tento druh promítání bývá nazýván také jako tzv. *dvouúběžníková perspektiva* či *dvoustředová perspektiva*. [2]



Obrázek 15 Dvoubodová perspektiva

2.2.3 Tříbodová perspektiva

Tato projekce je nejobecnější případ, který vzniká, pokud průmětna promítá všechny tři souřadnicové osy. Protažením hran osově orientovaných kvádrů můžeme vysledovat tři hlavní úběžníky. [2]



Obrázek 16 Tříbodová perspektiva

3 TECHNOLOGIE PRO IMPLEMENTACI

Tato kapitola popisuje teoreticky vybrané technologie, které budou použity v praktické části práce a implementovány v programu.

3.1 OpenGL

Knihovna OpenGL (Open Graphics Library) byla navržena firmou SGI (Silicon Graphics Inc.) jako aplikační programové rozhraní (Application Programming Interface - API) k akcelerovaným grafickým kartám resp. celým grafickým subsystémům. OpenGL byla navržena s důrazem na to, aby byla použitelná na různých typech grafických akcelerátorů a aby ji bylo možno použít i v případě, že na určité platformě žádný grafický akcelerátor není nainstalován - v tom případě se použije softwarová simulace. V současné době lze knihovnu OpenGL použít na různých verzích unixových systémů (včetně Linuxu a IRIXu), OS/2, MacOS (X) a na platformách Microsoft Windows. [8]

3.2 SDL a SDL_Image

SDL - "Simple Directmedia Layer" je cross-platformní (přenositelná) knihovna distribuovaná pod licencí GNU LGPL verze 2, která poskytuje přímý přístup k funkcím zvuku, klávesnice, myši, joysticku, 3D akcelerátorům (přes OpenGL) a 2D video bufferu.

Programy s knihovnou SDL můžete přeložit pod operačnímy systémy: Linux, Windows, Windows CE, BeOS, MacOS, Mac OS X, FreeBSD, NetBSD, OpenBSD, BSD/OS, Solaris, IRIX, and QNX. K dispozici je neoficiální podpora systémů: AmigaOS, Dreamcast, Atari, AIX, OSF/Tru64, RISC OS, SymbianOS, and OS/2. Knihovna je napsaná v jazyce ANSI C a použitelná tím pádem i v C++. Dá se však použít i v dalších jazycích, jako jsou: Ada, C#, Eiffel, Erlang, Euphoria, Guile, Haskell, Java, Lisp, Lua, ML, Objective C, Pascal, Perl, PHP, Pike, Pliant, Python, Ruby, and Smalltalk.

Základní funkce SDL umožňují: inicializovat grafiku, přistupovat k pixelům, vykreslit bitmapový obrázek, přehrát zvuk, číst události z klávesnice a joysticku a pracovat s vlákny.

Kromě základních funkcí v SDL jsou k dispozici také rozšiřující knihovny a jednou z nich je i SDL_Image. Samotná knihovna SDL umožňuje načítat obrázky pouze z grafického formátu BMP. Knihovna SDL_Image umožňuje do grafického surface knihovny SDL načíst mnoho dalších grafických formátů a to formáty: TGA, BMP, PNM, XPM, XCF, PCX, GIF, JPG, TIF, LBM a PNG. [8]

3.3 Formát ASE

Formát modelů ASE je ASCII (textový) formát programu 3D Studio Max. Dá se do něj ovšem exportovat také v programu ActorX, Maya nebo dalších (některé až s exportním pluginem).

Formát je založen na identifikátorech a to ve formě řetězců *HVĚZDIČKA_PAK_JMÉNO, které jsou následovány daným počtem hodnot oddělených whitespacey (mezery nebo odřádkování). Dále je zde několik identifikátorů obklopených složenými závorkami. Každý identifikátor začíná na novém řádku. [13]

3.4 FreeType fonty

Použitím knihovny FreeType Font rendering library můžeme snadno vypisovat vyhlazené znaky, které vypadají mnohem lépe než písmena u bitmapových fontů. Základní problém s použitím bitmapových fontů je, že OpenGL bitmapy jsou binárními obrázky. To znamená, že si OpenGL pamatuje pouze 1 bit na 1 pixel. Přiblížíme-li text, výsledek vypadá přibližně takto:

Obrázek 17 Bitmapový (vpravo) a vyhlazený FreeType font (vlevo)

Okolo okrajů FreeType fontů se nachází spousta šedých pixelů, které jsou typické pro vyhlazené (anti-aliasované) fonty. Šedé pixely vylepšují znaky při pohledu z dálky. [7]

II. PRAKTICKÁ ČÁST

4 VÝVOJ PROGRAMU

Program byl napsán v jazyce C++. Pro vývoj programu jsem používal nejprve freewareové vývojové prostředí Dev-C++. Později jsem přešel na (taktéž freewareové) prostředí Code::Blocks. Rozhodl jsem se tak z důvodu mnohem větší vyspělosti tohoto prostředí (možnost schovávání bloků kódu pro větší přehlednost, lepší práce s debugerem atd.) a také z toho důvodu, že Dev-C++ se již oficiálně nevyvíjí.

