PŘÍRODOVĚDECKÁ FAKULTA UNIVERZITY PALACKÉHO
KATEDRA INFORMATIKY

BAKALÁŘSKÁ PRÁCE

Výukový program pro geometrii (OpenGL)

2010 Branislav Šimala

Anotace

Cílem práce je vytvořit výukový program, který by názorně předvedl základní pojmy afinní geometrie. Zejména afinní souřadnice, matice přechodu mezi afinními bazemi, matice afinních transformací, projektivní prostory a perspektivní promítání.

Děkuji RNDr Peterovi Sebestyénovi Ph.D za vedení práce a čas věnovaný při konzultacích. Dále děkuji RNDr Michalovi Krupkovi Ph.D za cenné rady a připomínky k práci.

Obsah

1. Motivace **7**

2. Teoretický úvod **8**
2.1. Afinní prostory 8
2.2. Afinní báze a souřadnice 9
2.3. Matice přechodu mezi afinními bazemi 9
2.4. Euklidovské prostory 10
2.5. Matice afinních transformací 11
2.5.1. Afinita 11
2.5.2. Translace 12
2.5.3. Rotace 12
2.5.4. Změna měřítka 13
2.5.5. Speciální transformace 14
2.5.6. Skládání transformací 14
2.6. Projektivní prostory 14
2.7. Perspektivní promítání 15
2.8. Grafická knihovna OpenGL 16

3. Programátorská dokumentace **18**
3.1. Použité technologie 18
3.2. Nastavení vývojového prostředí 19
3.3. Rozbor aplikace 19
3.3.1. Modul main 19
3.3.2. Modul stdafx 20
3.3.3. Modul doublebuffering 21
3.3.4. Modul opengl_init 21
3.3.5. Modul themes 22
3.3.6. Modul auxiliary 22
3.3.7. Modul souradnice 24
3.3.8. Modul baze 25
3.3.9. Modul transformace 26
3.3.10. Modul promitani 28
3.3.11. Třída FPS 29
3.4. Tvorba nápovědy 29
3.5. Tvorba instalátoru 29
3.6. Testování aplikace 30

4. Uživatelská příručka **31**
4.1. Požadavky na HW 31
4.2. Instalace aplikace 31
4.3. Ovládání aplikace 33

4.3.1. Obecně k ovládání témat 35
4.3.2. Ovládání tématu: Afinní báze a souřadnice 36
4.3.3. Ovládání tématu: Matice přechodu mezi afinními bázemi . 40
4.3.4. Ovládání tématu: Matice afinních transformací 44
4.3.5. Ovládání tématu: Projektivní prostory a perspektivní promítání 51

Závěr **54**

Conclusions **55**

Reference **56**

E. Obsah přiloženého CD **57**

Seznam obrázků

1. Translace bodu 12
2. Rotace bodu 13
3. Změna měřítka 13
4. Rozsah perspektivního promítání určené jehlanem (frustum) . . . 16
5. Vykreslování scény 17
6. Volba jazyka instalace 31
7. Volba cílového umístění aplikace 32
8. Dokončení instalace aplikace 32
9. Spuštěná aplikace s úvodní obrazovkou 33
10. Informování uživatele o úspěšném načtení souboru 33
11. Neúspěšné načtení souboru 34
12. Neúspěšné vyvolání nápovědy 34
13. Tool bar 35
14. Demonstrace tématu „Afinní báze a souřadnice“ 37
15. Upozornění na chybně zadanou hodnotu 38
16. Upozornění na chybně zadanou vektorovou bázi 39
17. Řešení příkladu 40
18. Demonstrace tématu „Matice přechodu mezi afinními bázemi“ . . 41
19. Značka okénka zaměření V 41
20. Zobrazené okénko zaměření 42
21. Upozornění na chybně zadanou hodnotu 43
22. Upozornění na chybně zadanou vektorovou bázi 43
23. Demonstrace tématu „Matice afinních transformací“ 45
24. Volba geometrického útvaru 46
25. Tlačítko pro vyvolání dialogu nastavení transformační matice . . . 47
26. Dialog nastavení transformační matice 47
27. Volba úhlu rotace 48
28. Upozornění na chybně zadanou velikost úhlu 48
29. Zobrazení aktuálních souřadnic pozice kurzoru myši 49
30. Demonstrace tématu „Projektivní prostory a perspektivní promítání“ 51
31. Upozornění na chybně zadanou hodnotu 53
32. Upozornění na chybně zadanou hodnotu 53

1. Motivace

K pochopení matematických základů dvojrozměrné a trojrozměrné počítačové grafiky je potřeba seznámit se s některými oblastmi analytické geometrie. Tomuto účelu je věnován předmět *Geometrie 2*, určený zejména studentům informatických oborů Přírodovědecké fakulty na Univerzitě Palackého v Olomouci, který sám autor absolvoval. Na základě této skutečnosti lze zcela zaujatě využít všech zkušeností načerpaných během tohoto kurzu.

Studium geometrie vyžaduje jistou představivost, zejména prostorovou. Umět představit si teoretické pojmy (definice, věty, tvrzení), může snáze vést k pochopení probíraného problému. Zkoušející totiž pár cílenými dotazy snadno pozná, že student dané látce nerozumí. Zřejmě ne každý však potřebnou představivostí disponuje. Dá se tomu však dopomoci vhodnou doprovodnou ilustrací. Často k tomu poslouží papír nebo tabule a k demonstraci lehčích pojmů zcela jistě stačí. Avšak v případě těch težších, kdy se jedná např. o trojrozměrný prostor, se situace poněkud komplikuje. Je to dáno i tím, že „redukce“ takového prostoru na dvojrozměrný znamená určitou ztrátu informace. To motivuje k tomu, abychom našli lepší pomocný prostředek.
Nabízí se vytvořit počítačovou aplikaci, která dokáže:

- zachytit i složitější pojmy
- modelovat dvoj i trojrozměrný prostor
- interaktivně spolupracovat s uživatelem

Výhody spočívají zejména v možnosti pohybovat se ve vymodelovaném prostoru, různě natáčet úhel pohledu, zadávat hodnoty parametrů konkrétní úlohy atd. To vše bez nutnosti kreslit si pro každý případ nové obrázky. Student pak okamžitě vidí, co se stane po změně nějaké hodnoty a je tak schopen zkoumat například i hraniční případy použití různých definic. Právě tyto možnosti by měly pomoci při objasňovaní pojmů a na první pohled ne zcela jasných definic. Dokonce může nastat situace, kdy student zjistí, že si jistý pojem až dosud vykládal mylně. Sám autor se s tímto problémem potýkal a na řadu záludností přišel až při vývoji aplikace. To je ukazatel toho, že taková aplikace by opravdu mohla být přínosem při studiu geometrie.

2. Teoretický úvod

V této kapitole uvedeme důležité pojmy afinní geometrie, protože jejich problematikou se prakticky celá aplikace zabývá. Bez těchto základních znalostí se zřejmě neobejdeme, jelikož by pak aplikace jen těžko mohla plnit své poslání, totiž snahu uvedené pojmy objasnit, přiblížit, či pochopit. Budeme přitom co nejvíce vycházet z oficiálního učebního textu [1] k předmětu *Geometrie 2*. Jedná se o podkapitoly 2.1. – 2.3., 2.6. a definice 1 – 3, 9 – 11. Tento výklad doplníme některými pojmy z [2], konkrétně půjde o podkapitoly 2.4., 2.5.1. a definice 4 – 8. Dále pojmy z [3], konkrétně z podkapitoly 2.5.. A konečně pojmy z [4], konkrétně z podkapitoly 2.7..

Na závěr kapitoly ještě přidáme základní informace o grafické knihovně OpenGL, která byla zvolena pro výstup grafického obsahu.

2.1. Afinní prostory

Předpokládáme, že známe pojem množiny všech reálných čísel a pojem n-rozměrného vektorového prostoru. Na tomto základě se nejprve definuje tzv. n-rozměrný afinní prostor A.

Definice 1. Řekneme, že na neprázdné množině A je dána *struktura afinního prostoru*, je-li dán vektorový prostor V a zobrazení $Add_A : A \times V \to A$ takové, že

1. pro všechna $a \in A$, $u, v \in V$

$$Add_A(Add_A(a, u), v) = Add_A(a, u + v),$$

2. pro všechna $a, b \in A$ existuje právě jedno $v \in V$ tak, že

$$Add_A(a, v) = b.$$

Množina A se strukturou afinního prostoru se nazývá *afinní prostor*, její prvky se nazývají *body*. Vektorový prostor V se nazývá *zaměření* afinního prostoru A.

Je-li V vektorový prostor konečné dimenze, říká se o afinním prostoru A, že je také konečné dimenze. V takovém případě *dimenzí afinního prostoru* A nazýváme dimenzi vektorového prostoru V. Afinní prostor dimenze 1 se nazývá *afinní přímka*, afinní prostor dimenze 2 *afinní rovina*.

Zobrazení Add_A se nazývá *sčítání bodů a vektorů*. Pro libovolné $a \in A$ a $v \in V$ se bod $Add_A(a, v)$ označuje $a + v$ a nazývá se *součet bodu a a vektoru v*.

Vektor v se označuje $b - a$. Existuje pro libovolné a a b, a to právě jeden. Přiřazení $(a, b) \mapsto b - a$ je tedy jednoznačně určené zobrazení z $A \times A$ do V. Vektoru $b - a$ se říká *rozdíl bodů b a a*.

Na vektorovém prostoru V zavádíme *kanonickou strukturu afinního prostoru* takto: zaměřením prostoru V volíme samotný vektorový prostor V, pro libovolné dva vektory $u, v \in V$ klademe $Add_V(u, v) = u + v$, kde součet na pravé straně je součet vektorů ve vektorovém prostoru V.

2.2. Afinní báze a souřadnice

Definice 2. Mějme afinní prostor A dimenze n se zaměřením V. Je-li $a_0 \in A$ bod a e báze vektorového prostoru V, nazýváme dvojici $\varphi = (a_0, e)$ *afinní bazí afinního prostoru* A. Bod a_0 nazýváme *počátkem* báze φ.

Definice 3. Pro libovolný bod $a \in A$ nazýváme *afinními souřadnicemi bodu* a *vzhledem k afinní bázi* $\varphi = (a_0, e)$ $(n+1)$-tici:

$$a_\varphi = \begin{bmatrix} (a - a_0)_e^1 \\ (a - a_0)_e^2 \\ \vdots \\ (a - a_0)_e^n \\ 1 \end{bmatrix}.$$

Souřadnicové vyjádření bodu a vzhledem k afinní bázi φ tedy vznikne tak, že se najdou souřadnice vektoru $a - a_0$ vzhledem k bázi e vektorového prostoru V a doplní se číslem 1.

Je-li $a_0 = [0, \ldots, 0] \in \mathbf{R}^n$ a e kanonická báze vektorového prostoru $\mathbf{R}^n$, nazývá se afinní báze (a_0, e) afinního prostoru $\mathbf{R}^n$ *kanonickou afinní bazí* prostoru $\mathbf{R}^n$.

2.3. Matice přechodu mezi afinními bazemi

Mějme dvě afinní báze $\varphi = (a_0, e)$, $\bar{\varphi} = (\bar{a_0}, \bar{e})$ afinního prostoru A dimenze n se zaměřením V. Známe vzájemné vztahy obou afinních bazí, tj. souřadnice počátku a_0 vzhledem k afinní bázi $\bar{\varphi}$ a souřadnice vektorů vektorové báze e vzhledem k vektorové bázi $\bar{e}$ a naopak. Nyní chceme najít afinní souřadnice bodu $a \in A$ vzhledem k afinní bázi $\bar{\varphi}$, známe-li afinní souřadnice tohoto bodu vzhledem k afinní bázi φ. To se udělá tak, že sestavíme (po blocích) tzv. *matici přechodu od afinní báze* φ *k afinní bázi* $\bar{\varphi}$:

$$M_{\varphi,\bar{\varphi}} = \left(\begin{array}{ccc|c} & M_{e,\bar{e}} & & (a_0)_{\bar{\varphi}} \\ \hline 0 & \cdots & 0 & \end{array} \right),$$

kde $M_{e,\bar{e}}$ je matice přechodu od vektorové báze e k vektorové bázi $\bar{e}$ a $(a_0)_{\bar{\varphi}}$ jsou afinní souřadnice počátku a_0 vzhledem k afinní bázi $\bar{\varphi}$. Tyto hodnoty přitom

nezávisí na volbě bodu a (stačí je vypočítat pouze jednou). Takto sestavená matice pak bude splňovat vztah:

$$a_{\bar{\varphi}} = M_{\varphi,\bar{\varphi}} \cdot a_{\varphi},$$

pro námi hledané afinní souřadnice bodu $a \in A$ vzhledem k afinní bázi $\bar{\varphi}$.

2.4. Euklidovské prostory

V afinním prostoru A „neumíme“ měřit vzdálenosti a velikosti úhlů. Jestliže doplníme vlastnosti afinního prostoru A o další vlastnosti (zavedení skalárního součinu dvou vektorů), nazveme jej n-rozměrným euklidovským prostorem. Připomeňme, že skalární součin na vektorovém prostoru V je zobrazení $g : V \times V \to \mathbf{R}$, splňující pro libovolnou trojici vektorů $u, v, w \in V$ a libovolný skalár $a \in \mathbf{R}$:

1. $g(u + v, w) = g(u, w) + g(v, w)$ a $g(au, w) = ag(u, w)$,
2. $g(u, v) = g(v, u)$,
3. Jestliže $u \neq 0$, pak $g(u, u)$ 0.

Mějme afinní bázi $\varphi = (a_0, e)$ n-rozměrného euklidovského prostoru A se zaměřením V. Uvažujme všechny možné vzájemné skalární součiny vektorů $e_1, \ldots, e_n$ vektorové báze V. Získáme tak n^2 čísel $g_{ij} = e_i \cdot e_j$ pro $1 \leq i, j \leq n$. Tyto čísla tvoří symetrickou čtvercovou matici (g_{ij}), která se nazývá *Gramova matice* uspořádané n-tice vektorů $(e_1, \ldots, e_n)$. Známe-li v dané afinní soustavě souřadnic příslušnou Gramovu matici, pak se skalární součin dvou vektorů u, v počítá následovně:

$$u \cdot v = \left(\sum_{i=1}^{n} u_i e_i \right) \cdot \left(\sum_{j=1}^{n} v_j e_j \right) = \sum_{i,j=1}^{n} u_i v_j g_{ij}.$$

Definice 4. Afinní prostor A nazýváme *euklidovským prostorem*, jestliže jeho zaměřením je vektorový prostor V se skalárním součinem.

Definice 5. Nechť V je vektorový prostor se skalárním součinem. *Velikost* $\|u\|$ *vektoru* $u \in V$ definujeme rovností:

$$\|u\| = \sqrt{(u \cdot u)}.$$

Definice 6. *Odchylku* φ dvou nenulových vektorů u, v definujeme rovností:

$$\cos \varphi = \frac{u \cdot v}{\|u\| \|v\|},$$

a podmínkou $\varphi \in \langle 0, \pi \rangle$.

Definice 7. Nechť $a, b \in A$. Potom *vzdáleností bodů* a, b rozumíme číslo, které budeme označovat $d(a, b)$ a které definujeme rovností:

$$d(a, b) = \|b - a\|.$$

Jako příklad euklidovského prostoru si uveďme aritmetický afinní n-rozměrný prostor $\mathbf{R}^n$. Jeho zaměřením je n-rozměrný aritmetický vektorový prostor $\mathbf{R}^n$.

2.5. Matice afinních transformací

V aplikacích počítačové grafiky je často třeba modifikovat tvar a polohu rovinného objektu, tj. posunout ho o daný vektor, otočit o určitý úhel kolem pevného bodu, zvětšit nebo zmenšit jeho velikost, atd. Uvedené modifikace lze provádět pomocí afinních transformací (translace, rotace, změna měřítka, zkosení, ...). Jelikož se budeme zabývat pouze rovinnými transformacemi, budou pojmy v následujících podkapitolách konkretizovány přímo pro dvojrozměrný prostor.

2.5.1. Afinita

Než pokročíme k výkladu konkrétních transformací, bude potřeba objasnit ještě jeden důležitý pojem.

Definice 8. V euklidovské rovině A si zvolme afinní bázi $\varphi = (a_0, e)$. Uvažujme zobrazení $F : A \to A$, které každému bodu $x = [x_1, x_2, 1]$ přiřadí bod $x' = [x'_1, x'_2, 1]$ pomocí rovnic:

$$x'_1 = a_{11}x_1 + a_{12}x_2 + b_1,$$

$$x'_2 = a_{21}x_1 + a_{22}x_2 + b_2,$$

v nichž a_{ij}, b_i jsou libovolná reálná čísla taková, že $\det(a_{ij}) \neq 0$. Zobrazení F uvedených vlastností nazýváme *afinitou* v euklidovské rovině A.