4.1 Výběr knihoven

Kromě výběru vývojového prostředí, ve kterém budeme program psát, je před samotným vývojem aplikace velice důležitou otázkou také použití grafických knihoven. Knihovny jsem se nažil vybírat tak, aby měly podporu akcelerované 3D grafiky, byli dostatečně široce používané a vyvinuté, byli pokud možno použitelné pro co největší počet platforem a hlavně nebyly příliš komplikované na pochopení a použití při programování.

Knihovnu SDL jsem si vybral nejen z důvodu jednoduchosti jejího použití, ale také díky tomu, že s ní již mám drobné zkušenosti. Knihovna má navíc velkou výhodu díky své obrovské přenositelnosti. Inicializace grafiky a vytvoření okna s podporou OpenGL je u ní otázkou zavolání dvou funkcí, čímž má v jednoduchosti oproti většině knihoven velký náskok.

Rozšiřující knihovnu SDL_Image jsem použil z důvodu použitelnosti většího množství formátů, jelikož podpora pouze formátu BMP mi přišla nedostatečná.

Při výběru knihovny pro zobrazování fontů jsem se rozhodoval, jestli použít bitmapové fonty, outline (3D bitmapové) fonty nebo FreeType fonty. Vzhledem ke kvalitě vyhlazování jsem se nakonec rozhodl použít knihovnu Freetype 2 s třídou freetype staženou z www.nehe.ceske-hry.cz.

4.2 Vytvoření projektu a přilinkování knihoven

Na internetových stránkách www.devpaks.org jsem si nejprve vyhledal tzv. devpaky (tj. instalační balíčky knihoven pro Dev-C++) pro SDL, SDL_image a Freetype 2, OpenGL. Poté jsem tyto knihovny nainstaloval a přilinkoval tak, že jsem do vlastností projektu (přesněji parametru linkeru) přidal parametry dle následující tabulky.

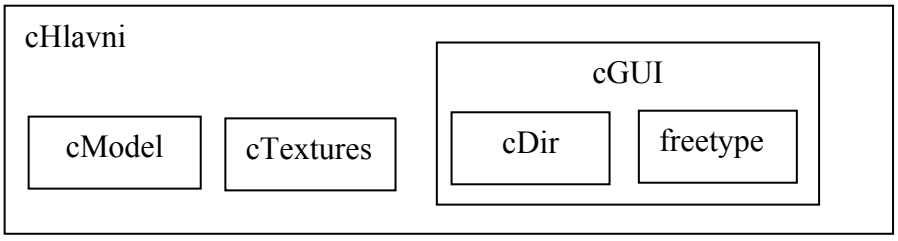
Tabulka 2 Tabulka parametrů přilinkování knihoven

Knihovny	**Parametry linkeru**
SDL	-lSDLmain -lSDL
SDL_Image	-lSDL_image
Fretype 2	-lfreetype
OpenGL	-lopengl32 -lGLU32

Poté jsem ještě v Dev-C++ ve vlastnostech projektu nastavil cestu k mým hlavičkovým souborům.

Importování projektu do Code::Blocks bylo velmi jednoduché, neboť Code::Blocks má funkci pro importování projektů ve formátu DEV z Dev-C++. Po importování bylo nutné ještě znovu nainstalovat všechny devpaky, které naštěstí také podporuje. Knihovny zůstanou po importu přilinkované. Po dokončení importu se projekt uloží ve formátu CBP.

4.3 Rozvržení projektu do tříd



Obrázek 18 Diagram tříd programu

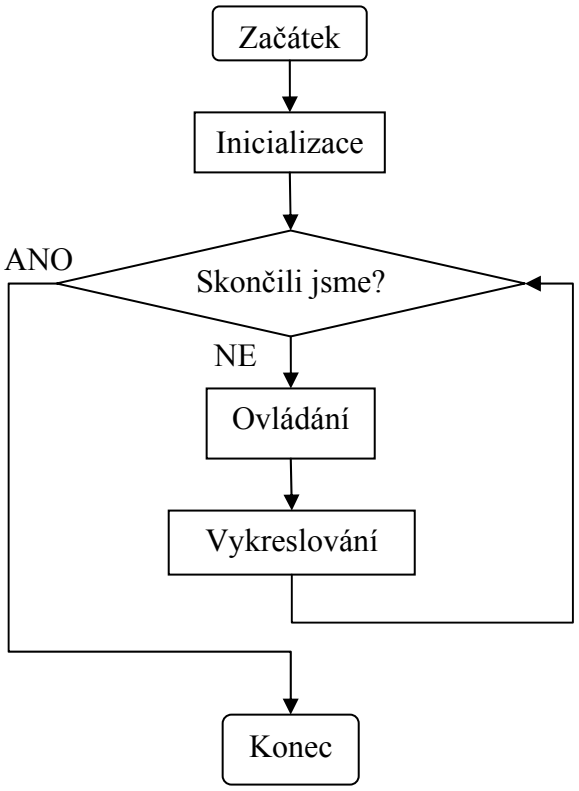
Projekt jsem rozdělil do několika tříd provázaných agregací. Hlavní třídou je třída *cHlavni*, která má svoji instanci umístěnou přímo ve funkci *main*. V ní je agregovaná třída *cModel* pro načítání a vykreslování zvoleného modelu, třída *cGUI* pro práci s uživatelským rozhraním, *cTextures* pro načítání textur a jejich mapování na model a na ovládací prvky. V třídě *cGUI* se ještě pracuje s instancí třídy *cDir*, která slouží pro zjištění adresářů

v souborech. Třída freetype pochází z knihovny Freetype.h stažené z nehe.ceskehry.cz a jedná se nadstavbu knihovny Freetype 2.

4.3.1 Třída cHlavni

Třída *cHlavni* je hlavní třída programu. Stará se o inicializaci grafiky, vytvoření okna, změnu rozměrů okna, volá vykreslování jednotlivých prvků scény v různých druzích promítání a především hlavní smyčku, ve které tyto metody volá.

4.3.1.1 Hlavní smyčka programu



Obrázek 19 Vývojový diagram hlavní smyčky programu

Hlavní smyčka je jádrem celého programu. Nejprve se provede inicializace proměnných a grafiky a poté se jde do smyčky. Ve smyčce nejprve ošetříme události klávesnice a myši a poté vykreslíme scénu. Ukončení smyčky je dáno příslušnou událostí (zavření okna, klávesa Esc)

4.3.1.2 Vytvoření okna aplikace

Při spuštění programu se nejdříve zavolá konstruktor. Ten zavolá funkci *Init*. Funkce *Init* vrací při úspěšné inicializaci hodnotu true, respektive při neúspěchu false. Jako první se ve funkci *Init* vykoná inicializace grafiky SDL funkcí *SDL_Init*. Poté nastavíme atributy kontextu renderingu pro OpenGL (tj. zapneme doublebuffering, nastavíme buffery, zapneme celoobrazovkový antialiasing atd.). Poté funkcí *SDL_SetVideoMode* vytvoříme okno o zadaných rozměrech a barevné hloubce s podporou OpenGL a možností změny rozměrů.

Dále zavoláme funkci *InitGL*, ve které zavoláme funkci pro načtení textur, nastavíme barvu pozadí, zapneme blending (průhlednost objektů ve scéně), nastavíme testování hloubky a perspektivu a zavoláme funkce pro inicializaci uživatelského rozhraní a fontů.

Funkce *Init* ještě provede nastavení kláves pro 300ms prodlevy před opakovanou akcí po události klávesnice. Zavolá se funkce *ResizeGL* (viz. dále) a nastaví titulek okna. Nakonec zavoláme funkci pro načtení modelu ze souboru a provedeme inicializaci všech proměnných.

4.3.1.3 Změna velikosti okna

Změna velikosti okna se provádí po události *SDL_VIDEORESIZE*. Pokud jsme povolili resizing při inicializaci, tak provedeme opětovné vytvoření okna funkcí *SDL_SetVideoMode* s novými rozměry získanými ze struktury *SDL_Event*, přesněji z položek *resize.w* a *resize.h*. V každém případě se poté zavolá funkce *ResizeGL*.

V ní nejprve nastavíme tzv. viewport, což je plocha, na kterou budeme vykreslovat scénu z OpenGL. V našem případě bude mít identické rozměry jako má celé okno. Zvolíme projekční matici a uděláme z ní jednotkovou. Dále pomocí funkce *gluPerspective* nastavíme perspektivní promítání s úhlem pohledu 45° a současným poměrem rozměrů okna a ořezávací roviny. Posléze zvolíme matici modelu a z té uděláme taktéž jednotkovou. Nakonec předáme nové rozměry okna do proměnných pro výpočet polohy prvků uživatelského rozhraní.

4.3.1.4 Ovládání

Ovládání je implementováno v metodě *ProcessEvent*, která se nachází ve zdrojovém souboru *ovladani.h*. V této funkci jsou všechny následující příkazy ve smyčce událostí. Ukončení smyčky určuje funkce SDL_PollEvent s parametrem adresy událostí.

Ovládání je větveno pro všechny druhy promítání a je tedy specifické pro každý druh promítání.

4.3.1.5 Úprava promítání

Výběr promítání je implementován v metodě *Draw*. V této funkci nejprve „vyčistíme“ obrazovku zavoláním funkce glClear a nastavíme matici modelu na jednotkovou. Následuje podmíněné větvení pro výčtový typ značící druh promítání.

Tabulka 3 Tabulka hodnot výčtového typu tPromitani

Výčtová hodnota	**Význam v programu**
rovnobezne_stredove	Porovnání rovnoběžného a středového promítání
mongeovkaE	Mongeova projekce dle normy ISO E
mongeovkaA	Mongeova projekce dle normy ISO A
kosouhle	Ukázka kosoúhlého promítání
axonometrie1	Ukázka pravoúhlé axonometrie
axonometrie2	Ukázka izometrie
axonometrie3	Ukázka dimetrie
axomentrie4	Ukázka trimetrie
stredove2	Ukázka dvoubodové perspektivy
stredove3	Ukázka tříbodové perspektivy

U všech těchto typů promítání se snažíme na začátku umístit současnou matici dočasně do zásobníku funkcí *glPushMatrix*, provést transformace (translaci, rotaci, změnu měřítka a úpravu promítání) a po vykreslení ze zásobníku vyjmout.