Afinita je vzájemně jednoznačným zobrazením roviny A na A, kde přímce odpovídá přímka. Dále afinita zachovává rovnoběžnost přímek, dělící poměr tří bodů a obecně nezachovává délky úseček a velikosti úhlů.

Translace, rotace, změna měřítka, zkosení, stejnolehlost, osová afinita atd. jsou zvláštní případy afinity popsané rovnicemi z definice 8.

V následující části textu je použito označení souřadnicových os písmeny x a y. Jde o přímky určené bodem a_0 a směrovými vektory e_1, e_2:

$$x = a_0 + \langle e_1 \rangle,$$

$$y = a_0 + \langle e_2 \rangle,$$

kde a_0 je počátek soustavy a $\langle e_i \rangle$ značí lineární obal vektoru e_i.

2.5.2. Translace

Mějme v rovině $\mathbf{R}^2$ bod a a vektor posunutí t. Posuneme-li bod a o vektor t do bodu a', pak pro bod a' bude platit transformační rovnice:

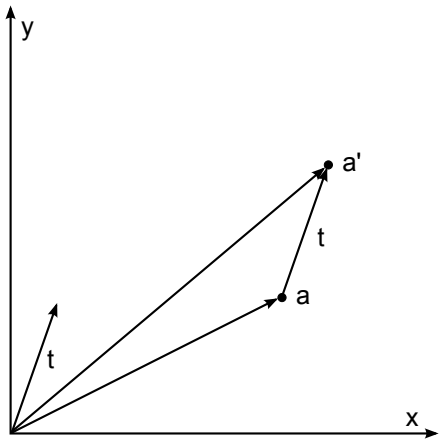
$$a' = a + t,$$

což rozepsáno po složkách dává:

$$a'_x = a_x + t_x,$$
$$a'_y = a_y + t_y.$$

Transformační rovnici můžeme psát maticově:

$$\begin{bmatrix} a'_x \\ a'_y \\ 1 \end{bmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{bmatrix} a_x \\ a_y \\ 1 \end{bmatrix}.$$



Obrázek 1. Translace bodu

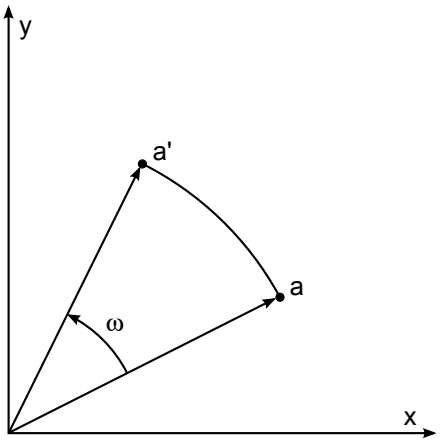
2.5.3. Rotace

Mějme v rovině $\mathbf{R}^2$ bod a, střed rotace nechť je v počátku soustavy souřadnic a úhel rotace je určen úhlem ω. Otočme bod a kolem počátku o úhel ω do bodu a'. Pro souřadnice otočeného bodu a' platí:

$$a'_x = a_x \cdot \cos\omega - a_y \cdot \sin\omega,$$
$$a'_y = a_x \cdot \sin\omega + a_y \cdot \cos\omega.$$

Rovnice můžeme přepsat maticově:

$$\begin{bmatrix} a'_x \\ a'_y \\ 1 \end{bmatrix} = \begin{pmatrix} \cos\omega & -\sin\omega & 0 \\ \sin\omega & \cos\omega & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{bmatrix} a_x \\ a_y \\ 1 \end{bmatrix}.$$



Obrázek 2. Rotace bodu

2.5.4. Změna měřítka

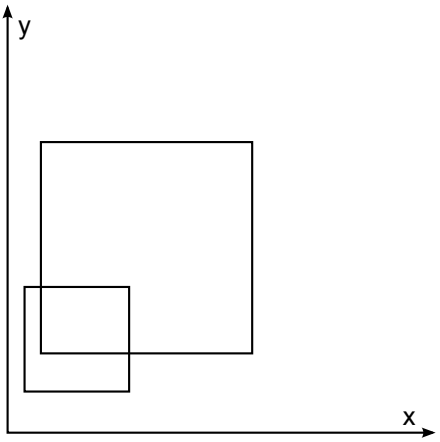
Tato transformace umožňuje měnit velikost objektu (zvětšit nebo zmenšit x-ovou nebo y-ovou souřadnici každého jeho bodu). Měřítková transformace je určena rovnicemi:

$$a'_x = s_x \cdot a_x,$$
$$a'_y = s_y \cdot a_y.$$

Maticový zápis tedy bude:

$$\begin{bmatrix} a'_x \\ a'_y \\ 1 \end{bmatrix} = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{bmatrix} a_x \\ a_y \\ 1 \end{bmatrix}.$$

Na obrázku 3. je příklad použití změny měřítka s hodnotami $s_x = s_y = 2$.



Obrázek 3. Změna měřítka

2.5.5. Speciální transformace

Ke speciálním transformacím patří například: osová souměrnost, středová souměrnost nebo zkosení. Uveďme jen odpovídající transformační matice.

Osová souměrnost. Za osu souměrnosti volíme osu x nebo osu y.

$$Os_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \; Os_y = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Středová souměrnost. Pro středovou souměrnost se středem v počátku souřadnic máme matici:

$$S = \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Zkosení. Běžně se používá zkosení ve směru osy x nebo ve směru osy y.

$$SH_x = \begin{pmatrix} 1 & 0 & 0 \\ s_x & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \; SH_y = \begin{pmatrix} 1 & s_y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

2.5.6. Skládání transformací

Transformace lze skládat do jediné matice postupným násobením elementárními transformacemi, což ve svých důsledcích vede na zrychlení vykreslování. Protože násobení matic není komutativní, záleží na pořadí ve kterém se transformace provádějí.

2.6. Projektivní prostory

Definice 9. Mějme vektorový prostor V dimenze $n+1$, kde $n \geq -1$. Množinu $P(V)$ všech jednorozměrných vektorových podprostorů prostoru V nazveme *projektivním prostorem nad vektorovým prostorem* V. Prvky projektivního prostoru $P(V)$ nazýváme *geometrickými body*. Je-li $a \in P(V)$ geometrický bod, pak libovolný vektor $v \in a$ nazýváme jeho *aritmetickým zástupcem*.

Pro aritmetického zástupce v geometrického bodu a platí $a = \langle v \rangle$.

Číslo n nazýváme *dimenzí* projektivního prostoru $P(V)$ a označujeme $\dim P(V)$. Je-li $n = 0$, nazýváme projektivní prostor $P(V)$ *projektivním bodem*, pro $n = 1$ hovoříme o *projektivní přímce*, pro $n = 2$ o *projektivní rovině*.

Je-li W vektorový podprostor vektorového prostoru V, je $P(W) \subseteq P(V)$. Projektivní prostor $P(W)$ nazýváme *projektivním podprostorem projektivního prostoru* $P(V)$. Je-li $\dim P(W) = \dim P(V) - 1$, nazýváme projektivní podprostor $P(W)$ *nadrovinou* v projektivním prostoru $P(V)$.

Definice 10. Mějme nenulové lineární zobrazení $F : V \to W$ a množinu $U = \{\langle v\rangle \in P(V) | F(v) \neq 0\}$. Zobrazení $\langle F\rangle : U \to P(W)$ definované předpisem $\langle F\rangle(\langle v\rangle) = \langle F(v)\rangle$, nazýváme *projektivním zobrazením vytvořeným lineárním zobrazením* F.

Definice 11. Libovolnou bázi $v_0, v_1, \ldots, v_n$ vektorového prostoru V nazveme *aritmetickou bazí projektivního prostoru* $P(V)$. Souřadnice nenulového vektoru $v \in V$ v této bázi nazveme *homogenními souřadnicemi bodu* $\langle v\rangle \in P(V)$ *vzhledem k aritmetické bázi* $v_0, v_1, \ldots, v_n$.

2.7. Perspektivní promítání

V počítačové grafice je potřeba kromě afinních zobrazení používat ještě zobrazení jiného typu. Platí to především o *perspektivním promítání*.

Potřebujeme promítat 3D předměty na 2D plochu. Průmět 3D objektu je definovaný jako průsečíky promítacích paprsků vycházejících ze středu promítání do každého bodu objektu s promítací rovinou. Perspektivní promítání je tedy definováno středem promítání, který se nachází v konečné vzdálenosti od promítací roviny. Velikost objektů se mění se vzdáleností od středu promítání, zachovává úhly jen pro plochy rovnoběžné s promítací rovinou, rovnoběžné čáry se nezobrazí jako rovnoběžné.

Body v prostoru reprezentujeme v homogenních souřadnicích, tzn. každé promítání definujeme maticí 4×4.

Střed promítání zvolme v počátku souřadnicového systému, promítací rovina je na ose z ve vzdálenosti d od středu promítání. Promítáme bod $P = [x, y, z]$ a hledáme souřadnice promítnutého bodu $P' = [x', y', z']$. Všechny hodnoty z jsou dovoleny, kromě $z = 0$. Tuto projekci lze zapsat v maticovém tvaru jako:

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{d} & 0 \end{pmatrix}.$$

Pokud ji vynásobíme homogenním bodem $P = [x, y, z, 1]^T$ dostaneme:

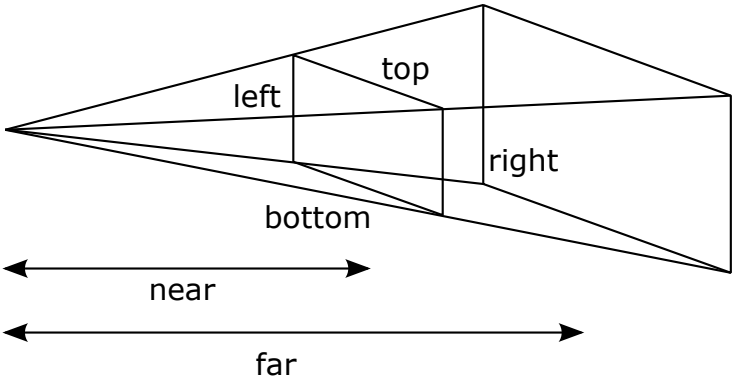
$$\left[x, y, z, \frac{z}{d}\right]^T.$$

Vydělíme poslední souřadnicí a máme:

$$[x', y', z', 1] = \left[\frac{x}{z/d}, \frac{y}{z/d}, d, 1\right].$$

Dělení z má za následek, že vzdálené objekty se jeví menšími. Touto projekcí přijdeme o informaci o hloubce (je to dáno lineární závislostí třetího a čtvrtého sloupce matice).

Naproti tomu projekční matice, kterou používá OpenGL, je poněkud složitější [8] a slouží k definování „jehlanu“ (angl. frustum), který vymezuje oblast, ve které se jednotlivé objekty nebo jejich části zobrazí na obrazovku – provádí se tzv. *clipping* (obr. 4.).



Obrázek 4. Rozsah perspektivního promítání určené jehlanem (frustum)

OpenGL perspektivní projekční matice vypadá následovně:

$$\begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}.$$

2.8. Grafická knihovna OpenGL

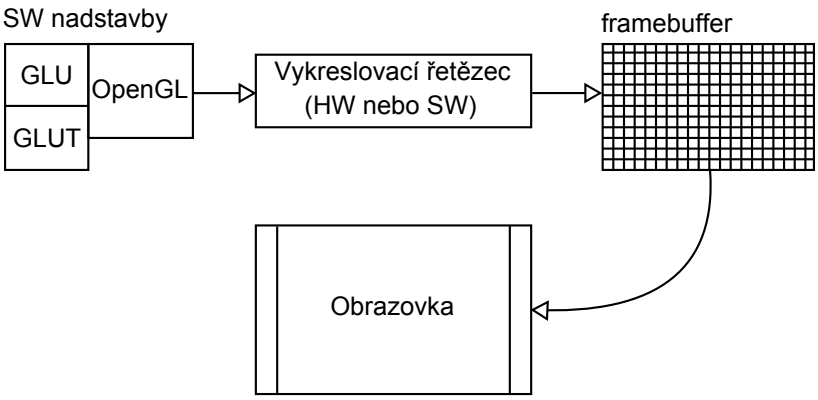
Knihovna OpenGL [5] (Open Graphic Library) je navržena jako aplikační programové rozhraní (Application Program Interface - API) k akcelerovaným grafickým kartám. Důraz je přitom kladen na to, aby byla použitelná na různých typech grafických akcelerátorů a to i v případě, že na určité platformě žádný grafický akcelerátor není nainstalován[1].

Z programátorského hlediska se OpenGL chová jako stavový automat. To znamená, že během zadávání příkazů pro vykreslování lze průběžně měnit vlastnosti vykreslovaných primitiv (barva, průhlednost) nebo celé scény (volba způsobu vykreslování, transformace) a toto nastavení zůstane zachováno do té doby, než ho explicitně změníme.

Vykreslování scény se provádí procedurálně – voláním funkcí OpenGL se vykreslí výsledný rastrový obrázek. Výsledkem volání těchto funkcí je rastrový obrázek uložený v tzv. framebufferu, kde je každému pixelu přiřazena barva, hloubka,

[1] v tom případě se použije softwarová simulace

alfa složka popř. i další atributy. Z framebufferu lze získat pouze barevnou informaci a tu je možné následně zobrazit na obrazovce – viz následující obrázek 5..



Obrázek 5. Vykreslování scény

Pomocí funkcí poskytovaných knihovnou OpenGL lze vykreslovat obrazce a tělesa složená ze základních geometrických prvků, které nazýváme grafická primitiva. Mezi tato primitiva patří bod, úsečka, trojúhelník, čtyřúhelník, plošný konvexní polygon, bitmapa (jednobarevný rastrový obraz) a pixmapa (barevný rastrový obraz). Existují i funkce, které podporují proudové vykreslování některých primitiv – lze například vykreslit polyčáru (line loop), pruh trojúhelníků (triangle strip), pruh čtyřúhelníků (quad strip) nebo trs trojúhelníků (triangle fan). Na vrcholy tvořící jednotlivá grafická primitiva lze aplikovat různé transformace (otočení, změna měřítka, posun, perspektivní projekce), pomocí kterých lze poměrně jednoduše vytvořit animace.

OpenGL nezaručuje, že při spuštění identického programu používajícího knihovnu OpenGL na různých platformách nebo různých grafických akcelerátorech dostaneme vždy přesně stejný výsledek. Pokud bychom oba výsledné rastrové obrázky porovnali pixel po pixelu, mohli bychom zjistit mírné rozdíly v barvách. Může to být způsobeno například odlišnou přesností reprezentace čísel na grafické kartě, odlišnými algoritmy pro interpolaci barvy, normály a texturových souřadnic nebo jinou bitovou hloubkou Z-bufferu. Celkové geometrické a barevné podání scény by však mělo být zachováno.

3. Programátorská dokumentace

3.1. Použité technologie

Aplikace *Geometrie* byla vyvinuta pro operační systém Microsoft Windows pomocí aplikačního programového rozhraní win32 API [7] (formálně Windows API), což je API vyvinuté firmou Microsoft pro operační systém Microsoft Windows. Všechny programy v Microsoft Windows musí nezávisle na použitém programovacím jazyce komunikovat prostřednictvím Windows API, které obsahuje nejen základní funkce, ale i funkce pro vytváření uživatelského rozhraní a další. Ve spojení s OpenGL jsou to velice silné nástroje.

Funkčnost Windows API lze rozdělit do několika kategorií, nejdůležitější z nich je však pro naši aplikaci kategorie **uživatelské rozhraní**, které poskytuje funkce pro tvorbu a řízení oken a dalších základních prvků jako jsou tlačítka a posuvníky. Dále zpracovává vstup z klávesnice a myši a jiných funkcí spojených s GUI. Tato funkční jednotka se nachází v `user32.dll`. Do této kategorie patří:

knihovna běžných prvků (Common Control Library) `comctl32.dll`,
která poskytuje aplikaci přístup k pokročilejším prvkům operačního systému. Zahrnuje věci jako stavový řádek, zobrazení průběhu výpočtu, toolbary a záložky.

knihovna běžných dialogových oken `comdlg32.dll`,
která poskytuje aplikaci standardní dialogová okna pro otevření a ukládání souborů, volby barvy a fontů, apod.