U porovnání rovnoběžného a středového promítání vykreslíme nejprve model ve středovém promítání a poté se posuneme o vzdálenost, kterou udává proměnná *spacing* a vykreslíme model v kolmém rovnoběžném promítání tak, že před kreslením zavoláme funkci glOrtho s parametry okna a ořezávací roviny. Po vykreslení se vrátíme zpět do středového promítání jen vyjmutím předchozí projekční matice ze zásobníku.

Stejný postup také používáme i u obou rovnoběžných projekcí s tím, že mezi sebou necháme vykreslit více pootočených modelů.

U kosoúhlé projekce a axonometrií také nejprve přejdeme do rovnoběžného promítání. Ovšem zde navíc ještě řešíme zkosení a změnu měřítka v jednotlivých osách pomocí modifikace projekční matice vynásobením maticí pro změnu měřítka v osách

$$T_{Sc} = \begin{bmatrix} S_X & 0 & 0 & 0 \\ 0 & S_Y & 0 & 0 \\ 0 & 0 & S_Z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{14}$$

a zkosení v patřičné rovině

$$T_{Shyz} = \begin{bmatrix} 1 & SHy & SHz & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, T_{Shxz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ SHx & 1 & SHz & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, T_{Shxy} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ SHx & SHy & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{15}$$

U dvojbodové a tříbodové perspektivy vyjdeme z rovnoběžného promítání a vynásobíme příslušnou projekční matici maticí pro jednostředovou perspektivu

$$T_{P1b} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \frac{1}{d_z} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{16}$$

pro dvojstředovou perspektivu

$$T_{P2b} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & \frac{1}{d_y} \\ 0 & 0 & 1 & \frac{1}{d_z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{17}$$

nebo pro trojstředovou

$$T_{P3b} = \begin{bmatrix} 1 & 0 & 0 & \frac{1}{d_x} \\ 0 & 1 & 0 & \frac{1}{d_y} \\ 0 & 0 & 1 & \frac{1}{d_z} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (18)$$

4.3.2 Třída model

Tato třída se stará o načítání modelu ze souboru ve formátu ASE a vykreslování modelu přes funkce OpenGL.

Tabulka 4 Tabulka použitých vlastností modelu

Řetězec v souboru ASE	Atribut třídy cModel	Význam proměnné
*MESH_NUMVERTEX	**int** pocet_vertexu;	Počet vertexů (vrcholů) modelu
*MESH_NUMFACES	**int** pocet_faceu;	Počet trojúhelníků (face)
*MESH_VERTEX	**tSouradnice** * vertex;	Dynamické pole souřadnic vertexů
*MESH_FACE	**tTrojuhelnik** * face;	Dynamické pole indexů vrcholů trojúhelníků
*MESH_NUMTVERTEX	**int** pocet_koordinatu;	Počet texturových koordinátů
*MESH_TVERT	**tKoordinaty** * koordinaty;	Dynamické pole texturových koordinátů
*MESH_NUMTVFACES	**int** pocet_koordinatu;	Počet otexturovaných trojúhelníků
*MESH_TFACE	**tTrojuhelnik** * tface;	Otexturované vertexy

4.3.2.1 Načtení dat ze souboru

Ve funkci *nacti* program otevře textový soubor pro čtení a porovnává postupně řetězce v souboru funkcí strcmp ze standardní knihovny string.h jazyka C. Pokud je řetězec nalezen, načtou se následující čísla ze souboru do patřičného atributu dle tabulky 4.

4.3.2.2 Vykreslení modelu

Vykreslování modelu je realizováno ve funkci *kresli*. Nejprve nabindujeme příslušnou texturu, pokud byla vybrána. Pokud textura vybrána nebyla, použijeme zelenou barvu a model po vykreslení ještě „obtáhneme“ zelenou čarou. Model budeme vykreslovat po jednotlivých trojúhelnících. Na začátku každého vykreslovaného vrcholu nejprve zavoláme funkci *glTexCoord2f*, kterou naneseme na daný trojúhelník kus textury o daných souřadnicích, které jsme si předtím načetli do pole texturových koordinátů, pokud jsme samozřejmě použili texturu. Až poté zavoláme funkci *glVertex3f* s parametry z pole vertexů, kterými určíme umístění vrcholů trojúhelníků v prostoru.

4.3.3 Třída cTextures

Tato třída obsahuje dvě metody. První metoda slouží k načtení obrázku ze souboru do grafického surface knihovny SDL. Druhá metoda z tohoto surface vytvoří texturu v OpenGL a tento surface uvolní z paměti.

4.3.3.1 Načtení obrázků ze souboru

Ve funkci *LoadBitmap* nejprve načteme obrázek ze souboru funkcí *IMG_Load* z knihovny SDL_Image a dále budeme pracovat s ukazatelem na tento obrázek v paměti (na strukturu *SDL_Surface*). Načtený obrázek je v surface zrcadlově otočený, takže jej nejprve musíme otočit do správné polohy. Vytvoříme si dočasné dynamické pole celých čísel tmpbuf, zjistíme si pozici posledního a prvního řádku obrázku a začneme každý řádek symetricky přehazovat standardní funkcí *memcpy* a stejně tak přehazujeme barevnou složku pro každý pixel řádků. Po každé této operaci se samozřejmě posouváme ukazatelem na řádku. Po skončení cyklu uvolníme *tmpbuf* z paměti a vrátíme ukazatel na opravený surface.

4.3.3.2 Vytvoření textur

Po úspěšném provedení funkce *LoadBitmap* vygenerujeme číslo textury a poté vygenerujeme samotnou texturu funkcí *glTexImage2D* s rozměry, které získáme ze struktury *SDL_Surface*. Dále nastavíme filtrování textury na lineární. Před uzavřením této metody ještě uvolníme paměť surfaceů a v případě, že vše proběhlo úspěšně, vrátíme *true*.

4.3.4 Třída cGUI

Tato třída slouží k obsluze a vykreslování grafického uživatelského rozhraní. To se skládá z ovládacího panelu, několika výpisů a jednoduchého výběru modelu a textury. Je zde několik metod. Metoda *drawMatrix* vypisuje projekční matici za pomoci fontů a knihovny Freetype. Funkce *drawOsy* vykresluje barevné osy skládající se z přímek a kuželů a funkce drawText vykresluje veškerý zbylý text (parametry promítání,). Dále je zde několik drobných funkcí pro zjištění polohy myši.

4.3.5 Třída cDir

Tato třída zjišťuje a vypisuje obsah adresáře models v hlavním adresáři programu.

4.3.5.1 Zjištění obsahu adresáře

Obsah adresáře se zjišťuje v metodě *init*. Ke zjišťování obsahu adresáře jsem se rozhodl použít standardní knihovnu jazyka C dirent.h, se kterou již mám trochu zkušeností. Nejprve pomocí funkce *opendir* otevřeme adresář. Poté jej prohledáme ve smyčce čtení, ve které voláme funkci *readdir*, které nám zjišťuje jednotlivé položky adresáře a ukládá do struktury *dirent*. Název souboru poté získáme z položky d*_name*, což je již pole znaků.

4.3.5.2 Vypsání obsahu adresáře

Adresář vypíšeme na obrazovku funkcí standardně *print* ze třídy *freetype*. Vypsání proběhne za podmínky, že jsme otevřeli výběrové menu. Pokud jsme ještě žádný model nevybrali, vypíšeme „Model nevybran“ a to samé provedeme i v případě textury.

5 UŽIVATELSKÝ MANUÁL PROGRAMU

5.1 Ovládání programu

Po spuštění programu by jsme měli nejprve vybrat model. To provedeme pomocí nabídky, která se objeví kliknutím na nápis „Model nevybrán“. Po jejím vysunutí stačí model kliknutím ne jeho název vybrat. Obdobně funguje i výběr textury. Model se nám zobrazí až poté, kdy vybereme druh protínání. Ten se vybírá pomocí číselných kláves nad písmennou částí klávesnice dle následující tabulky.

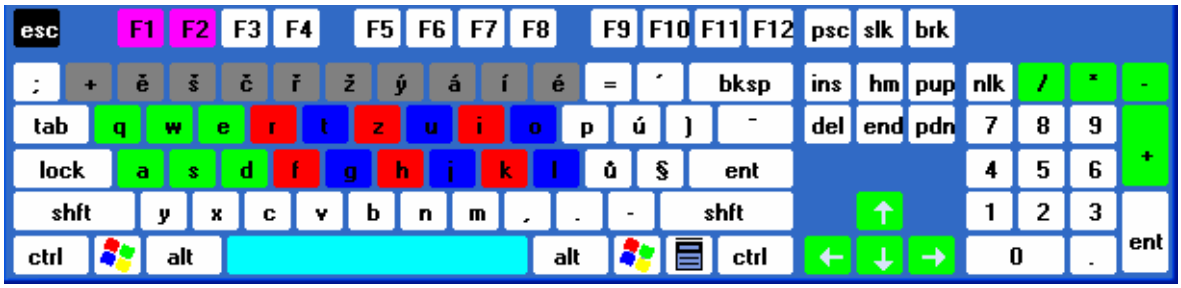
Tabulka 5 Tabulka číselných kláves pro výběr promítání v programu

Klávesa	**Druh zobrazeného promítání**
1	Porovnání rovnoběžného a středového promítání
2	Mongeova projekce dle normy ISO E
3	Mongeova projekce dle normy ISO A
4	Ukázka kosoúhlého promítání
5	Ukázka pravoúhlé axonometrie
6	Ukázka izometrie
7	Ukázka dimetrie
8	Ukázka trimetrie
9	Ukázka dvoubodové perspektivy
0	Ukázka tříbodové perspektivy

V programu se také zobrazuje ovládací panel, který je možno skrýt klávesou F2. Klávesou F1 můžeme skrýt také všechny výpisy na obrazovku a bude se zobrazovat pouze model.

5.1.1 Pohyb modelu klávesnicí

Pro otáčení modelu kolem osy x slouží klávesy W *a* S, pro otáčení kolem osy z Q *a* E a pro rotaci kolem osy y A a D. Posouvání modelu na osách x a y je možné pomocí šipek a na ose z klávesami plus a mínus na numerické klávesnici. Ke zvětšování a zmenšování modelu můžeme použít klávesy hvězdička a lomeno. Resetování na původní pozici můžeme provést mezerníkem.