Pro výstup grafického obsahu byla použita knihovna OpenGL [5]. Z důvodu její nezávislosti na použitém operačním systému, grafických ovladačích a správcích oken, neobsahuje žádné funkce pro práci s okny, pro vytváření grafického uživatelského rozhraní (GUI), ani pro zpracování událostí. Tyto funkce jsou zajištěny voláním funkcí Windows API.

Pro dosažení co největší nezávislosti na použité platformě zavádí knihovna OpenGL vlastní primitivní datové typy, například `GLbyte`, `GLint`, `GLfloat` nebo `GLdouble`.

Programátorské rozhraní knihovny OpenGL je vytvořeno tak, aby knihovna byla použitelná v téměř libovolném programovacím jazyce. Primárně je k dispozici hlavičkový soubor pro jazyky C a C++. V tomto souboru jsou deklarovány nové datové typy používané knihovnou, některé symbolické konstanty (např. `GL_POINTS`) a sada cca 120 funkcí tvořících vlastní rozhraní.

Win32 API je rozhraní pro programování zejména v jazyce C a C++. Spolu se skutečností uvedenou výše byl pro programování aplikace zvolen právě jazyk C++. Aplikace byla vyvíjena ve vývojovém prostředí Microsoft Visual Studio 2008 (použitím Visual C++), které je zaměřeno na vývoj pro Windows (Vista).

3.2. Nastavení vývojového prostředí

Pro kompilaci programu bylo potřeba mít správně nainstalovaný GLUT (OpenGL Utility Toolkit) pro win32, který obsahuje knihovny `glut32.lib`, `glut32.dll` a hlavičkové soubory `gl.h`, `glu.h`, `glut.h`. Knihovna `glut32.lib` musí být zkopírována do složky `C:/WINDOWS/system32/`, knihovna `glut32.dll` a hlavičkové soubory do složek `../Lib/` a `../Include/GL/`, které jsou ve složce, kde je nainstalované vývojové prostředí. Tyto knihovny a hlavičkové soubory jsou přiloženy na CD (viz příloha E.).

Dalším krokem bylo přidat potřebné knihovny do linkeru. To se provádí v nastavení projektu. Jde o tyto knihovny: `opengl32.lib`, `glu32.lib`, `glut32.lib`, `comctl32.lib` a `shlwapi.lib`. Ve vývojovém prostředí Microsoft Visual Studio lze rovněž pro vkládání knihoven použít direktivu `#pragma comment()`.

Kromě knihovny `glut32.lib` jsou ostatní staticky linkovatelné, tj. obsahující kompletní kód a jsou součástí operačního systému Windows (v případě knihovny `opengl32.lib` součástí ovladačů grafické karty). Knihovna `glut32.lib` je exportní a po jejím přilinkování bude program vyžadovat spolu se spustitelným souborem ještě dll verzi této knihovny `glut32.dll`.

3.3. Rozbor aplikace

S vyjímkou implementace jediné (jednoduché) třídy **FPS** bylo jako styl programovaní zvoleno procedurální paradigma. Zdrojový kód je rozdělen do několika modulů s příslušnými hlavičkovými soubory. Jedná se o tyto moduly:
main, **stdafx**, **doublebuffering**, **opengl_init**, **themes**, **auxiliary**, **souradnice**, **baze**, **transformace** a **promitani**.

Nyní následuje popis každého modulu a u nejdůležitějších i popis funkcí v nich definovaných. V hlavičkových souborech jsou uvedeny prototypy těchto funkcí. Použití funkčních prototypů je v jazyce C++ povinné.

3.3.1. Modul main

Modul tvoří základní stavební prvek celé aplikace. Obsahuje definice funkcí, které určují strukturu celého programu. Jedná se o hlavní funkci a funkci obslužující hlavní smyčku zpráv okna. Jejich funkční prototypy jsou:

```
int APIENTRY _tWinMain(HINSTANCE, HINSTANCE, LPTSTR, int);
LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
```

Pro zachování co největší přehlednosti kódu jsou všechny funkce zajišťující obsluhy jednotlivých zpráv definovány v jiných modulech.

Ve funkci `_tWinMain` se nejprve vytváří okno aplikace. Tuto inicializaci má na starosti funkce definovaná v modulu **opengl_init** a bude o ní řeč v kapitole 3.3.4.. Dále následuje smyčka zpráv. Ta probíhá dokud není aplikace ukončena a každým průchodem cyklu zjišťuje, zda došla nějaká zpráva. V případě že:

ano – je tato zpráva přeložena a poslána dál do funkce `WndProc`, která veškeré přijaté zprávy z této smyčky obsluhuje.
ne – dojde k vykreslení OpenGL scény a následné prohození bufferů (double buffering). Vykreslování OpenGL scény tedy rovněž probíhá stále dokola ve smyčce.

Ve funkci `WndProc`, jak již bylo zmíněno, probíhá obsluha zaslaných zpráv. Mezi nejdůležitější patří obsluhy zpráv:

WM_CREATE – je volána ihned po inicializaci okna (jednorázově) a má na starosti zobrazení stavového řádku, toolbaru a dvou toolboxů po stranách okna aplikace.

WM_COMMAND – obsluhuje menu aplikace a je dále dělena na konkrétní položky podle toho, na kterou nabídku bylo kliknuto.

WM_KEYDOWN, WM_KEYUP – reaguje na stisk a uvolnění klávesy. Stisknutá resp. uvolněná klávesa je určena naplněním konkrétního prvku pole `bool keys[256]` hodnotou `true` resp. `false` podle parametru obsluhované zprávy.

WM_MOUSEMOVE – zde je potřeba přepočítávat souřadnice polohy myši. V klientské části okna je totiž počátek souřadného systému v levém horním rohu a „roste“ směrem doprava a dolů. Naproti tomu souřadný systém OpenGL má počátek v levém dolním rohu a „roste“ směrem doprava a nahoru. Dále je tu volána funkce pro ovládání OpenGL scény pomocí myši.

WM_LBUTTONDOWN, WM_LBUTTONUP – reaguje na stisk a uvolnění levého tlačítka myši. V místě stisku jsou opět přepočítány souřadnice. Hodnota souřadnic místa stisku je po dobu stisku neměnná.

WM_PAINT – je volána pokaždé, kdy je potřeba překreslit celé okno. To nastává např. při pohybu okna a změně velikosti. Rovněž při vyvolání nějakého dialogu je potřeba překreslovat okno, zejména při jeho pohybu přes okno aplikace. V tomto okamžiku jsou zpracovávány obsluhy zpráv a volání funkce pro vykreslování OpenGL scény umístěné ve smyčce zpráv se tedy po tuto dobu nemůže dostat na řadu. Proto je nutné ji umístit i do obsluhy této zprávy. Tuto nutnost potvrzuje fakt, že při pohybu nějakého dialogového okna přes OPenGL scénu by po něm zůstávala nepříjemná stopa, jako následek jejího momentálního nepřekreslování.

WM_SIZE – zde se volají funkce pro změnu velikosti OpenGL scény, stavového řádku, toolbaru a toolboxů po stranách okna aplikace.

V hlavičkovém souboru *main.h* jsou uvedeny všechny hlavičkové soubory ostatních modulů.

3.3.2. Modul stdafx

Tento modul vygeneruje vývojové prostředí Microsofot Visual Studio vždy při založení nového projektu. Jeho jediné poslání je odkaz na jeho předkom-

pilovanou hlavičku *stdafx.h*, ve které jsou uvedeny všechny důležité hlavičkové soubory. Jsou to zejména *windows.h* a *glut.h*.

3.3.3. Modul doublebuffering

Modul obsahuje funkce, které se volají v obsluze zprávy WM_PAINT. Jde o trojici funkcí, jejichž funkční prototypy jsou:

```
void Pametovy_buffer(HWND);
void Hlavni_buffer(PAINTSTRUCT);
void Likvidace_bufferu();
```

První funkce má za úkol vytvořit paměťové kontextové zařízení a přiřadit mu paměťovou bitmapu. Druhá funkce pak zkopírovat z paměťové bitmapy příslušnou část odpovídající oblasti určené k překreslení. Mezi tyto dvě funkce je pak v obsluze volána funkce pro překreslení OpenGL scény.

Poslední funkce se volá při zavírání aplikace a ruší vytvořený paměťový kontext a paměťovou bitmapu. Řeč o kontextu zařízení bude v kapitole 3.3.4..

3.3.4. Modul opengl_init

Zde jsou definovány funkce pro vytvoření okna, inicializaci OpenGL, změnu velikosti okna a jeho zrušení. O vytvoření okna aplikace se stará funkce s prototypem:

```
bool CreateGLWindow(int, int, int);
```

Jejími parametry jsou po řadě šířka a výška okna a barevná hloubka. Vytvářené okno je mimo jiné nutné nastavit tak, aby se v něm mohla zobrazovat OpenGL scéna. Každý OpenGL program je spojen s Rendering Contextem. Rendering Context říká, která spojení volá OpenGL, aby se spojilo s Device Context (kontext zařízení). Aby program mohl kreslit do okna, potřebujeme vytvořit Device Context, který napojí okno na GDI (grafické rozhraní). Nastavují se tedy tyto důležité proměnné:

`HDC hDC` – Device Context
`HGLRC hRC` – Rendering Context
`HWND hWnd` – handle okna
`HINSTANCE hInstance` – instance aplikace

Následuje naplnění struktury `PIXELFORMATDESCRIPTOR`. Zejména volba double bufferingu, RGBA, barevné hloubky a Z-Bufferu. Po získání kontextu zařízení Windows najde vhodný Pixel Format a nastaví ho. Poté se získá Rendering Context, který se aktivuje. Nakonec už zbývá volat funkci pro inicializaci OpenGL. Její funkční prototyp je:

```
bool InitGL();
```

Zde se nastaví jemné stínování, černé pozadí, hloubkový buffer a typ hloubkového testování. Dále ještě nastaví nejlepší perspektivní korekci.

Veškeré potřebné návody a kroky ke zprovoznění OpenGL v API okně ve Windows jsou popsány v [9].

3.3.5. Modul themes

Tento modul slouží jako rozcestník pro jednotlivá výuková témata, kterými se aplikace zabývá. Každé téma má svou vykreslovací funkci a funkce pro ovládání scény pomocí myši a klávesnice. Ty jsou vždy definované v příslušných modulech. Prototypy funkcí definovaných v tomto modulu jsou:

```
void DrawGLScene(int);
void OvladaniKlavesnice(int);
void OvladaniMys(int);
```

Parametrem funkcí je vždy číslo tématu. Struktura všech tří funkcí je obdobná, obsahuje přepínač a podle parametru funkce je volána příslušná funkce týkající se zvoleného tématu. Ve funkci `DrawGLScene` je navíc nastavení viditelné oblasti OpenGL scény v hlavním okně aplikace a výpočet počtu volání této funkce za sekundu, tj. fps (frames per second). Zodpovědnost za ní má metoda třídy `FPS`. Tato hodnota je zobrazena vpravo ve stavovém řádku a zmínka o ní bude v kapitole 3.3.11..

Funkce `DrawGLScene` spolu s funkcí `OvladaniKlavesnice` je volána ve smyčce zpráv v hlavní funkci `_tWinMain` a funkce `OvladaniMys` v obsluze zprávy WM_MOUSEMOVE ve funkci `WndProc`, o kterých byla řeč v kapitole 3.3.1..

3.3.6. Modul auxiliary

Jde o nejrozsáhlejší modul z hlediska počtu definic funkcí. Funkce zde spolu nijak nesouvisí ve smyslu nějakého společného prostoru problému. Modul slouží spíše jako skladiště, kde jsou funkce definovány, aby se „nepletly“ v modulech, do kterých sice třeba patří, ale znepřehledňovaly by kód a zbytečně modul zvětšovaly. Kromě toho jsou zde definovány i funkce využitelné ve více modulech, proto dostal označení jako „pomocný“.

V hlavičkovém souboru *auxiliary.h* je deklarována třída `FPS` a důležitá struktura `SOURADNICE`, jejíž členy jsou souřadnice `x,y,z` datového typu `GLfloat`. Tato struktura provází prakticky celý program a značně ulehčuje práci s body a vektory.

Z velkého počtu funkcí vyjmenujeme ty důležitější a podrobněji rozebereme ty nejvíce důležité. Začneme těmi důležitými a místo prototypů uvedeme jejich zodpovědnosti:

Vytvoření stavového řádku, toolbaru, dvou toolboxů po stranách okna aplikace a změny velikosti právě jmenovaných, obsluha smyčky zpráv pro levý toolbox,

obsluha smyčky zpráv pro pravý toolbox. Implementace těchto funkcí je často dost specifická pro programování ve win32 API, proto nemá smysl se jimi podrobně zabývat. Veškerá dokumentace k API funkcím je na MSDN [10].

Změna levého toolboxu v závislosti na volbě tématu, nastavení antialiasingu a tisknutí textu do OpenGL scény. Jedná se o nějaký přepínač nebo konkrétní nastavení parametrů OpenGL.

Pokračujme těmi nejvíce důležitými funkcemi jmenováním jejich prototypů a následného popisu:

```
SOURADNICE ReseniSoustavy3(SOURADNICE, SOURADNICE, SOURADNICE,
SOURADNICE);
SOURADNICE ReseniSoustavy2(SOURADNICE, SOURADNICE, SOURADNICE);
```

Tato dvojice funkcí řeší soustavu lineárních rovnic $Ax = b$. Jde o verze 2 resp. 3 rovnic o 2 resp. 3 neznámých. Řešení je realizováno pomocí *Cramerovy věty*. Prvním parametrem je vektor pravé strany b a další parametry postupně sloupce matice A. Jako výsledek je vrácena neznámá x v podobě datového typu `SOURADNICE`.

```
GLfloat Determinant3(SOURADNICE, SOURADNICE, SOURADNICE);
GLfloat Determinant2(SOURADNICE, SOURADNICE);
```

Tato dvojice funkcí vrací determinant matice zadané postupně jejími sloupci. Primární účel těchto funkcí je ověřování lineární nezávislosti vektorů. Přirozeně jsou tedy jako parametry zadávány dvojice nebo trojice vektorů.

```
SOURADNICE VyjadriBod(GLfloat m[3][3], SOURADNICE);
```

Tato funkce vynásobí zadanou matici 3×3 zadaným bodem a vrátí nový bod, který je výsledkem tohoto násobení. Funkce je využitá při vyjadřování bodu v jiné bázi. Tzn. jde o násobení matice přechodu s bodem.

```
void NasobeniMatic(GLfloat m1[3][3], GLfloat m2[3][3], GLfloat
m[3][3]);
```

Jde o jednoduchou funkci, která provádí maticové násobení dvou matic v pořadí jak jsou zadány parametry. Do posledního parametru se uloží výsledek.

```
SOURADNICE PrepocetSouradnicMysi(int, int);
```

V kapitole 3.3.1. bylo zmíněno, že obsluha zprávy WM_MOUSEMOVE má na starosti mimo jiné přepočet souřadnic myši. Parametry funkce jsou postupně x-ová a y-ová souřadnice myši ve Windows, které se získají z parametrů obsluhované zprávy. Návratová hodnota funkce je typu `SOURADNICE`, tzn. v členských proměnných jsou uloženy potřebné výsledky přepočtu.

Jde o poměrně jednoduchou záležitost, výsledná x-ová složka vznikne odečtením šířky levého toolboxu od původní x-ové složky. Výsledná y-ová složka vznikne přičtením výšky klientské části okna k -1 násobku původní y-ové složky. K oběma

hodnotám je navíc přičten přírustek vzniklý případným pohybem OpenGL scény pomocí funkce `glTranslatef`, protože po jejím volání se počátek posunuje.

```
bool Between3(SOURADNICE, SOURADNICE, SOURADNICE, SOURADNICE);
bool Between4(SOURADNICE, SOURADNICE, SOURADNICE, SOURADNICE,
SOURADNICE);
```

Tyto funkce zjistí, zda se kurzor myši nachází nad trojúhelníkem resp. čtyřúhelníkem zadaným jeho body. Prvním parametrem jsou souřadnice kurzoru myši, další pak body daného objektu. V případě čtyřúhelníku se využije jeho rozdělení na dva trojúhelníky. Pro řešení je využit následující postup:
Základem jsou tři přímky tvořené stranami trojúhelníku a jejich rovnice v normálovém tvaru $ax + by + c = 0$. Normálové vektory (a, b) se zjistí ze směrových vektorů – ty vzniknou rozdílem dvou bodů, proto se odvíjejí od pořadí v jakém jsou tyto body předány funkci. Z rovnic se vyjádří parametry c. Nyní se už dá zjistit, kam směřují normálové vektory – dosazením protějších bodů do rovnic. Pokud bude výsledek rovnice ≥ 0, pak tento vektor směřuje dovnitř trojúhelníku, naopak bude-li výsledek rovnice ≤ 0, pak směřuje ven z trojúhelníku. Tedy mohou nastat dvě možnosti:
směřuje dovnitř – pak po dosazení souřadnic kurzoru myši do rovnice musí platit, že výsledek je ≥ 0,
směřuje ven – pak po dosazení souřadnic kurozoru myši do rovnice musí platit, že výsledek je ≤ 0.
Pokud je kurzor myši nad trojúhleníkem, musí platit podmínky všech tří rovnic zároveň.