Obrázek 20 Funkční klávesy programu

Veškeré ovládání programu je shrnuto na obrázku 18. Fialově jsou zvýrazněny klávesy pro skrývání okolního grafického rozhraní, šedou barvou jsou klávesy pro přepínání projekcí, červeně se mění parametry úhly a modře vzdálenosti u parametrů projekcí. Zeleně označené klávesy slouží pro pohyb modelu v prostoru a bledě modrým mezerníkem můžeme resetovat pozici modelu. Program lze také ukončit klávesou Esc.

5.1.2 Pohyb modelu myší

Při stisknutí levého tlačítka myši můžeme modelem rotovat kolem os *x* a *y* a při stisknutém pravém tlačítku myši můžeme model posouvat v osách *x* a *y*. Kolečkem myši ještě můžeme model zvětšovat a zmenšovat. V programu je také zobrazován ovládací panel, kde můžeme kliknutím myši na šipky na levém kruhu pootáčet modelem a na pravém kruhu posouvat model podél os *x* a *y*.

Obrázek 21 Ovládací panel programu

5.1.3 Změna parametrů promítání

Změnu parametrů můžeme provést dvěma způsoby a to myší nebo klávesnicí. U některých druhů promítání můžeme měnit všech 6 parametrů (trimetrie), u některých 2 (kosoúhlé promítání, isometrie) či 4 (dimetrie) a u některých žádný (pravoúhlé promítání, porovnání pravoúhlého a středového promítání). Pro změnu (zmenšení nebo zvětšení) úhlu nebo vzdálenosti stačí kliknout levým tlačítkem myši na patřičnou šipku na ovládacím panelu. Pokud chceme udělat změnu přesněji, musíme použit klávesy zvýrazněné na obrázku 18 červenou (úhly) nebo modrou barvou (vzdálenosti).

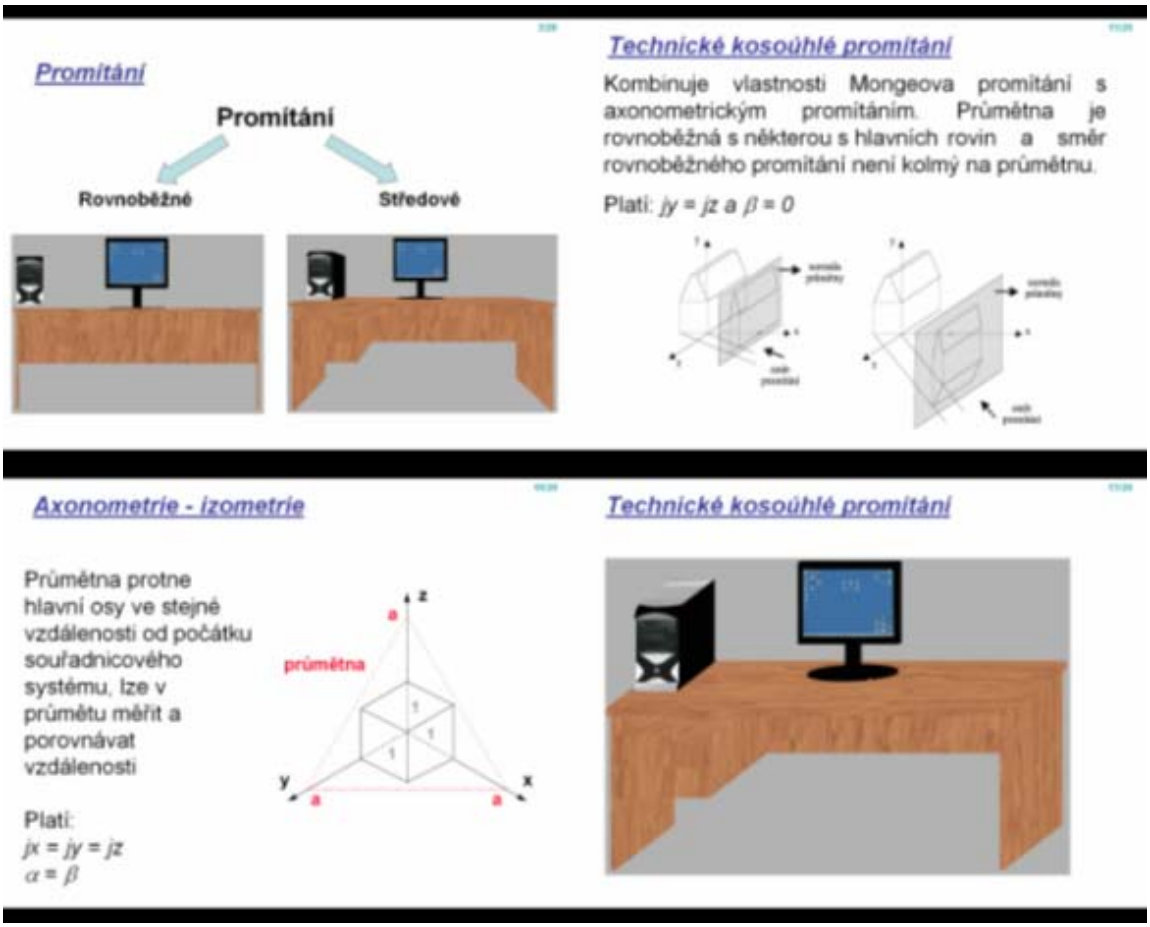
5.2 Podporované modely a textury

Program podporuje modely ve formátu ASE. Model v tomto formátu můžeme vyexportovat např. v programech 3D Studio Max, Maya, ActorX nebo také např. v Blenderu s nainstalovaných pluginem pro tento export. Model musí v každém případě obsahovat pouze jeden ucelený geometrický objekt. V případě modelu, který se skládá z více objektů, musíme tyto objekty před exportem sjednotit. Takto vytvořené soubory můžeme umístit do adresáře models, který se nachází v hlavním adresáři programu. Odtud pak program může model načíst do nabídky a zobrazit.

Textury díky použití knihovny SDL_image mohou být v téměř libovolném formátu. Podporovány jsou formáty TGA, BMP, PNM, XPM, XCF, PCX, GIF, JPG, TIF, LBM a PNG. Textury jsou prověřeny v rozlišení maximálně 1024 x 1024 pixelů a funguje korektně na všech testovaných počítačích. V rozlišení 2048 x 2048 pixelů již dochází u některých starších grafických karet k výraznému snížení rychlosti aplikace a v některých případech se nemusí takováto textura vůbec zobrazit. Textury musí mít každopádně čtvercové rozměry (stejný počet pixelů v obou délce i šířce) a tyto rozměry musí být ve tvaru 2^n , kde n = 1, 2, ..., 11. Program může mít s jinými nestandardními rozlišeními problémy.

6 PREZENTACE PRO VÝUKU

Prezentaci jsem vytvořil podle zadaní v programu MS PowerPoint. Popsal jsem v ní pro potřeby výuky stručně druhy promítání, které jsem rozebral podrobněji v teoretické části práce a doplnil ke každému druhu promítání obrázek z mého programu. Tímto jsem splnil poslední bod zadání bakalářské práce.



Obrázek 22 Ukázka prezentace pro výuku

ZÁVĚR

Seznámil jsem se s druhy promítání, které se v současné počítačové 3D grafice používají. V teoretické části jsem podrobně charakterizoval a vysvětlil principy všech rovnoběžných i středových druhů promítání. U složitějších druhů promítání jsem odvodil výpočty jejich transformací a také jsem u všech promítání pro přiblížení přiložil několik názorných obrázků, ze kterých si lze jednotlivé druhy jednoduše představit.

Dále jsem v praktické části provedl návrh aplikace založené na OpenGL. Program jsem se rozhodl vytvořit objektově a rozdělil jsem jej do několika tříd. Z těchto tříd jsem v praktické části tohoto textu popsal princip nejdůležitějších metod pro chod programu.

Aplikaci jsem naprogramoval v jazyce C++ ve vývojových prostředích Dev-C++ a Code::Blocks. Při tvorbě tohoto programu jsem využil znalosti získané v teoretické části a vytvořil ukázky všech probraných druhů promítání na různých modelech. Do programu je taktéž možno importovat vlastní modely dle výše popsaného postupu a volit již vytvořené nebo i vlastní textury pokrývající tyto modely. Aplikace má jednoduché grafické uživatelské rozhraní a intuitivní ovládání, kterým je umožněno měnit pozici a velikost modelu v prostoru, modifikovat parametry jednotlivých druhů promítání pomocí klávesnice i myši.

Program jsem také úspěšně otestoval na řadě počítačů ve škole (včetně toho v přednáškové místnosti) i mimo ni a to včetně až 6 let starých počítačů a mohu konstatovat, že je program velice nenáročný na hardware a s programem jde pracovat bez problémů na všech mnou testovaných počítačových sestavách. Program je také možné díky zvoleným technologiím zkompilovat i na spoustě jiných operačních systémech než je Windows.

Poté jsem napsal v dalším úseku praktické části uživatelskou dokumentaci programu. Vysvětlil jsem, jak se program ovládá a jaké jsou jeho možnosti co do podpory různých grafických formátů modelů a textur.

Na základě teoretických poznatků jsem nakonec vytvořil prezentaci v programu MS PowerPoint, kterou je možno využít při výuce předmětu Počítačová grafika. Prezentaci jsem obohatil o obrázky vytvořené vyvinutým programem.

Doufám, že mnou vytvořená práce povede ke zkvalitnění výuky předmětu Počítačová grafika a pomůže tak studentům lépe pochopit základy problematiky 3D počítačové grafiky.

ZÁVĚR V ANGLIČTINĚ

I got acquainted with the types of projection which are used in the current computer 3D graphics. In the theoretical part, I characterized and explained the principles of all the parallel and central types of projection in detail. In more complex types of projection, I derived calculations of their transformations, and I also attached several illustrative pictures in all types of projection, from which the individual types can be imagined easily.

Furthermore, I performed a design of an application base on OpenGL in the practical part. I decided to create the program as an object-like, and divided it into several classes. In the practical part of this text I described the principle of the most important methods for the program operation from these classes.