3.3.7. Modul souradnice

Modul obsahuje definice funkcí zodpovědných za vykreslení OpenGL scény týkající se tématu *afinní báze a souřadnice*, jejího ovládání pomocí myši i klávesnice, výpisu textu (zejména hodnot souřadnic bodů a vektorů a stručnému označení demonstrované problematiky), obsluhy smyčky zpráv pro levý toolbox, ukládání a načítání do souboru. Jmenujme funkční prototypy a popis jednotlivých funkcí:

```
void AfinniSouradnice();
```

Jedná se o scénu v prostoru, proto je na začátku funkce zvolena matice *modelview* a následně vynulována. Pohled na scénu je realizován pomocí funkce `gluLookAt` a je počátečně natočen podle osy x a y o 45 stupňů pomocí funkcí `glRotatef`. Počátek je v prostředku okna a po celou dobu zůstává na jednom místě. Po vykreslení afinní soustavy souřadnic následuje nejdůležitější část funkce, a to vyjádření bodu a v této soustavě. K tomu se využije funkce `ReseniSoustavy3` z modulu **auxiliary**, která byla popsána v kapitole 3.3.6.. Pro složky souřadnic bodu $a' = (x, y, z)$ je totiž potřeba vyřešit rovnici $a - a_0 = xe_1 + ye_2 + ze_3$, kde a_0 je počátek této soustavy a e je vektorová báze.

Tím ovšem všechny výpočty nekončí, protože například bod zadaný v afinní bázi je nutno nějak transformovat do systému OpenGL, aby se vykreslil tam, kde má. Postup je obdobný výše uvedenému, jde jen o vyjádření jiných proměnných z uvedených rovnic. Využití je zde konkrétně v situaci, kdy se vykreslují pomocné čáry směřující z bodu a' k souřadným osám (je známé místo doteku na ose vyjádřené v afinní bázi).

```
static void VypisTextu(SOURADNICE);
```

Funkce vypisuje jednoduchý text na zvolené místo ve scéně. To znamená, že při aktuálně naplněné *modelview* matici se text pohybuje spolu s případným pohybem scény. To se hodí zejména pro popisy os. Je však ale potřeba umístit i text napevno na jisté místo obrazovky. V tomto případě funkce postupuje následovně: zvolí se *projection* matice, uloží se její hodnoty pomocí funkce `glPushMatrix`, resetuje se, zvolí se ortogonální projekce (2D), zvolí se zpět *modelview* matice, uloží se její hodnoty a resetuje. Parametrem funkce je bod a', ten totiž není definovaný jako globální, ale vzniklý a vypočítaný za běhu programu.

Nyní už lze libovolně vypisovat text v rovině. Nakonec se musí opět zvolit *projection* matice, vrátí se zpět její uložené hodnoty pomocí funkce `glPopMatrix`, zvolí se zpět *modelview* matice a vrátí zpět její uložené hodnoty.

```
void AfinniSouradniceKlavesnice();
```

Ovládání je poměrně jednoduché, reaguje se na stisk několika kláves. Hlavně směrových šipek, které scénou otáčejí kolem osy x a y. Aby byla zajištěna konstantní rychlost otáčení (různě rychlé grafické karty), je velikost úhlu otočení za sekundu dělena hodnotou FPS.

```
void AfinniSouradniceMys();
```

Obsluha má pouze malou funkčnost, jelikož jde o scénu v prostoru, dá se využít pouze k otáčení scény kolem os zmáčknutím levého tlačíčtka a následným táhnutím.

Funkce pro ukládání a načítání souboru pracují tak, že jsou uloženy binárně celé struktury souřadnic. Následné načítání pak probíhá přímým přístupem od určitého bytu v souboru odpovídajícímu pozici začátku dat, které patří tomuto tématu.

3.3.8. Modul baze

Modul obsahuje definice funkcí zodpovědných za vykreslení OpenGL scény týkající se tématu *matice přechodu mezi afinními bazemi* a funkcí se stejnou zodpovědností jako v modulu **souradnice**. Všechny čtyři moduly mají podobnou strukturu, proto se budeme věnovat vždy jen výraznějším implementačním rozdílům. Začneme funkcí pro vykreslení s prototypem:

```
void MaticePrechodu();
```

Jedná se o scénu v rovině, proto je na začátku funkce zvolena matice *projection* a následně vynulována. Pohled na scénu je realizován pomocí funkce `gluOrtho2D`, jedná se tedy o ortogonální projekci, která má počátek v levém dolním rohu. Tento počátek je předposunut o $(20, 30)$ px pomocí funkce `glTranslatef`, aby při spuštění aplikace nesplýval s levým dolním rohem.

Následuje hlavní část funkce, která sestává ze série několika výpočtů. Jde především o přepočty mezi souřadnicemi bodů v jejich bázích. Ty jsou realizovány dle teorie, tedy násobením příslušné matice přechodu. Použity jsou přitom funkce definované v modulu **auxiliary**. Jedná se o použití funkcí `VyjadriBod` a `ReseniSoustavy2`. Tato různá vyjádření je opět potřeba transformovat do systému OpenGL tak, aby se korektně vykreslily tam, kde mají.

Součástí vykreslované scény je i demonstrace zaměření vektorového prostoru. To zajišťuje funkce s prototypem:

```
void Zamereni();
```

a je volána vždy při najetí kurzoru myši nad příslušnou značku „V“ umístěnou v pravém dolním rohu scény. Před samotným vykreslením potřebných údajů je třeba uložit hodnoty *projection* matice pomocí funkce `glPushMatrix` a její následné vynulování, aby bylo možné tento kus scény zobrazovat vždy v pravém dolním rohu. Následně jsou zpět vráceny původní hodnoty *projection* matice pomocí funkce `glPopMatrix`.

Implementace funkce pro ovládání scény pomocí klávesnice s prototypem:

```
void MaticePrechoduKlavesnice();
```

je podobná té v modulu **souradnice**. Rozdíl je ve funkčnosti směrových kláves. Ty pohybují scénou různými směry konstantní rychlostí 200 px za sekundu. Tento pohyb zabezpečuje volání funkce `glTranslatef`.

Trochu rozsáhlejší je implementace funkce pro ovládání scény pomocí myši. Její prototyp je:

```
void MaticePrechoduMys();
```

Pro správné vyhodnocení pozice jednotlivých bodů se kterými bude možno pohybovat je nutné transformovat jejich souřadnicová vyjádření v jejich bázích do systému OpenGL, provést nutné výpočty a převod zpět do jejich původních vyjádření, která jsou po výpočtech modifikována. Vůle pro zachycení bodu myší je stanovena na ± 5 px.

3.3.9. Modul transformace

Svou celkovou funkčností jde o nejrozsáhlejší modul. Kromě definic funkcí se stejnou zodpovědností jako v jiných modulech obsahuje dále definice funkcí

pro obsluhy smyček několika dialogových oken potřebných pro nejrůznější nastavení transformačních matic a dále definici fukce realizující tzv. „teselaci“. Začneme jako obvykle funkcí pro vykreslení OpenGL scény týkající se tématu *matice afinních transformací*. Její prototyp je:

```
void AfinniTransformace();
```

Začátek implementace je stejný jako v případě funkce `MaticePrechodu` z modulu **baze**. Dále následuje naplnění transformačních matic `GLfloat M1[3][3]` a `GLfloat M2[3][3]` podle typu zvolené transformace a poté výpočet transformovaných bodů. Ten probíhá následovně: v případě, že je zvolena jednoduchá transformace se matice `M1` vynásobí s bodem určeným k transformaci pomocí funkce `VyjadriBod`, kde parametry jsou postupně matice a bod. Výsledkem bude transformovaný bod. V případě, že je zvolena složená transformace, vynásobí se postupně matice `M1` a `M2` pomocí funkce `NasobeniMatic`. Výsledek bude uložen v matici `GLfloat M[3][3]` a až tato matice se násobí bodem určeným k transformaci.

Při vykreslování transformovaného objektu jako je trojúhelník a čtyřúhelník je zapnut blending, tj. průhlednost. V případě vzájemného překrytí původního a transformovaného objektu pak budou vždy vidět hrany objektů. Blending se musí aktivovat voláním funkce `glEnable` s parametrem `GL_BLEND` a nastavením míchací funkce `glBlendFunc` s parametry `GL_SRC_ALPHA` a `GL_ONE_MINUS_SRC_ALPHA`. Poté se už jen nastaví barva objektu pomocí funkce `glColor4f`, kde se jako čtvrtý parametr uvede hodnota 0–1, což je alfa složka.

Ovládání scény pomocí klávesnice a myši je stejné jako v modulu **baze**. Implementační rozdíly spočívají jen v tom, s čím vším je možno pohybovat. Funkční prototypy jsou:

```
void AfinniTransformaceKlavesnice();
void AfinniTransformaceMys();
```

Myší lze pohybovat i transformovanými objekty, a to navíc v závislosti na zvolené transformaci (translace a obecná = libovolný pohyb, rotace = pohyb po části kružnice resp. elipsy, změna měřítka = pohyb mění velikost objektu). Nejvýznamnější problém je řešení pohybu objektu při rotaci. Myšlenka je následující: při pohybu transformovaného objektu se bude měnit přímo úhel rotace `GLfloat alfa1` resp. `GLfloat alfa2`, pokud jde o jednoduchou resp. složenou transformaci. Tento úhel je dán odchylkou dvou vektorů určených původním neorotovaným bodem a pozicí kurzoru myši spočítanou podle vzorce z definice 6. V triviálním případě, kdy uživatel klikne přímo na bod určující transformovaný objekt, by se dal původní neorotovaný bod přímo zjistit, neboť tento bod je znám po celou dobu programu (navíc je definovaný jako globální). Pomocí těchto dvou bodů by se už dala odchylka spočítat. Ve většině případů však uživatel klikne někam dovnitř trasformovaného objektu a je nutné jeho netransformovaný „ekvivalent“ spočítat na základě znalosti aktuálního úhlu rotace. Celkové řešení problému tedy

bude obecné a zahrne i triviální případ. Popišme postup: provede se „zpětná“ transformace, tj. inverzní matice se vynásobí s bodem místa stisku levého tlačítka myši a dostaneme tak původní netransformovaný bod, který bude po dobu stisku konstantní. Máme tedy pevně daný bod, dále bod daný pozicí kurzoru myši a nyní už můžeme počítat odchylku, která však díky omezenému definičnímu oboru bude nabývat pouze hodnot v intervalu $\langle 0, \pi \rangle$.

Nyní ke vzpomínané teselaci [6]. Ani knihovna OpenGL ani samotný grafický akcelerátor nepodporují vykreslování obecných polygonů. Teselace je proces, pomocí kterého se obecný polygon převádí na nepravidelnou trojúhelníkovou síť. Nástroj pro převod obecných polygonů do formy vhodné pro vykreslování je dostupný až v nadstavbové knihovně GLU a nazývá se podle své funkce teselátor. Teselátory pomocí specializovaných funkcí z knihovny GLU získají specifikaci takzvaného jednoduchého polygonu a po provedení teselace vrátí seznam grafických primitiv (trojúhelníků či trsů a pruhů trojúhelníků), které je již možné pomocí funkcí OpenGL přímo vykreslit. Reakce na různé události vznikající při práci teselátorů se provádí pomocí tzv. callback funkcí. V aplikaci je problém teselace řešen ve funkci s prototypem:

```
void Teselace(GLdouble vrcholy[4][3]);
```

Parametrem funkce je pole vrcholů čtyřúhelníku. Kromě vlastního zadávání jednotlivých polygonů je nutné správně reagovat na data poslaná do callback funkcí. Celý postup je následující: nejprve se vytvoří objekt pro teselaci, poté se registrují callback funkce, jmenovitě jsou to funkce s těmito prototypy:

```
void CALLBACK callbackVertex(GLdouble coords[3]);
```
Tato funkce je volána při vytvoření každého vrcholu teselovaného polygonu.

```
void CALLBACK callbackBegin(GLenum);
```
Tato funkce je volána při zahájení vykreslování grafické primitivy.

```
void CALLBACK callbackEnd();
```
Tato funkce je volána při ukončení vykreslování grafické primitivy.

```
void CALLBACK callbackCombine(GLdouble coords[3], GLdouble
*data[4], GLfloat weight[4], GLdouble **dataOut);
```
Tato funkce je volána při výskytu průsečíku dvou hran polygonu.

Nakonec se specifikují jednotlivé vrcholy kontur, ze kterých se polygon skládá a zruší se objekt pro teselaci.

3.3.10. Modul promitani

Modul obsahuje definice funkcí zodpovědných za vykreslení OpenGL scény týkající se tématu *projektivní prostory a perspektivní promítání*. Začneme funkčním prototypem:

```
void PerspektivniPromitani();
```

Jedná se o scénu v prostoru, volba *modelview* matice a dalších parametrů včetně úhlů natočení scény je stejná jako v případě funkce v modulu **souradnice**. Následuje vykreslení souřadné soustavy pro lepší orientaci v prostoru a vykreslení krychle. Krychle je dána pouze jedním bodem a délkou hrany, ostatní vrcholy se proto dále musí dopočítat. Dalším krokem je vykreslení projektivní roviny. Pro toto vykreslení je zapnut blending – rovina je poloprůhledná, aby za ni bylo vidět. Zbývá už jen aby se matice zobrazení 4×4 vynásobila s každým vrcholem krychle. Dostanou se tak body, které budou ležet v projektivní rovině. Krychle se do ní tedy takto promítne.

Implementace funkcí pro ovládání scény pomocí klávesnice a myši jsou stejné jako v modulu **souradnice**. Jelikož jde o scénu v prostoru, lze jí pouze různě otáčet a přibližovat či oddalovat. Funkční prototypy jsou:

```
void PerspektivniPromitaniKlavesnice();
void PerspektivniPromitaniMys();
```

Stejně jako v ostatních modulech jsou zde definovány funkce pro výpis textu, ukládání do a načítání ze souboru a obsluha smyčky zpráv pro levý toolbox. Jejich definice jsou založeny na stejných principech.

3.3.11. Třída FPS

Jde o velmi jednoduchou třídu, která má pouze dvě metody:

`void Vypocet();`
Obsahuje čítač průchodů, každou sekundu vypíše jeho hodnotu do stavového řádku a poté vynuluje. Tato hodnota se uloží do proměnné `GLfloat fps`.

`inline GLfloat GetFPS();`
Pouze vrátí hodnotu `fps`. Tato metoda je volána vždy, když je scénou hýbáno pomocí klávesnice, aby byla zachována konstantní rychlost pohybu.

Třída se tedy pouze stará o výpočet aktuální rychlosti vykreslování OpenGL scény (počet vykreslení za sekundu).

3.4. Tvorba nápovědy

Aplikace obsahuje vestavěnou nápovědu ve formátu *CHM*. Tento formát je zkompilovaný soubor nápovědy pro Windows ve formátu *HTML*. Soubor nápovědy byl vytvořen pomocí programu *HelpNDoc*, který je zdarma k osobnímu použití a lze jej získat z http://www.helpndoc.com.

3.5. Tvorba instalátoru

Instalátor byl vytvořen pomocí programu *Inno Setup*, což je prostředí sloužící k vývoji instalačního průvodce pro prostředí operačních systémů Microsoft

Windows. Programuje se v něm pomocí skriptu, který se také může automaticky vygenerovat jednoduchým průvodcem. *Inno Setup* je šířen jako produkt zdarma s možností použití výsledných instalačních průvodců ke komerčním účelům. Lze jej získat z http://jrsoftware.org/isinfo.php.

3.6. Testování aplikace

Aplikace byla vyvíjena a testována na stroji s operačním systémem Windows Vista, s procesorem *Intel Core2 Duo CPU T7300 @ 2.00GHz 2.00GHz*, 2GB operační paměti RAM a grafickou kartou *ATI Mobility Radeon X2300 HD* s taktem 480MHz a pamětí 128MB.

Dále byla testována na stroji s operačním systémem Windows 7, s procesorem *Intel Pentium Dual CPU E2160 @ 1.80GHz 1.80GHz*, 1GB operační paměti RAM a grafickou kartou *ATI Radeon HD 2600 Pro* s taktem 700MHz a pamětí 256MB.

Nakonec ještě testována na stroji s operačním systémem Windows XP, s procesorem *Intel Pentium M 735 1.4GHz*, 1GB operační paměti RAM a grafickou kartou *ATI Mobility Radeon* s taktem 166MHz a pamětí 16MB.

Při zapnutém antialiasingu a vypisováním textového obsahu do vykreslované scény bylo ovládání aplikace plynulé a bez zpoždění. Vyjímka byla zaznamenána pouze u posledně jmenovaného stroje s nízkým taktem jádra grafické karty a malou video pamětí. Zde klesly hodnoty FPS (indikované vpravo ve stavovém řádku aplikace) řádově na desítky a pohybování objektů pomocí myši bylo mírně zpožděné. Obecně by nemělo FPS klesnout pod hodnotu 30. Celkově aplikace i se spuštěnou nápovědou spotřebovala 10–11MB paměti RAM.

V odstínech barev byly velmi nepatrné rozdíly, celkové barevné podání scény je však zachováno. Antialiasing byl na všech testovaných strojích celkem dost kvalitní, avšak případné použití v aplikaci bude záležet na jeho podpoře konkrétní grafické karty.

Z výše uvedeného zřejmě chod aplikace a plynulost ovládání podstatně závisí na grafickém akcelerátoru.

4. Uživatelská příručka

4.1. Požadavky na HW

Aplikace je určena pro operační systémy Windows (XP a vyšší) a grafické karty s podporou OpenGL.

Minimální:

Operační systém: Windows XP/Vista/7
Procesor: 1GHz
Operační paměť: 512MB RAM (Vista 1GB RAM)
Grafická karta: 3D grafická karta s 64MB

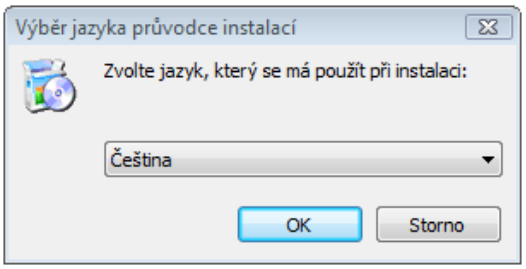
Doporučené:

Operační systém: Windows XP/Vista/7
Procesor: Pentium Core2 Duo nebo Dual CPU 2.00GHz
Operační paměť: 2GB RAM
Grafická karta: 3D grafická karta s 256MB

4.2. Instalace aplikace

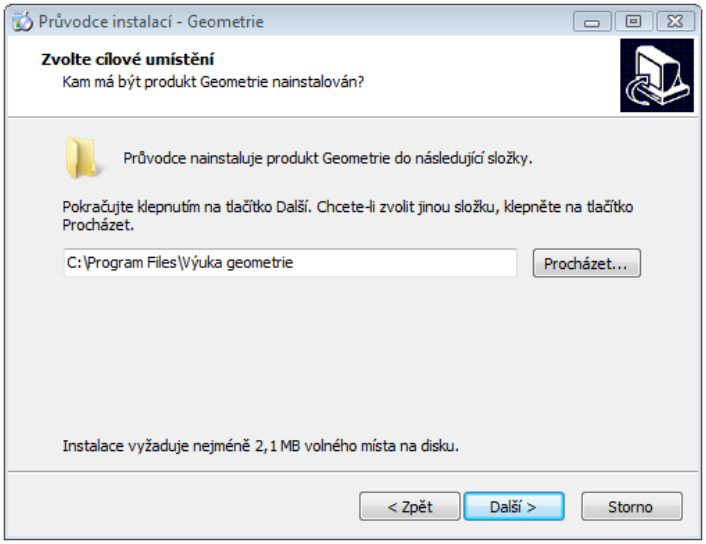
Pro nainstalování aplikace spustíme instalátor *GeometrieSetup*, který se nachází ve složce `bin/` na přiloženém CD (viz příloha E.). Nyní v několika krocích průvodce nastavíme parametry instalace.

Jako první budeme vyzváni k volbě jazyka instalace, na výběr je angličtina a čeština.



Obrázek 6. Volba jazyka instalace

Pokračujeme volbou cílového umístění, kam se má aplikace nainstalovat. Standardně je přednastavena složka s názvem „Výuka geometrie“ umístěná ve složce `C:/Program Files/`.



Obrázek 7. Volba cílového umístění aplikace

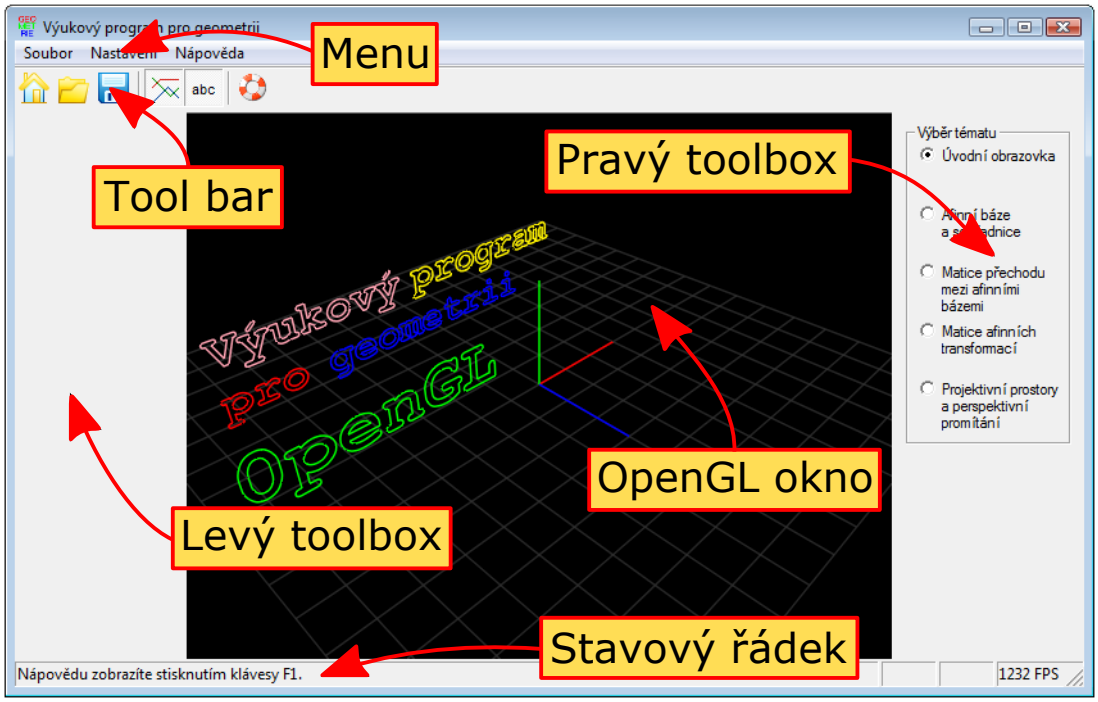
Dále průvodce nabízí možnost vytvořit zástupce aplikace v nabídce Start a na ploše. Následně po potvrzení stiskem tlačítka „Instalovat“, se aplikace nainstaluje. Nakonec po úspěšné instalaci průvodce ohlásí dokončení instalace a zaškrtnutím příslušného políčka lze aplikaci hned spustit.

Obrázek 8. Dokončení instalace aplikace

Aplikaci je možno spustit souborem *Geometrie.exe* v nainstalované složce nebo pomocí zástupce v nabídce Start či na ploše, pokud si uživatel při instalaci tuto volbu zvolil.

4.3. Ovládání aplikace

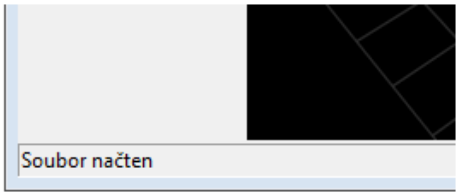
Ovládání aplikace je velmi jednoduché a intuitivní. Na následujícím obrázku 9. je vyobrazeno jak vždy vypadá aplikace po spuštění a zároveň je popsána její struktura.



Obrázek 9. Spuštěná aplikace s úvodní obrazovkou

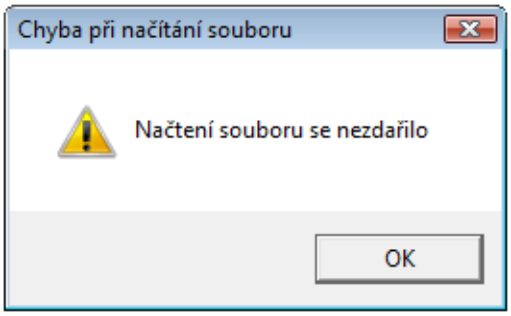
Menu

Nabídka **Soubor** - obsahuje položky pro načtení a uložení souboru, dále položku pro ukončení celé aplikace. Ukládání resp. načítání spočívá v uložení resp. načtení všech editovatelných hodnot ze všech témat a je možné kdykoliv v průběhu práce s aplikací. Neukládají se ani nenačítají hodnoty natočených kamer (pohled na scénu) nebo posunutí scény. Ukládané a načítané soubory mají příponu **.geo*. O úspěšnosti uložení či načtení je uživatel informován zprávou ve stavovém řádku.



Obrázek 10. Informování uživatele o úspěšném načtení souboru

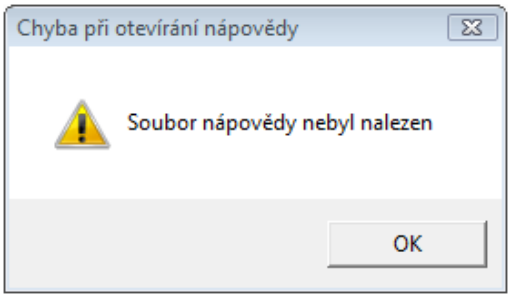
V případě, že je uložení či načtení neúspěšné, zobrazí se chybová zpráva v dialogovém okně. Ukončení aplikace předchází dialogové okno s možností potvrdit či zrušit tuto akci.



Obrázek 11. Neúspěšné načtení souboru

Nabídka **Nastavení** - obsahuje položku pro zapnutí/vypnutí vyhlazování čar a bodů (antialiasingu) v OpenGL okně, dále položku pro zapnutí/vypnutí vypisování textu v OpenGL okně, to vše nezávisle na zvoleném tématu.

Nabídka **Nápověda** - obsahuje položku pro zobrazení vestavěné nápovědy k aplikaci a položku pro zobrazení informací o aplikaci. Nápověda je obsažena v souboru *Napoveda.chm*, což je komprimovaný soubor nápovědy. Pro přístupnost nápovědy integrované v aplikaci je nutné, aby byl tento soubor ve stejné složce jako soubor *Geometrie.exe*. V opačném případě by se při pokusu o zobrazení objevila chybová zpráva v dialogovém okně a k zobrazení nápovědy by nedošlo.



Obrázek 12. Neúspěšné vyvolání nápovědy

Tool bar

Některé funkce z menu jsou pro pohodlnější ovládání aplikace umístěny na toolbaru. Při najetí kurzoru myši nad libovolné tlačítko se zobrazí tzv. „tooltip“, aby bylo ihned zřejmé, kterou funkci z menu zastupuje. V případě antialiasingu a vypisování textu na OpenGL obrazovku jde o stavové funkce, proto mají tato tlačítka po dobu jejich aktivnosti efekt zamáčknutí. Snadno se tak indikuje, zda jsou funkce v danou chvíli zapnuté nebo vypnuté.

Obrázek 13. Tool bar

Levý toolbox

Obsahuje konkrétní prvky k ovládání vybraného tématu. Při zvolené úvodní obrazovce neobsahuje žádné prvky. Scénu v OpenGL okně úvodní obrazovky totiž nelze nijak specifičtěji ovládat.

Pravý toolbox

Obsahuje po celou dobu běhu aplikace možnost zvolit si konkrétní téma z afinní geometrie, které má být demonstrováno. Mezi tématy lze libovolně přepínat a data se po celou dobu práce s aplikací neztratí. Ztráta dat nastane při ukončení aplikace bez předchozího uložení.

OpenGL okno

V tomto okně se vykresluje OpenGL scéna zvoleného tématu.

Stavový řádek

Informuje uživatele o vybraných stavech. První kolonka pak zejména o úspěchu nebo neúspěchu uložení či načtení souboru. Druhá a třetí kolonka slouží pro konkrétní potřeby témat a poslední kolonka informuje uživatele o aktuální rychlosti vykreslování snímků za sekundu v OpenGL okně (FPS = frames per second).

4.3.1. Obecně k ovládání témat

Každé téma lze ovládat pomocí klávesnice nebo myši a má své ovládací prvky na levém toolboxu. Konkrétní ovládání je popsáno pro každé téma zvlášť:

- téma *Afinní báze a souřadnice* 4.3.2.
- téma *Matice přechodu mezi afinními bázemi* 4.3.3.
- téma *Matice afinních transformací* 4.3.4.
- téma *Projektivní prostory a perspektivní promítání* 4.3.5.

Obecné principy

Ovládání je jednoduché a intuitivní a funguje u všech témat na stejném principu. Samozřejmostí je tzv. princip drag and drop. Obecně se na levém toolboxu

zadávají konkrétní hodnoty souřadnic, vektorů, délek, velikostívatd. do editačních polí a následně se aktualizují potvrzovacím tlačítkem. Jelikož se OpenGL okno neustále překresluje, změny se projeví ihned.

PZN: „drag and drop“ = Kurzorem myši najedeme nad objekt, levým tlačítkem myši jej uchopíme a následně táhneme. Po dobu táhnutí musí být levé tlačítko myši stále stisknuté. Puštění tlačítka znamená puštění objektu.

Ekvivalentní ovládání

V případě témat pracujících v rovině, lze číselné hodnoty měnit aktivně pomocí myši přímo v OpenGL okně a obsahy editačních polí na levém toolboxu se ihned aktualizují. Maximálně je dodržena snaha o ekvivalenci mezi ovládáním pomocí levého toolboxu a myši. Popřípadě mezi ovládáním pomocí klávesnice a myši, pokud se jedná o témata pracující v prostoru.

Omezení myši

Omezení plynou z podstaty věci. Nejsou to omezení, která by nějak „limitovala“ uživatele. Jedná se o fakt, že myš je zařízení, které pracuje v rovině. Není tedy možné nějak jednoduše pomocí myši hýbat s body v prostoru (celém). Tento nedostatek je však plně pokryt možností zadat souřadnice bodu ručně v editačním poli na levém toolboxu.

Nyní ke konkrétnímu ovládání jednotlivých témat:

4.3.2. Ovládání tématu: Afinní báze a souřadnice

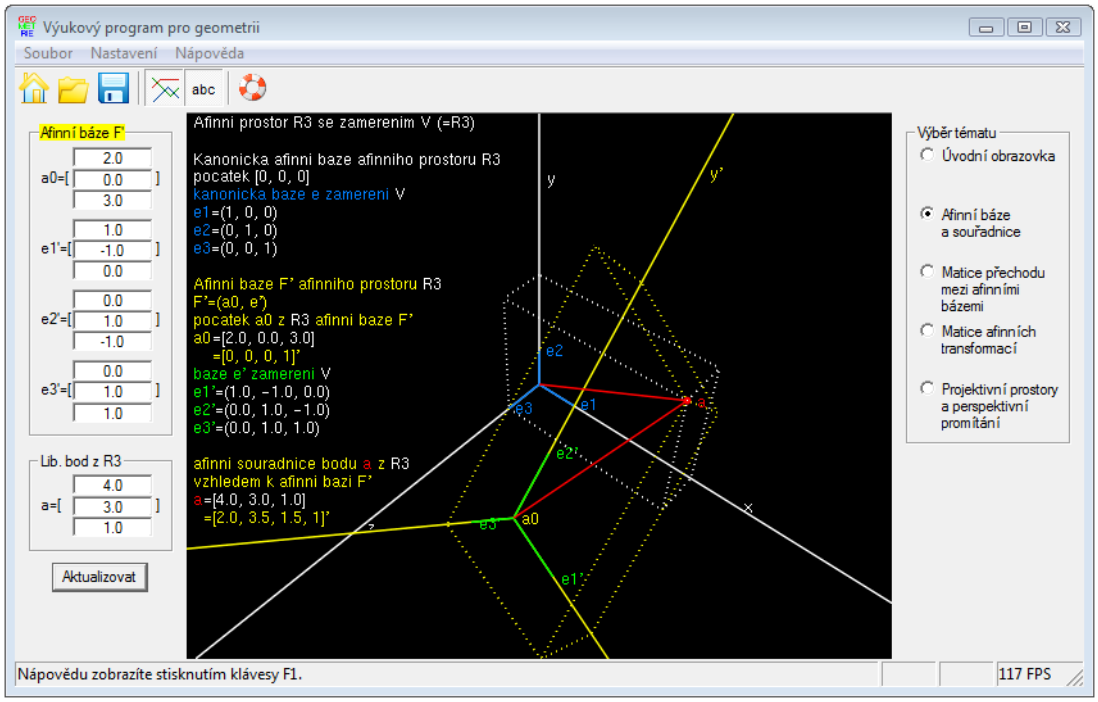
Spuštěná aplikace po volbě tohoto tématu by mohla vypadat obdobně, jako na obrázku 14.

Popis tématu

Máme dán afinní prostor $\mathbf{R}^3$ se zaměřením V. Bod $a_0 \in \mathbf{R}^3$ a e' je báze vektorového prostoru V. Spolu tvoří (žlutou) ***afinní bázi*** afinního prostoru $\mathbf{R}^3$. Nakonec pro libovolný bod $a \in \mathbf{R}^3$ zavádíme ***afinní souřadnice*** vzhledem k žluté afinní bázi.

V OpenGL okně je ještě vykreslena kanonická afinní báze prostoru $\mathbf{R}^3$. Souřadnice počátku a_0 žluté afinní báze jsou vyjádřeny vzhledem k této kanonické afinní bázi, aby bylo umožněno uživateli ho někam v prostoru „umístit“. To platí i pro zadání bodu a, jehož afinní souřadnice vzhledem k žluté afinní bázi program spočítá.

Cílem tématu je vystihnout geometrický význam afinních souřadnic bodu a a dále naznačit, že v afinní soustavě souřadnic nejsou v obecném případě vektory e'_1, e'_2, e'_3 navzájem kolmé a nejsou ani nutně stejně dlouhé.



Obrázek 14. Demonstrace tématu „Afinní báze a souřadnice“

Barevné značení

Text ve výpisu na OpenGL obrazovce barevně maximálně odpovídá vykreslovaným objektům a je rozdělen do čtyř logických celků, oddělených od sebe prázdným řádkem:

1.celek říká, že se jedná o afinní prostor $\mathbf{R}^3$, jehož zaměřením V je samotný prostor $\mathbf{R}^3$. Na V je zavedena kanonická struktura afinního prostoru.

2.celek popisuje bílou barvou kanonickou afinní bázi afinního prostoru, modrou barvou jsou zvýrazněny bázové vektory této kanonické afinní báze.

3.celek popisuje žlutou barvou editovatelnou afinní bázi afinního prostoru, zelenou barvou jsou zvýrazněny bázové vektory této afinní báze.

4.celek popisuje žlutou barvou afinní souřadnice bodu a vzhledem k žluté afinní bázi popsané ve 3. celku.

Bílá barva složek modrých a zelených bázových vektorů vyjadřuje fakt, že tyto vektory jsou vybrány ze zaměření V.

Bílá a žlutá barva hodnot souřadnicových složek bodů pak odpovídá barvám bází, vzhledem ke kterým jsou vyjádřeny.

Červená barva zvýrazňuje vektory vzniklé rozdílem bodů a a počátky afinních bází.

Tečkované úsečky opticky naznačují, kde se bod a v prostoru nachází. Zároveň ukazují na místa na osách, ze kterých jsou určeny souřadnice tohoto bodu vzhledem k příslušné bázi.

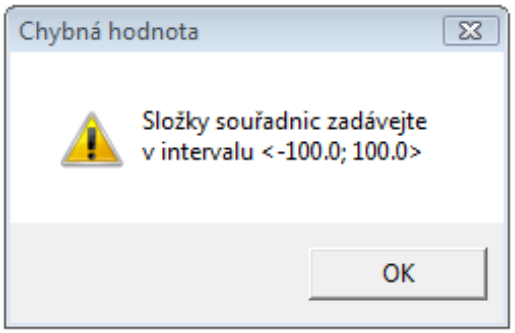
Žluté podbarvení textu „Afinní báze F'“ na levém toolboxu uživateli ihned naznačuje, které afinní báze se hodnoty týkají.

Ovládání levého toolboxu

Na levém toolboxu jsou uvedeny všechny body a vektory, jejichž souřadnicové složky může v tomto tématu uživatel editovat. Stačí zadat hodnotu do editačního políčka a pro potvrzení stisknout tlačítko „Aktualizovat“ nebo klávesu „Enter“. Po potvrzení se všechny hodnoty v editačních polích aktualizují.

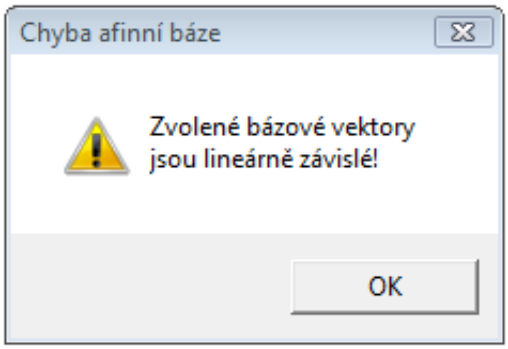
Lze zadat celočíselnou hodnotu, číslo s desetinnou čárkou (počet míst omezuje jen velikost políčka, ovšem hodnota se zaokrouhluje na jedno desetinné místo) nebo nechat políčko prázdné, v tom případě se jako hodnota bere číslo 0.

Omezení: hodnoty je možno zadávat pouze z uzavřeného intervalu $\langle -100.0; 100.0 \rangle$. V případě, že bude zadána jiná hodnota, bude na to uživatel upozorněn chybovou zprávou v dialogovém okně a k aktualizaci těchto hodnot nedojde.



Obrázek 15. Upozornění na chybně zadanou hodnotu

Pokud uživatel zadá hodnoty složek bázových vektorů tak, že by byly lineárně závislé (a nemohly tak tvořit bázi vektorového prostoru), bude na to upozorněn chybovou zprávou v dialogovém okně a k aktualizaci těchto hodnot nedojde.



Obrázek 16. Upozornění na chybně zadanou vektorovou bázi

Ovládání pomocí klávesnice

Pomocí směrových tlačítek (šipek)
vlevo, vpravo: rotace scény kolem vertikály vedené středem OpenGL okna.
nahoru, dolů: rotace scény kolem horizontály vedené středem OpenGL okna.

Page Up: přiblížení scény.
Page Down: oddálení scény.

Ovládání pomocí myši

Pomocí **scroll**-ovacího kolečka se scéna přibližuje a oddaluje.
Principem drag and drop kdekoliv v OpenGL okně se scéna rotuje, přitom pohyb myši do stran odpovídá směrovým tlačítkům na klávesnici.
Stisknutí levého tlačítka myši je provázeno změnou kurzoru „šipky“ na „ručičku“.

Praktické použití

Toto téma nám může posloužit při řešení řady úloh. Jako příklad uveďme následující:

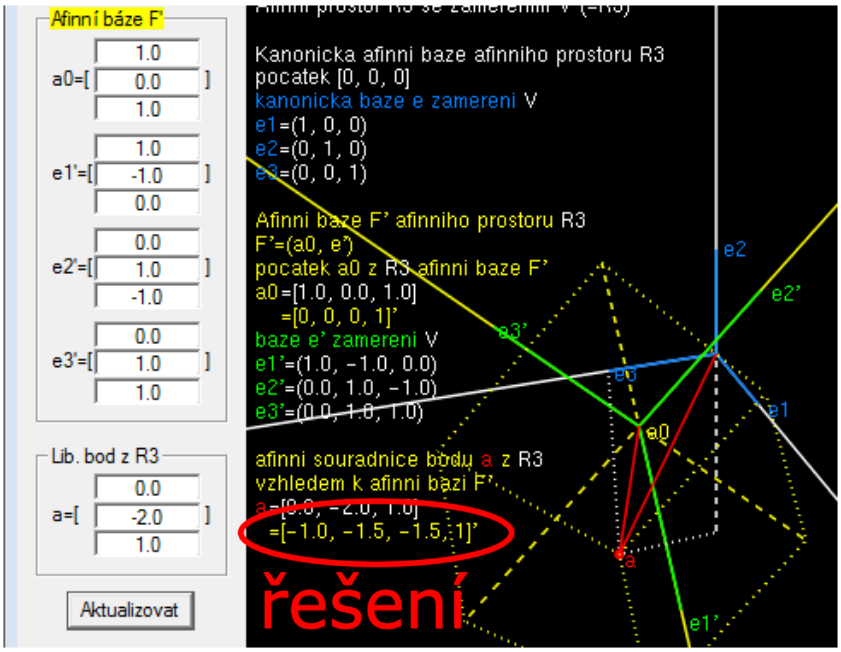
Př.: Nalezněme souřadnice bodu $a = [0, -2, 1] \in \mathbf{R}^3$ vzhledem k afinní bázi $\varphi = (a_0, e)$, kde

$$a_0 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, e_1 = \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}, e_2 = \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix}, e_3 = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}.$$

Řešení dostaneme jednoduše zadáním všech hodnot do příslušných editačních políček na levém toolboxu a pro potvrzení stiskneme tlačítko „Aktualizovat“ nebo „Enter“ na klávesnici. Na obrázku 17. je zobrazen výsledek úlohy:

$$a_\varphi = \begin{bmatrix} -1 \\ -1.5 \\ -1.5 \\ 1 \end{bmatrix}.$$

Všimněme si ještě, že čárkovaně (nikoliv tečkovaně) je naznačeno protažení souřadnicových os do záporné části.



Obrázek 17. Řešení příkladu

4.3.3. Ovládání tématu: Matice přechodu mezi afinními bázemi

Spuštěná aplikace po volbě tohoto tématu by mohla vypadat obdobně, jako na obrázku 18.

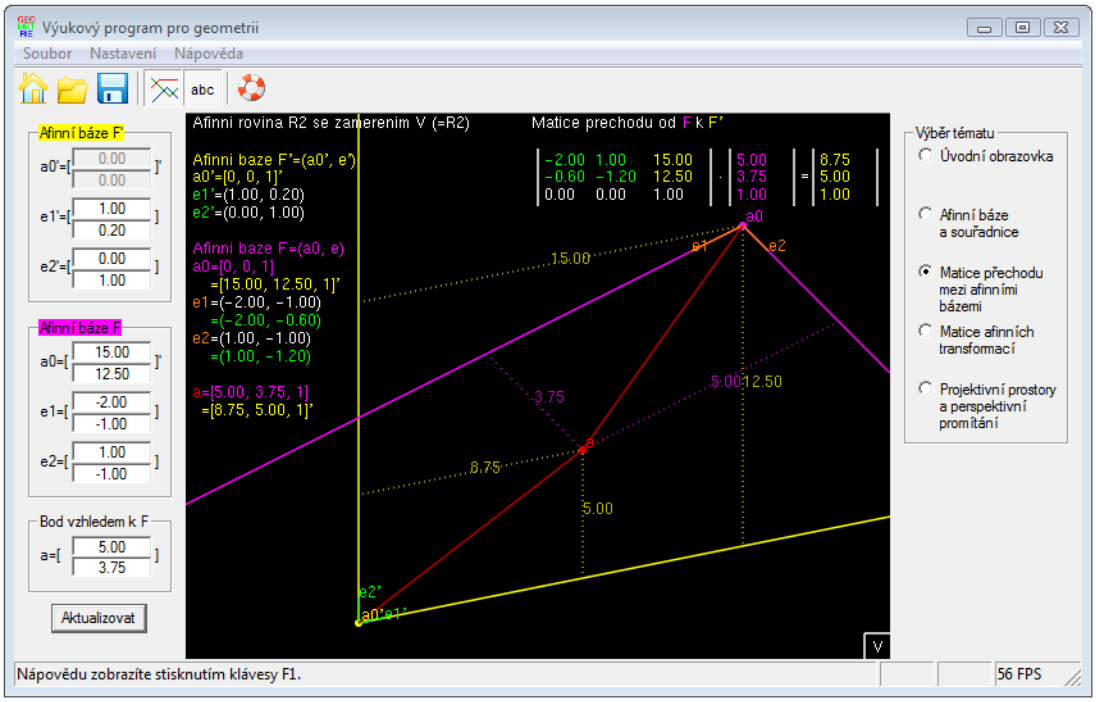
Popis tématu

Máme dány dvě afinní báze (žlutá a fialová). Známe jejich vzájemné vztahy:

- souřadnice počátku fialové afinní báze vzhledem k žluté afinní bázi.
- souřadnice vektorů vektorové báze fialové afinní báze (vyznačeny oranžově) vzhledem k vektorové bázi žluté afinní báze (vyznačeny zeleně) a naopak.

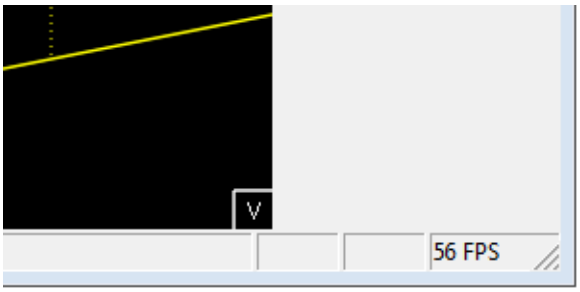
Nyní chceme najít:
afinní souřadnice bodu a vzhledem k žluté afinní bázi, známe-li afinní souřadnice tohoto bodu vzhledem k fialové afinní bázi.

Příslušná ***matice přechodu*** od fialové afinní báze k žluté afinní bázi je zobrazena v pravém horním rohu OpenGL okna. Tato matice je násobena maticí představující afinní souřadnice bodu a vzhledem k fialové afinní bázi. Výsledkem tohoto násobení je matice představující hledané afinní souřadnice bodu a vzhledem k žluté afinní bázi.

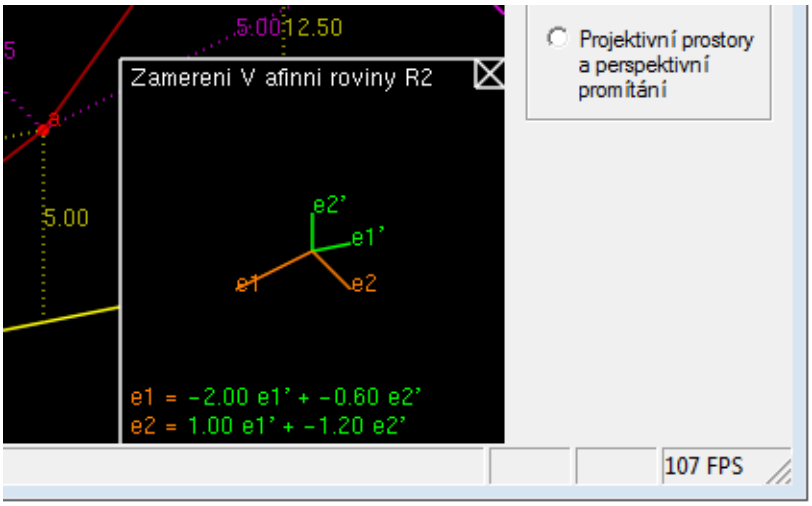


Obrázek 18. Demonstrace tématu „Matice přechodu mezi afinními bázemi“

Vztahy mezi vektorovými bázemi je možno zobrazit v okénku zaměření V (obr. 20.), jehož vyvolání lze provést stisknutím klávesy „V“ nebo najetím kurzoru myši nad značku, která se nachází v pravém dolním rohu OpenGL okna (obr. 19.).



Obrázek 19. Značka okénka zaměření V



Obrázek 20. Zobrazené okénko zaměření

Barevné značení

Text ve výpisu na OpenGL obrazovce barevně maximálně odpovídá vykreslovaným objektům a je rozdělen do čtyř logických celků, oddělených od sebe prázdným řádkem:

1.celek říká, že se jedná o afinní prostor $\mathbf{R}^2$, jehož zaměřením V je samotný prostor $\mathbf{R}^2$. Na V je zavedena kanonická struktura afinního prostoru.

2.celek popisuje žlutou afinní bázi afinního prostoru, zelenou barvou jsou zvýrazněny bázové vektory této žluté afinní báze.

3.celek popisuje fialovou afinní bázi afinního prostoru, oranžovou barvou jsou zvýrazněny bázové vektory této fialové afinní báze.

4.celek popisuje afinní souřadnice bodu a vzhledem k fialovým a žlutým afinním bázím.

Bílá barva složek oranžových a zelených bázových vektorů vyjadřuje fakt, že tyto vektory jsou vybrány ze zaměření V.

Zelená čtveřice čísel v matici přechodu vyjadřuje fakt, že se jedná o souřadnice vektorů oranžové vektorové báze vyjádřené v zelené vektorové bázi.

Žluté podbarvení textu „Afinní báze F'“ a fialové podbarvení textu „Afinní báze F“ na levém toolboxu uživateli ihned naznačuje, kterým afinním bázím hodnoty náleží.

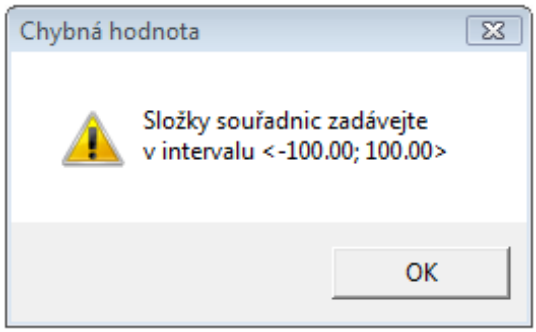
Ovládání levého toolboxu

Na levém toolboxu jsou uvedeny všechny body a vektory, jejichž souřadnicové složky může v tomto tématu uživatel editovat. Stačí zadat hodnotu do editačního

políčka a pro potvrzení stisknout tlačítko „Aktualizovat“ nebo klávesu „Enter“. Po potvrzení se všechny hodnoty v editačních polích aktualizují.

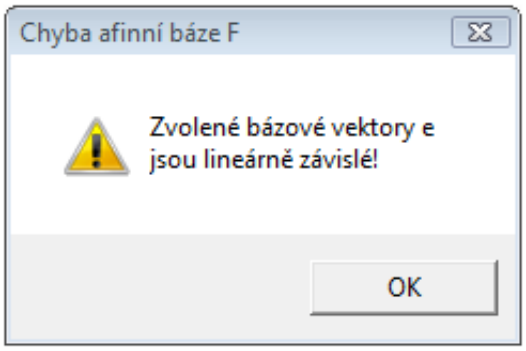
Lze zadat celočíselnou hodnotu, číslo s desetinnou čárkou (počet míst omezuje jen velikost políčka, ovšem hodnota se zaokrouhluje na dvě desetinná místa) nebo nechat políčko prázdné, v tom případě se jako hodnota bere číslo 0.

Omezení: hodnoty je možno zadávat pouze z uzavřeného intervalu $\langle -100.00; 100.00 \rangle$. V případě, že bude zadána jiná hodnota, bude na to uživatel upozorněn chybovou zprávou v dialogovém okně a k aktualizaci těchto hodnot nedojde.



Obrázek 21. Upozornění na chybně zadanou hodnotu

Pokud uživatel zadá hodnoty složek bázových vektorů tak, že by byly lineárně závislé (a nemohly tak tvořit bázi vektorového prostoru), bude na to upozorněn chybovou zprávou v dialogovém okně a k aktualizaci těchto hodnot nedojde.



Obrázek 22. Upozornění na chybně zadanou vektorovou bázi

Ovládání pomocí klávesnice

Pomocí směrových tlačítek (šipek)
vlevo, vpravo, nahoru, dolů: pohybuje celou scénou v OpenGL okně daným směrem.

V: zobrazí nebo skryje okénko se zaměřením V.

Ovládání pomocí myši

Při najetí kurzoru myši nad objekt, se kterým lze pohybovat, se tento kurzor změní na „ručičku“.

Myší lze:

- hýbat s body (měnit jejich souřadnice).
- měnit velikost a směr vektorů vektorových bází.
- zobrazit a skrýt okénko se zaměřením V.

První dvě operace jsou ekvivalentní s „ručním“ zadáváním hodnot v levém toolboxu, poslední je ekvivalentní se stiskem klávesy „V“.

Pokud některá souřadnicová složka dosáhne hranice intervalu vymezeného pro hodnoty složek, dále už pohyb za tuto hranici není možný.

Vyjímku tvoří pouze pohyb počátku žluté báze, ten je ekvivalentní stisku směrových tlačítek (šipek), tj. pohybuje se s celou scénou v OpenGL okně (souřadnice tohoto bodu v OpenGL okně nejsou pro uživatele ani demonstraci tématu nijak důležité).

Velikost a směr vektorů se mění tak, že se kurzorem myši najede nad jeho současný konec a použije princip drag and drop.

V aplikaci je zabráněno úmyslnému (i neúmyslnému) vytvoření lineární závislosti bázových vektorů, a to tak, že se případná kolidující hodnota souřadné složky ovládaného vektoru nikdy neaktualizuje a zároveň se taková poloha vektoru ani nevykreslí. Vizuálně to lze pozorovat „přeskočením“ této polohy při dostatečně jemném pohybu myši.

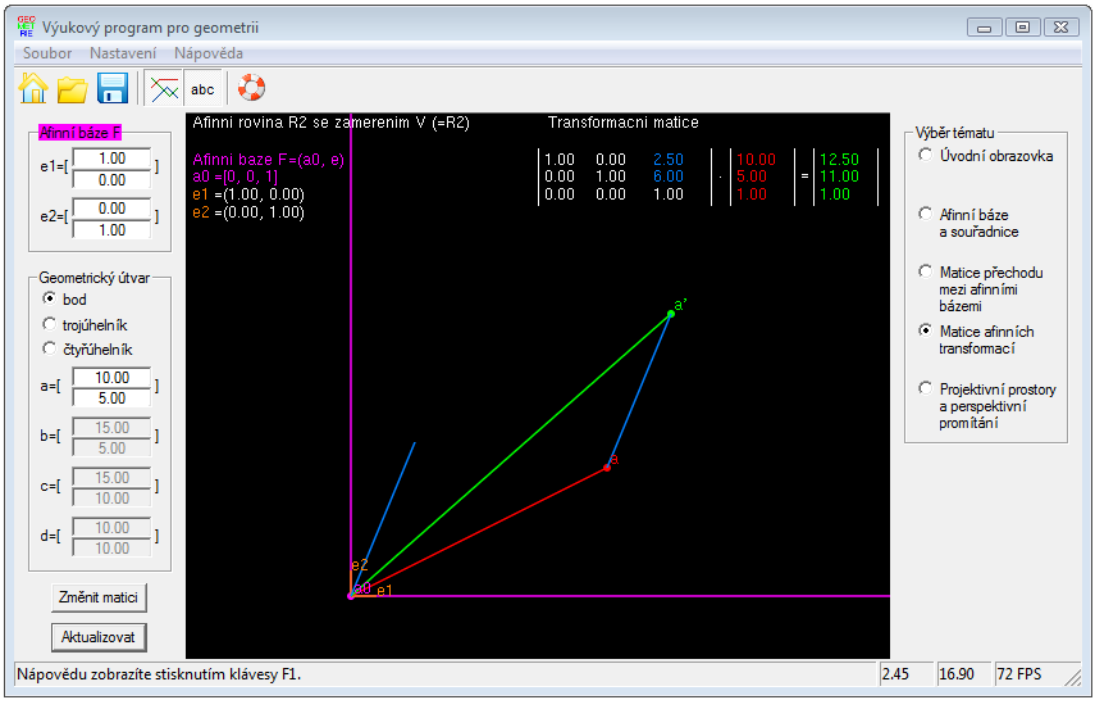
Najetím kurzoru myši nad značku „V“ umístěnou v pravém dolním rohu OpenGL okna, se vyvolá okénko se zaměřením V. Toto okénko lze opět skrýt kliknutím na křížek, který se nachází v pravém horním rohu tohoto okénka. V okénku nelze hýbat žádnými zobrazenými objekty.

4.3.4. Ovládání tématu: Matice afinních transformací

Spuštěná aplikace po volbě tohoto tématu by mohla vypadat obdobně, jako na obrázku 23.

Popis tématu

Máme dánu afinní bázi (fialovou) a nějaký rovinný geometrický útvar (červený). Útvar si můžeme zvolit, nabízí se bod, trojúhelník nebo čtyřúhelník. Tento útvar budeme následně transformovat. Příslušný transformovaný útvar se zobrazí zeleně.



Obrázek 23. Demonstrace tématu „Matice afinních transformací“

Transformace je dána ***maticí afinní transformace***, která je zobrazena v pravém horním rohu OpenGL okna. Na výběr je matice reprezentující translaci, rotaci, změnu měřítka a libovolnou jinou transformaci.

Transformační matice je násobena maticí představující afinní souřadnice konkrétně zvoleného bodu (červeného) vzhledem k fialové afinní bázi. Výsledkem tohoto násobení je matice představující hledané afinní souřadnice příslušného transformovaného (zeleného) bodu vzhledem k fialové afinní bázi.

Barevné značení

Text ve výpisu na OpenGL obrazovce barevně maximálně odpovídá vykreslovaným objektům.

Na V je zavedena kanonická struktura afinního prostoru. Jedná se o afinní prostor $\mathbf{R}^2$, jehož zaměřením V je samotný prostor $\mathbf{R}^2$.

Bílá barva složek oranžových bázových vektorů vyjadřuje fakt, že tyto vektory jsou vybrány ze zaměření V.

Červenou barvou je označen původní (netransformovaný) útvar.

Zelenou barvou je označen příslušný již transformovaný útvar.

Červená úsečka zvýrazňuje vektor vzniklý rozdílem konkrétně vybraného původního bodu a počátkem fialové afinní báze.

Zelená úsečka zvýrazňuje vektor vzniklý rozdílem příslušného transformovaného bodu a počátkem fialové afinní báze.

Modrá úsečka zvýrazňuje vektor posunutí, pokud je zvolena translační matice.

Modrá křivka (oblouk) mezi konkrétně vybraným původním a příslušným transformovaným bodem zvýrazňuje úhel rotace, pokud je zvolena rotační matice.

Fialové podbarvení textu „Afinní báze F“ na levém toolboxu uživateli ihned naznačuje, které afinní báze se hodnoty týkají.

Transformační matice

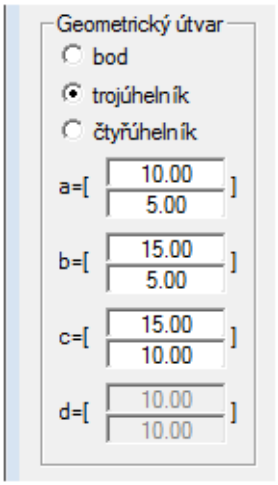
Prvky matice jsou barevně rozlišeny. Modrou barvou jsou označeny prvky, které mají vliv na konkrétně zvolenou transformaci a jdou tedy editovat. Ostatní prvky jsou pak označeny bílou barvou a pro danou matici jsou pevně dány.

Ovládání levého toolboxu

Ovládání je stejné jako bylo popsáno v předchozí kapitole v odstavci Ovládání levého toolboxu. Navíc se však v tomto tématu ještě ovládá:

Volba geometrického útvaru

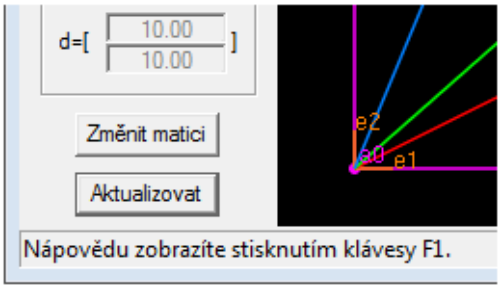
Jsou celkem tři možnosti volby: *bod, trojúhelník a čtyřúhelník*. Pro volbu jednoho z nich stačí kliknout na příslušné políčko a poté se aktivují potřebná editační pole pro zadávání souřadnic bodů, které zvolený útvar definují.



Obrázek 24. Volba geometrického útvaru

Změna transformační matice

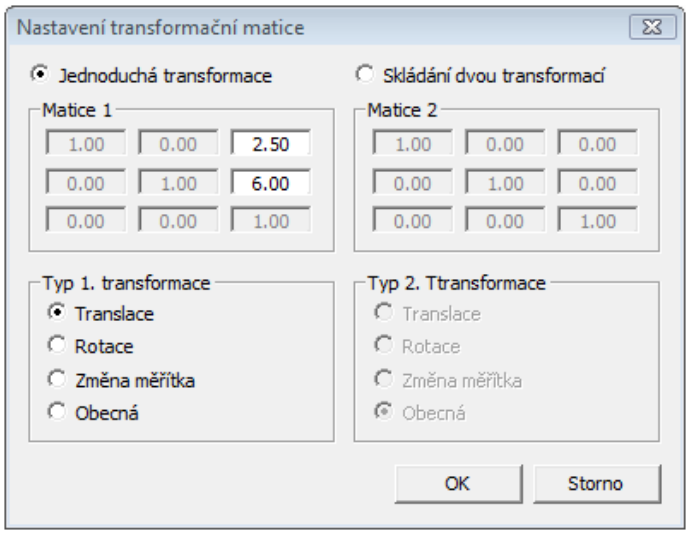
Pro možnost změny stačí kliknout na tlačítko „Změnit matici“, které se nachází ve spodní části levého toolboxu.



Obrázek 25. Tlačítko pro vyvolání dialogu nastavení transformační matice

Lze si volit matici určující tyto typy transformací: *translace, rotace, změna měřítka* a *obecná*. Volba každé z těchto typů transformací zaručuje, že bude provedena právě tato. To je zajištěno možnou editací pouze určitých prvků matice. Hodnoty lze zadat do aktivních editačních polí a potvrdit tlačítkem „OK“.

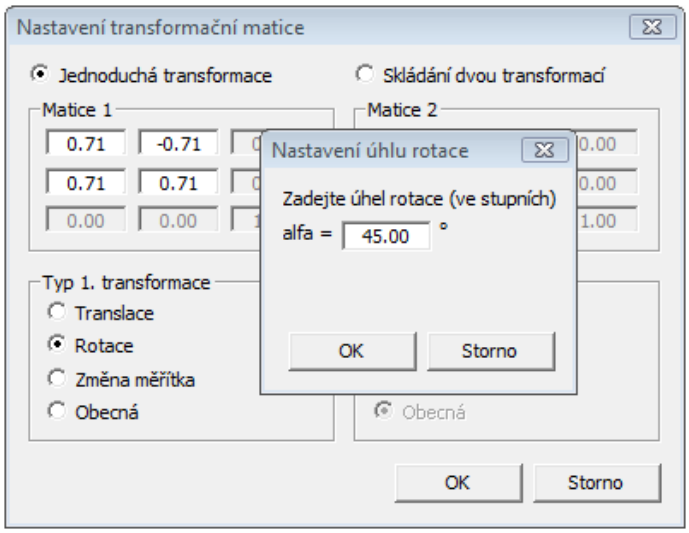
Kromě transformace určené pouze jednou maticí je možné transformaci složit ze dvou matic, stačí přitom kliknout na příslušné políčko, které se nachází vpravo nahoře vyvolaného dialogu nastavení (označeno jako „*Skládání dvou transformací*“). V takovém případě se matice vynásobí v pořadí jak jsou uvedeny v dialogu nastavení a v OpenGL okně se vypíše výsledná matice vzniklá tímto násobením.



Obrázek 26. Dialog nastavení transformační matice

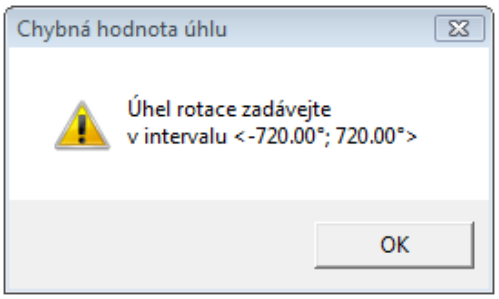
V případě, že bude zvolen typ transformace „Rotace“, a uživatel bude chtít změnit editovatelné prvky matice, pak po kliknutí do libovolného aktivního editačního pole se zobrazí další dialogové okno s volbou pro zadání úhlu rotace ve stupních. Program sám vypočítá konkrétní hodnoty prvků matice ze zadaného

úhlu. Kromě ulehčení práce uživatele je tak ještě zajištěno, že se bude skutečně jednat o rotaci.



Obrázek 27. Volba úhlu rotace

Omezení: hodnotu je možno zadávat pouze z uzavřeného intervalu $\langle -720.00; 720.00 \rangle$. V případě, že bude zadána jiná hodnota, bude na to uživatel upozorněn chybovou zprávou v dialogovém okně a k aktualizaci této hodnoty nedojde.



Obrázek 28. Upozornění na chybně zadanou velikost úhlu

Ovládání pomocí klávesnice

Pomocí směrových tlačítek (šipek)
vlevo, vpravo, nahoru, dolů: pohybuje celou scénou v OpenGL okně daným směrem.

Ovládání pomocí myši

Při najetí kurzoru myši nad objekt, se kterým lze pohybovat, se tento kurzor změní na „ručičku“.

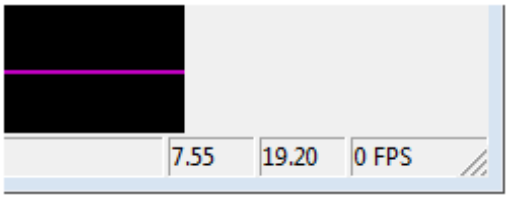
Myší lze:

- hýbat s konkrétními body (měnit jejich souřadnice).
- hýbat s celými objekty (pokud to typ transformace dovoluje).
- měnit velikost a směr vektorů vektorové báze.
- určovat konkrétní transformace.

Pokud některá souřadnicová složka dosáhne hranice intervalu vymezeného pro hodnoty složek, dále už pohyb za tuto hranici není možný.

Vyjímku tvoří pouze pohyb počátku fialové báze, ten je ekvivalentní stisku směrových tlačítek (šipek), tj. pohybuje se s celou scénou v OpenGL okně (souřadnice tohoto bodu v OpenGL okně nejsou pro uživatele ani demonstraci tématu nijak důležité).

Souřadnice pozice kurzoru myši ve fialové afinní bázi jsou zobrazeny ve dvou kolonkách ve stavovém řádku vpravo.



Obrázek 29. Zobrazení aktuálních souřadnic pozice kurzoru myši

Najetí kurzoru myši nad konkrétní vrchol a, b, c, d, znamená výběr tohoto vrcholu pro výpočet jeho souřadnic po transformaci a', b', c', d'. Výpočet je zobrazen v pravém horním rohu OpenGL okna (násobení transformační maticí).

Velikost a směr vektorů se mění tak, že se kurzorem myši najede nad jeho současný konec a použije princip drag and drop.

V aplikaci je zabráněno úmyslnému (i neúmyslnému) vytvoření lineární závislosti bázových vektorů, a to tak, že se případná kolidující hodnota souřadné složky ovládaného vektoru nikdy neaktualizuje a zároveň se taková poloha vektoru ani nevykreslí. Vizuálně to lze pozorovat „přeskočením“ této polohy při dostatečně jemném pohybu myši.

Podle typu zvolené transformace se přizpůsobuje i ovládání pomocí myši. Pro každý typ transformace je trochu specifické. Obecné principy jsou však vždy zachovány.

Translace
Pohybovat lze:

- dle typu zvoleného geometrického útvaru samostatně vrcholy a, b, c, d (indikováno bílým zvýrazněním).
- celým červeným (původním netransformovaným) útvarem.
- celým zeleným (transformovaným) útvarem = změna vektoru posunutí (libovolný pohyb).

Rotace
Pohybovat lze:

- dle typu zvoleného geometrického útvaru samostatně vrcholy a, b, c, d (indikováno bílým zvýrazněním).
- celým červeným (původním netransformovaným) útvarem.
- celým zeleným (transformovaným) útvarem = změna úhlu rotace (pohyb po oblouku, křivce).

Změna měřítka
Pohybovat lze:

- dle typu zvoleného geometrického útvaru samostatně vrcholy a, b, c, d (indikováno bílým zvýrazněním).
- celým červeným (původním netransformovaným) útvarem.
- celým zeleným (transformovaným) útvarem = změna velikosti útvaru závislá na směru pohybu.

Obecná
Pohybovat lze:

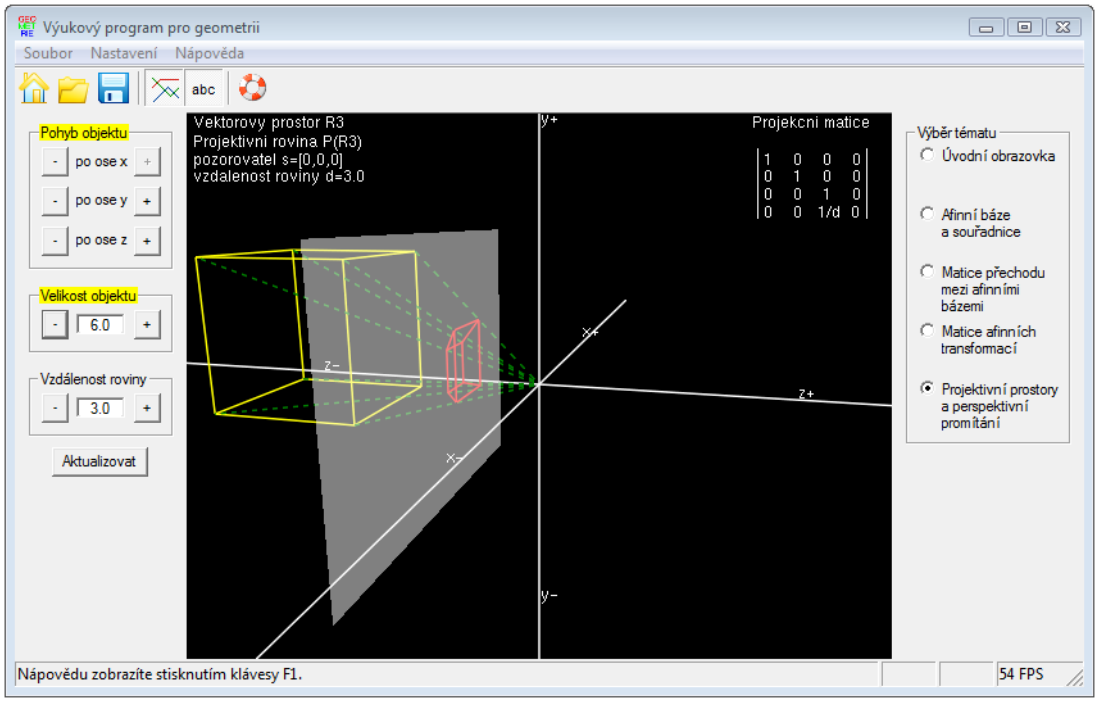
- dle typu zvoleného geometrického útvaru samostatně vrcholy a, b, c, d (indikováno bílým zvýrazněním).
- celým červeným (původním netransformovaným) útvarem.
- v případě zvoleného geometrického útvaru *trojúhelník* nebo *čtyřúhelník* samostatně vrcholy a', b', c', d' = jednoznačné určení transformační matice (indikováno bílým zvýrazněním).
- celým zeleným (transformovaným) útvarem = změna vektoru posunutí (libovolný pohyb).

V případě volby složené transformace ze dvou matic stejného typu, např. *rotace* a *rotace*, zůstává výše uvedené ovládání v platnosti. Měnit se budou vždy prvky druhé matice.

Pokud budou matice jiného typu, např. složení *translace* a *rotace*, pak je ovládání mírně omezené, a to tak, že nelze nijak pohybovat zeleným (transformovaným) útvarem.

4.3.5. Ovládání tématu: Projektivní prostory a perspektivní promítání

Spuštěná aplikace po volbě tohoto tématu by mohla vypadat obdobně, jako na obrázku 30.



Obrázek 30. Demonstrace tématu „Projektivní prostory a perspektivní promítání“

Popis tématu

Máme dán vektorový prostor $\mathbf{R}^3$, dále ***projektivní prostor*** (rovinu) $\mathrm{P}(\mathbf{R}^3)$. V prostoru je umístěna žlutá krychle a chceme tuto krychli promítnout do promítací roviny. Průmět (červený) vznikne z průsečíků promítacích paprsků (zelených čárkovaných), vycházejících ze středu promítání do každého vrcholu krychle, s promítací rovinou.

Střed promítání je v počátku souřadnicového systému, promítací rovina je na ose z ve vzdálenosti d od středu promítání. Takto je definováno ***perspektivní promítání***.

Vyjádření této projekce v maticovém tvaru je zobrazeno v pravém horním rohu OpenGL oka.

Z důvodu lepší názornosti je vykreslena pouze část promítací roviny, ovšem vždy tak, aby promítaný objekt v této části ležel. To je zařízeno adekvátním zvětšením části roviny, v případě že by se měl objekt promítnout mimo.

Barevné značení

Žlutou barvou je vykreslený objekt (krychle), který se má promítnout.

Červenou barvou je vykreslený promítnutý objekt na promítací rovině.

Poloprůhledně je vybarvená promítací rovina, aby bylo vidět i za ni.

Zelenou barvou jsou vybarveny promítací paprsky.

Žluté podbarvení textu „Pohyb objektu“ a „Velikost objektu“ na levém toolboxu uživateli ihned naznačuje, kterého objektu v OpenGL okně se ovládání týká.

Ovládání levého toolboxu

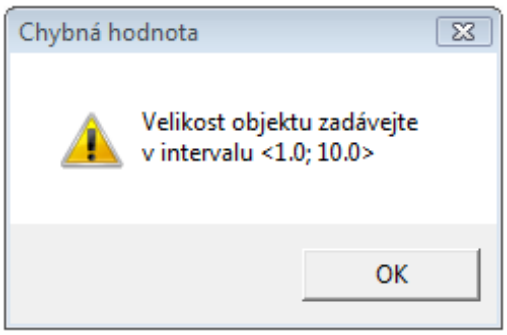
Na levém toolboxu je umístěno ovládání pro pohyb krychle po osách x, y, z. To představují tlačítka „+“ a „-“, směr pohybu je pak příznačně označen uvedením znaménka „+“ a „-“ za popisem konkrétní osy v OpenGL okně.

Stejným typem tlačítek je dále možno měnit velikost krychle, kde označení:
„+“ značí zvětšení velikosti o hodnotu 1.0,
„-“ značí zmenšení velikosti o hodnotu 1.0,
a vzdálenost d promítací roviny od středu promítání, kde označení:
„+“ značí zvětšení vzdálenosti d o hodnotu 1.0,
„-“ značí zmenšení vzdálenosti d o hodnotu 1.0.

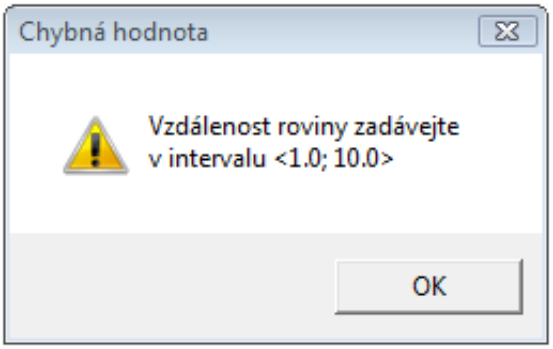
Velikost krychle a vzdálenost promítací roviny lze rovněž zadat přímo do příslušného editačního políčka a pro potvrzení stisknout tlačítko „Aktualizovat“ nebo klávesu „Enter“. Po potvrzení se všechny hodnoty v editačních polích aktualizují.
Lze zadat celočíselnou hodnotu, číslo s desetinnou čárkou (počet míst omezuje jen velikost políčka, ovšem hodnota se zaokrouhluje na jedno desetinné místo) nebo nechat políčko prázdné, v tom případě se jako hodnota bere číslo 0.

Omezení: hodnoty je možno zadávat pouze z uzavřeného intervalu $\langle 1.0; 10.0 \rangle$. V případě, že bude zadána jiná hodnota, bude na to uživatel upozorněn chybovou zprávou v dialogovém okně a k aktualizaci těchto hodnot nedojde.

Při ovládání tlačítky „+“ a „-“ je omezení indikováno zašednutím (deaktivací) příslušného tlačítka a není tak možné dostat se mimo povolený interval.



Obrázek 31. Upozornění na chybně zadanou hodnotu



Obrázek 32. Upozornění na chybně zadanou hodnotu

Ovládání pomocí klávesnice

Pomocí směrových tlačítek (šipek)
vlevo, vpravo: rotace scény kolem vertikály vedené středem OpenGL okna.
nahoru, dolů: rotace scény kolem horizontály vedené středem OpenGL okna.

Page Up: přiblížení scény.
Page Down: oddálení scény.

Ovládání pomocí myši

Pomocí **scroll**-ovacího kolečka se scéna přibližuje a oddaluje.
Principem drag and drop kdekoliv v OpenGL okně se scéna rotuje, přitom pohyb myši do stran odpovídá směrovým tlačítkům na klávesnici.
Stisknutí levého tlačítka myši je provázeno změnou kurzoru „šipky“ na „ručičku“.

Závěr

Cílem práce bylo vytvořit výukový program, který by názorně předvedl základní pojmy afinní geometrie. Byl přitom kladen důraz na názornost, jednoduché ovládání a uživatelsky přívětivé prostředí.

Vytvořená aplikace pokrývá základní teoretické pojmy přednášené v kurzu *Geometrie 2*. Je určena především pro studenty, jako pomůcka při studiu tohoto kurzu, ale například i pro širší skupinu uživatelů zainteresovaných v oblasti geometrie.

Nesporná výhoda spočívá zejména v možnosti řešit pomocí této aplikace množství (školních) příkladů. Uživatel má možnost si sám měnit různé hodnoty, přepínat a nastavovat parametry a tvořit tak vlastně nové zadání úloh. Nejde tedy jen o holé demonstrování pojmů se slabou nebo téměř žádnou interakcí s uživatelem.

Conclusions

The goal of this work was create an education software for presentation basic concepts of affine geometry. We emphasis on clearness, simple control and user friendly environment.

The software cover basic theoretic concepts of *Geometry 2* course. As an utility is determined mainly for students for study this course and people with interest in Geometry at all.

The main advantage is in possibility to solve many of school examples. The user can change different types of parameters, switch settings, and consequently change the problem. It is not just pure presentation of basic concepts without interaction with the user.

Reference

[1] Krupka, M. *Geometrie pro informatiky.* Učební text, Olomouc, 2008.

[2] Budinský, B. *Analytická a diferenciální geometrie.* Praha, SNTL, 1983, ISBN 04-005-83.

[3] Drdla, J. *Počítačová geometrie 1.* Učební texty UP Olomouc, 2004.

[4] Chalupecký, V. *Počítačová grafika.* Učební text FJFI ČVUT Praha, 2005.

[5] Tišnovský, P. Základní informace o OpenGL. *Seriál Grafická knihovna OpenGL* [online]. Poslední úpravy 1. 7. 2003. [citováno 14.května 2010]. URL: http://www.root.cz/clanky/graficka-knihovna-opengl-1.

[6] Tišnovský, P. Teselace a teselátory. *Seriál OpenGL a nadstavbová knihovna GLU* [online]. Poslední úpravy 7. 12. 2004. [citováno 10.června 2010]. URL: http://www.root.cz/clanky/opengl-a-nadstavbova-knihovna-glu-19.

[7] Skála, K. Win32 API - Úvod. *Seriál Win32 - API* [online]. Poslední úpravy 1. 1. 2006. [citováno 24.května 2010]. Dostupné z http://programujte.com.

[8] Neider, J., Davis, T., Woo, M. *OpenGL Programming Guide: The official guide to learning OpenGL (Third Edition).* Addison-Wesley Publishing Company, Silicon Graphics, Inc., 1999, ISBN 0-201-60458-2.

[9] Turek, M. a kol. *NeHe OpenGL Tutoriály.* Elektronická publikace, 2004. Dostupné z http://nehe.ceske-hry.cz.

[10] *Microsoft Developer Network.* http://www.microsoft.com/msdn.

E. Obsah přiloženého CD

Zde je uveden stručný popis obsahu přiloženého CD, tj. závazné adresářové struktury, důležitých souborů apod.

`bin/`
: Instalátor GeometrieSetup programu a samotný program Geometrie spustitelné přímo z CD. Adresář obsahuje i všechny potřebné knihovny a další soubory pro bezproblémové spuštění programu.

`doc/`
: Dokumentace práce ve formátu PDF, vytvořená dle závazného stylu KI PřF pro diplomové práce, včetně všech příloh, a všechny soubory nutné pro bezproblémové vygenerování PDF souboru dokumentace, tj. zdrojový text dokumentace, vložené obrázky, apod.

`src/`
: Kompletní zdrojové texty programu Geometrie se všemi potřebnými (převzatými) zdrojovými texty, knihovnami a dalšími soubory pro bezproblémové vytvoření spustitelných verzí programu.

`readme.txt`
: Instrukce pro instalaci a spuštění programu Geometrie, včetně požadavků pro jeho provoz.

Navíc CD obsahuje:

`ogl/`
: Potřebné knihovny a hlavičkové soubory pro správnou instalaci OpenGL/GLUTu (OpenGL Utility Toolkitu) a nastavení vývojového prostředí, aby byla možná případná kompilace programu.