I programmed the application in the C++ language in the development means of Dev-C++ and Code::Blocks. In creation of this program I used the knowledge gained in the theoretical part and I created illustrations of all of the given types of projection on various models. It is also possible to import own models into the program in accordance with the above-described procedure and to select already created or even own textures covering these models. The application has simple graphic interface and intuitive control by which the position and size of the model may be changed in space and parameters of individual types of projection may be modified through the use of a keyboard and mouse.

I also tested the program successfully on a series of the computers at school (including the one in the lecture room) and outside the school, and that is including the computers up to 6 years old, and I can state that the program is very undemanding on hardware. It is also possible to compile the program on a number of operational systems other than Windows thanks to used technology.

Afterwards, I wrote the users' program documentation in the next section of the practical part. I explained how the program is to be controlled and what its possibilities are when it comes to supporting various graphic formats of the models and textures.

On the basis of the theoretical pieces of knowledge, I finally created presentation in the MS PowerPoint application, which can be used in teaching the subject Computer Graphics. I enriched the presentation with the illustrations created by the developed program.

I hope that the work made by me will lead to improvement of teaching the Computer Graphics.

SEZNAM POUŽITÉ LITERATURY

[1] ŽÁRA, Jiří, BENEŠ, Bedřich, FELKEL, Petr. *Moderní počítačová grafika*. 1. vyd. Praha : Computer Press, 2005. 448 s. ISBN 80-7226-049-9.

[2] POKORNÝ, Pavel. *Základy počítačové grafiky*. 1. vyd. Zlín : Univerzita Tomáše Bati ve Zlíně, Fakulta Technologická, 2004. 120 s. ISBN 80-7318-161-4.

[3] MARTIŠEK, Dalibor. *Matematické principy grafických systémů*. 1. vyd. Brno : Littera, 2002. 278 s., 1 CD-ROM. ISBN 80-85763-19-2.

[4] SHREINER, Dave, et al. *OpenGL - Průvodce programátora*. 1. vyd. Praha : Computer Press, 2006. 679 s. ISBN 80-251-1275-6.

[5] NOVÁK, Ludvík, et al. *Algebra a geometrie*. 3. opravené vyd. Zlín : Univerzita Tomáše Bati ve Zlíně, Fakulta Technologická, 2003. 126 s. ISBN 80-7318-083-9.

[6] KLETEČKA, Jaroslav, FOŘT, Petr. *Technické kreslení*. 1. vyd. Praha : Computer Press, 1999. 193 s. ISBN 80-7226-192-4.

[7] LIBERTY, Jesse. *Naučte se C++ za 21 dní*. 1. vyd. Praha : Computer Press, 2002. 80-7226-774-4 s., CD-ROM. ISBN 80-7226-774-4.

[8] *CZ NeHe OpenGL* [online]. c2002-2007 [cit. 2007-05-18]. Dostupný z WWW: <http://nehe.ceske-hry.cz/>.

[9] ŠTUGEL, Juraj. *Vyuka pocitacovej grafiky* [online]. 2001 [cit. 2007-05-19]. Dostupný z WWW: <http://www.vpg.host.sk/>.

[10] Univerzita Palackého v Olomouci, katedra Matematické informatiky. *Promítání - Wikiknihy* [online]. 2004 , 4.12.2006 [cit. 2007-05-19]. Dostupný z WWW: <http://cs.wikibooks.org/wiki/Prom%C3%ADt%C3%A1n%C3%AD>.

[11] KOHL, Nate. *C/C++ Reference* [online]. 8.5.2007 [cit. 2007-05-19]. Dostupný z WWW: <www.cppreference.com >.

[12] BÍLEK, Petr. *Sally - Programování v jazyku C/C++* [online]. 2003-2006 [cit. 2007-05-19]. Dostupný z WWW: <http://www.sallyx.org/sally/c/>.

[13] *UnrealWiki: ASE File Format* [online]. 27.11.2006 [cit. 2007-05-19]. Dostupný z WWW: <http://wiki.beyondunreal.com/wiki/ASE_File_Format>.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

C	Programovací jazyk ANSI C.
C++	Programovací jazyk ISO C++ vzešlý z jazyka ANSI C.
Jx	Jednotková vzdálenost v ose x
jx	Jednotková vzdálenost v ose z
Jy	Jednotková vzdálenost v ose y
OpenGL	Open Graphics Library
Q	Zkrácení v ose
RotXx	Rotace osy x ve směru osy xy
RotXy	Rotace osy x ve směru osy y
RotXz	Rotace osy x ve směru osy z
RotYy	Rotace osy y ve směru osy x
RotYy	Rotace osy y ve směru osy y
RotYz	Rotace osy y ve směru osy z
RotZx	Rotace osy z ve směru osy x
RotZy	Rotace osy z ve směru osy y
RotZz	Rotace osy z ve směru osy z
SDL	„Simple Directmedia Layer“ – přenositelná knihovna, poskytuje přímý přístup k funkcím zvuku, klávesnice, myši, joysticku, 3D akcelerátorům (přes OpenGL) a 2D video bufferu.
T	Transformační matice
x	Osa x světového souřadnicového systému
y	Osa y světového souřadnicového systému
z	Osa z světového souřadnicového systému

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH