Vysoká škola strojní a textilní v Liberci
Fakulta strojní

DIPLOMOVÁ PRÁCE

1994

Martin Spěšný

VYSOKÁ ŠKOLA STROJNÍ A TEXTILNÍ V LIBERCI

Fakulta strojní

Katedra technické kybernetiky. Školní rok: 1993/94

ZADÁNÍ DIPLOMOVÉ PRÁCE

pro Martina SPĚŠNÉHO

obor 23-40-8 Automatické systémy řízení výrobních procesů ve strojírenství

Vedoucí katedry Vám ve smyslu zákona č. 172/1990 Sb. o vysokých školách určuje tuto diplomovou práci:

Název tématu:

3D aplikace pro MicroStation PC 5.0

Zásady pro vypracování:

1. Seznámit se s integrovaným prostředím systému MicroStation PC 5.0
2. Seznámit se s principy vytváření MDE aplikací
3. Vytvořit prostředky pro dynamickou manipulaci s vybranou množinou 3D prvků
4. Zhodnotit dosažené výsledky s ohledem na možné rozšíření a doplnění

Rozsah grafických prací:

Rozsah průvodní zprávy: 35 stran

Seznam odborné litaratury:

Manuály MicroStation PC 4.0. Huntsville, Bentley Systems, Inc. and Intergraph Corporation 1991

1 User's Guide
2 Reference Guide
3 Customization Guide
4 MDL Manual

Vedoucí diplomové práce: Ing. Zdeněk Stránský, CSc.

Konzultant: Ing. Miloslav Šrám

Zadání diplomové práce: 31.10.1993

Termín odevzdání diplomové práce: 27.5.1994

L.S.

Vedoucí katedry

Doc.Ing.Vojtěch Konopa, CSc.

Děkan

Prof.Ing.Jaroslav Exner, CSc

V Liberci dne 29.10. 1993

Vysoká škola strojní a textilní v Liberci

Fakulta strojní

Katedra technické kybernetiky

obor : Automatické systémy řízení výrobních procesů ve strojírenství

3D aplikace pro Microstation PC verze 5.00

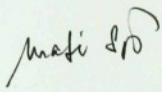
SF - KTK
MARTIN SPĚŠNÝ

Vedoucí práce : Ing. Zdeněk Stránský, CSc
VŠST Liberec - KTK

Konzultant : Ing. Miloslav Šrám
VUZORT, a.s. Praha

Počet stran : 40
Počet obrázků : 5

" *Místopřísežně prohlašuji, že jsem diplomovou práci vypracoval samostatně* s použitím uvedené literatury ".



V Liberci dne 27.5. 1994

Martin Spěšný

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce ing. Zdeňkovi Stránskému, CSc. a mému konzultantovi ing.Miloslavovi Šrámovi za odborné vedení a pomoc při vypracování diplomové práce.

Dále bych rád poděkoval dalším odborníkům a přátelům za poskytnutí potřebných podkladů, rad, informací a podpory během vypracování.

Obsah

1.Úvod

Existuje mnoho oborů lidské činnosti, které se zabývají konstruováním, modelováním popřípadě sestavováním různých 3D objektů pomocí počítače a vhodně zvoleného programu nebo systému. Moje aplikace vznikla jako ohlas na nedostatek aplikačního software, který se zabývá touto problematikou. Jde vlastně o dynamickou manipulaci objektů v prostoru.

Pod pojmem manipulace rozumějme pohyby posuvné a rotační, které mohou být vzájemně provázané, popřípadě dochází pouze k posuvu nebo pouze k rotaci. K vyřešení tohoto problému bylo zvoleno integrované prostředí Microstation verze 5.00, které mi bylo poskytnuto od pražské firmy VUZORT, a.s.

Tento systém patří mezi tzv. CAD systémy (Computer Aided Design), přeloženo jako Počítačem Podporované Projektování nebo raději Automatizace Inženýrských prací. Tyto systémy kladou obrovské nároky nejen na software a hardware, ale i na uživatele.

Jak již jsem předeslal, systém Microstation klade veliké nároky na hardware. Nejdříve byly tyto produkty vytvořeny pro počítače typu Macitosh a až v poslední době došlo k přenesení na u nás nejpoužívanější PC. Protože nemám možnost přístupu na Macintosh nebo na žádnou grafickou stanici vytvořil jsem aplikaci pro PC.

V prvé řadě jsem se musel seznámit s integrovaným prostředím Microstation. (se základními funkcemi, zakládáním nových výkresů, kreslením a používáním některých vymožeností tohoto systému). Teprve potom jsem mohl začít vytvářet program pomocí vestavěného jazyka MDL (Microsoft Development Language).

Výsledným produktem této práce je aplikační program vytvořený v prostředí MDL jazyka ukazující základní zásady při programování CAD systému. Tento aplikační program může být nápomocen v mnoha oblastech konstruování a modelování. Programu lzevyužít i v archeologii (pro kterou byl vlastně tento program vytvořen) při sestavování, vytváření, manipulaci a animaci různých nálezů (střepů či lidských kostí), které jsou přeneseny do CAD systému pravděpodobně pomocí snímací kamery a chovají se zde jako 3D objekty.

S těmito objekty je možno pracovat jako s drátovými modely, tak pro lepší představivost i s vystínovanými 3D prvky. Ovšem pro tento účel je nutno použít poněkud výkonnějších počítačů nebo grafických stanic.

2. CAD systémy

Všechny CAD systémy mají některé společné rysy programového a technického vybavení. Je to především interaktivní ovládání pomocí programového jazyka, menu, ikon a dále využívaných vstupních zařízení (myši, světelného pera, tabletu a popřípadě i skeneru a digitizeru) a výstupních zařízení (jako jsou tiskárna a plotr).

Všechny CAD aplikace, ať už se jedná o jakoukoliv oblast, mají několik důležitých společných rysů. Mezi hlavní patří standartizace, využití počítačových sítí a jeho hlavní vlastností je otevřenost.

3. Co je Microstation

Microstation je produktem americké firmy Intergraph Corporation. Společnost Intergraph působí v oblasti grafických systémů od sedmdesátých let a patří k těm největším.

Intergraph dodává komplexní řešení úloh z mnoha oblastí. V podvědomí vešla se svými geografickými a městskými informačními systémy tzv. GIS systémy (Geographical Information Systems), pak následovaly oblasti geodézie a kartografie, strojírenství, stavebnictví a architektura, projektování technologických celků, inženýrské sítě, správa technické dokumentace, digitální zpracování publikací, elektronika a další.

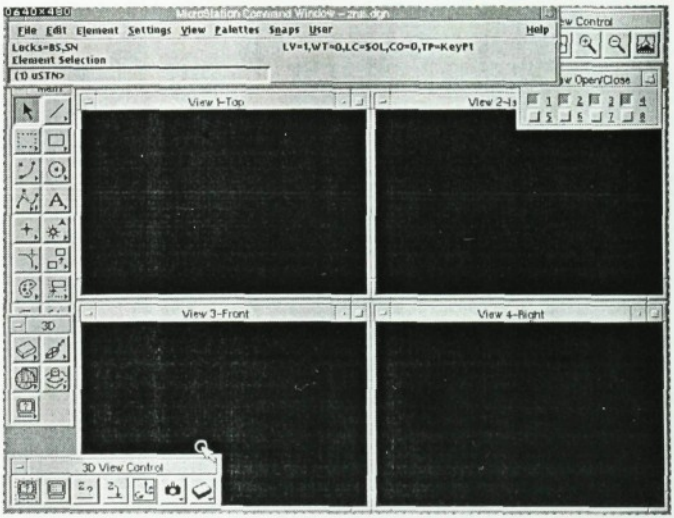
Jde bezesporu o velice progresivní systém, který má dobře vyřešenou spolupráci s databázemi a přehlednou správu grafických a negrafických dat, dokonalou práci ve 2D, 3D a s tím související vizualizace, jednoduchá animace a rendering.

Program Microstation pracuje na několika platformách. Existují verze pro MS-DOS a Windows 3.1 na PC, Windows NT na PC a DECu, UNIX na stanicích Intergraph, Sun, Hewlett Packard, IBM, DEC a Silicon Graphics.

Mezi velké výhody verze 5.00 patří možnost přímo číst a zapisovat DWG soubory, to znamená, že přenos souborů mezi AutoCADem a Microstation probíhá téměř bez ztráty informace. Z rastrových formátů podporuje Microstation import pro Intergraph RGB, TIFF, JPEG, Win BMP a další. Komunikace s ostatními CAD programy probíhá prostřednictvím vektorových formátů - DWG, IGES, CGM a DXF.

3.1. Uživatelské prostředí

Program používá standardní uživatelské prostředí OSF/Motif, které obsahuje roletová menu, dialogová okna, palety nástrojů a přesouvatelná okna s nastavitelnou velikostí. Velkou výhodou je také prohlížení výstupu na plotr. Výkres je



možné předběžně prohlížet v tom tvaru, v jakém se vykreslí na plotru. Nápověda je hypertextová, její okno může být neustále otevřené. Jakmile je aktivována nějaká funkce, automaticky je v okně vypsána nápověda k této funkci. Soubor s nápovědou lze prohledávat i na výskyt klíčových slov.

3.2. Paměť

Výrazné urychlení práce poskytuje výkresová cache, která je novinkou této verze. Pro urychlení přístupu k prvkům může být celý výkres nebo jeho část uložena v operační paměti. Změny ve výkresu jsou okamžitě ukládány na pevný disk.

3.3. Vizualizace

Jedním z důležitých výstupních funkcí Microstation je vizualizace. Protože je vizualizace úzce spjata s touto prací z hlediska výsledného efektu a konečné (vizuální) podoby modelů, chtěl bych vyzdvihnout některé možnosti a přednosti systému. Vizualizace je vlastně změna podoby 3D objektů z inženýrských schémat a řezů do srozumitelnější fotorealisticky vystínované podoby.

Microstation nám nabízí několik funkcí, pomocí kterých můžeme docílit velice výrazného efektu.

- program nabízí 7 metod renderingu
- redukovat schodovitost stínovaných obrázků
- možnost volení jednotlivého druhu osvětlení (sluneční, bodové, záblesk, prostorové rozptýlené, vzdálené a reflektor)
- pro zvýšení efektu lze rozostřením pozadí simulovat vzdálenost

- na tělesa je možné mapovat rastrové struktury
- simulace průhlednosti a průsvitnosti těles
- jednoduché animované scény je možné pořizovat, nahrávat a stříhat přímo v prostředí Microstation
- lze měnit perspektivu pohledu pomocí instalace kamery a jejích výměnných objektivů

Pokud nám nestačí fotorealistické ztvárnění našeho modelu je možné vytvořit animaci, např. letu kolem prostorového objektu nebo jako je vytvořeno v mé aplikaci, animace posuvu a rotace konkrétním elementem s možností volby dopředného nebo zpětného pohybu.

4. Programování v Microstation

Jednou z nejdokonalejších cest, jak si můžeme přizpůsobit Microstation pro vlastní potřebu, je použít aplikační software, který využívá a dále zlepšuje schopnosti systému.

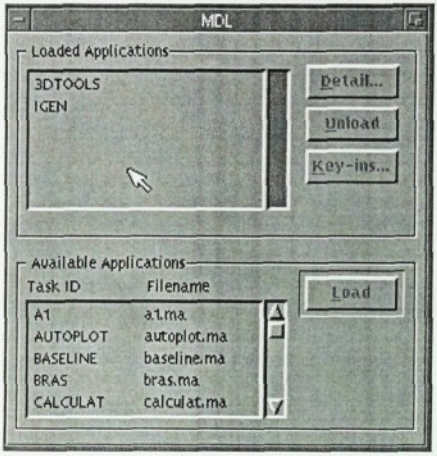
V této verzi je možno programovat těmito způsoby :

- pomocí jazyka uživatelských příkazů (Users commands - UCM) je možné využívat přímo příkazů vestavěných v Microstation nebo vytvářet makra, které jsou zapisovány do souborů s příponou *.UCM. UCM je určen pro tvorbu krátkých programů pro často se opakující posloupnosti příkazů.
- pomocí Microstation Customer Support Library (Micro CSL). Je to vlastně knihovna podprogramů (ve tvaru modulů - object) pro jazyk C a FORTRAN. Aplikace jsou spouštěny z Microstation jako samostatné DOS programy.
- pomocí vývojového jazyka Microstation tzv. MDL (Microstation development language)

5. Programování v MDL

MDL je jazyk C prováděný přímo v grafickém prostředí Microstation. MDL aplikace se také spouští přímo v Microstation, proto se jeví, jako by byly přímou součástí tohoto systému. Mnoho funkcí vestavěných přímo v prostředí je ve skutečnosti psáno jako MDL aplikace, jsou to např. funkce pro B-spline křivky (SPLINES), funkce pro spoje multičar (CUTTER) nebo okno nastavení geometrie (GEOMTOL).

Pro názornost bych rád ukázal, jak vypadá dialogové okno MDL aplikací s právě aktivními aplikacemi v horní části okna a s vybíranými aplikacemi v dolní části..



Protože základ MDL aplikací je v jazyce C, mají mnoho programových možností, jako klasické programovací jazyky. Mohou například definovat vlastní proměnné, používat struktury, uniony a ukazatele a také pracovat s funkcemi. Mezi některé nevýhody patří, že překlad zdrojového textu aplikace není převeden přímo do strojového kódu, ale do kódu, který je pak přímo interpretován v Microstation.

MDL je navržen ke zhotovení aplikací, které půjdou snadno přenášet i na jiné platformy Microstation (PC, Intergraph UNIX workstation, Mac, SPARC, HP a další). MDL také nabízí uživateli rozhraní, ve kterém může být uspořádání aplikace přeloženo i pro neanglicky mluvící trhy.

MDL aplikace mohou zpřístupnit dynamickou paměť přidělením přes správu paměti standardního C jazyka jako jsou např. funkce malloc nebo alloc a v mé aplikaci používaný realloc. Datové ukazatele jsou skutečné ukazatele na datový prostor Microstation.

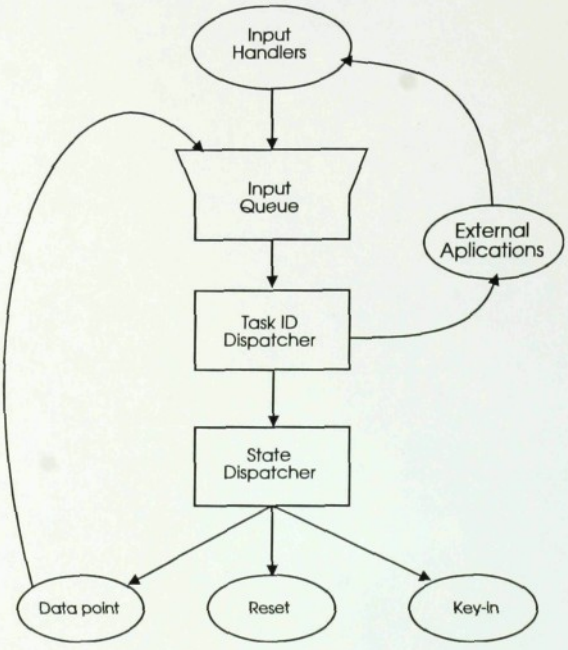
Velice propracované je vytváření dialogových oken. K tomuto účelu je v této verzi vytvořeno prostředí MDE (Microstation Development Environment), které umožňuje definovat několik typů standardních položek. Pomocí MDE prostředí je také vytvořeno dialogové okno pro tento program. Je to prostředí velice podobné klasickému prostředí Microstation. Zde máme možnost pomocí lehkého ovládání a grafické nabídky položek vytvořit jakékoliv dialogové okno včetně rozměrů. Pro potřeby této diplomové práce bylo vytvořeno dialogové okno s těmito položkami :

- ***Group Box Item*** - obdélník ohraničující položky, které k sobě tématicky náleží
- ***Label Item*** - používá se k zobrazení textového řetězce
- ***Text Item*** - slouží k zobrazení a zadávání vstupního textového nebo číselného řetězce
- ***Toggle Button Item*** - je to položka ukazující stav aplikační proměnné, nabývající hodnot vypnuto/zapnuto.
- ***Push Button Item*** - tlačítko, které při stlačení aktivuje nějakou předdefinovanou funkci.

MDL aplikace neprobíhají jako klasický program, ve smyslu nepřerušeného běhu programu, ale jsou založeny na technikách událostmi řízených programů. Tyto programy nejsou přerušeny při čekání na určitý vstup, ale probíhají dále a až při specifikém vstupu jako jsou např. stisk PUSH BUTTON tlačítka, KEY-IN tlačítka nebo DATAPOINT reagují. Řízení těchto událostí zajišťuje dispečer událostí.

Jádrem cyklu Microstation je vstupní fronta (Input Queue). Všechny položky, které čekají ve vstupní frontě jsou kontrolovány a obsluhovány pomocí ovladačů vstupů (Input Handlers). Mezi tyto ovladače patří například ovladač myši, klávesnice atd.

Pro ilustraci koncepce událostmi řízených aplikací ukazuje tento diagram zobrazující smyčku používanou v Microstation.



Událost, která se ocitla v čele fronty, je zpracována dispečerem ID úloh (Task ID Dispatcher). Dispečer zajistí ošetření odpovídajících aplikací (External Applications).

Jestliže neexistuje žádná aplikace pro zpracování události, nastane standardní zpracování dispečerem stavů (State Dipatcher), který převezme řízení.

6. Práce ve 3D

Vzhledem k tomu, že mým úkolem v této diplomové práci bylo umožnit jednoduchou manipulaci s 3D objekty, rád bych nejprve přiblížil prostředí k tomuto účelu potřebné.

6.1. Pracovní prostor ve 3D

Pracovní krychle je 3D variace pracovní roviny. Uživatel má k dispozici přes 4 109 základních jednotek ve směru os X, Y a Z. Souřadnice v pracovní krychli jsou vyjádřeny ve tvaru (x, y, z).

6.2. Prvky 3D

V prostředí Microstation se ve 3D pracuje převážně se složenými prvky. Jediný primitivní 3D prvek je kužel. Každý 3D objekt obsahuje atribut, který definuje, zda se jedná o těleso nebo plochu. Ve 3D lze pracovat se třemi základními typy složených prvků :

- plochy (SURFACE)
- tělesa (SOLID)
- B-spline plochy (B-SPLINE SURFACE)

Těleso můžeme vytvořit buď translací nebo rotací rovinného prvku. Za rovinný prvek můžeme použít cokoliv, s výjimkou textu. Provést translaci a rotaci lze také s řetězcem prvků a uzavřeným řetezcem. Někdy jsou při práci ve 3D potřebné funkce umožňující :

- povrch těles vypočíst a ne jen aproximovat

- určit některé potřebné parametry jako jsou povrch, objem, těžiště, moment setrvačnosti, na některých tělesech můžeme změřit i vzdálenosti
- uživateli umístit do výkresu ne pouze jeden souřadný systém.

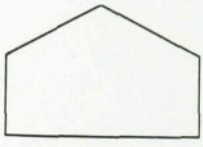
6.3. Pomocný souřadnicový systém (Auxiliary Coordinate System)

ACS je orientováný souřadný systém, který se liší od standardního tím, že je možno umístit ho kamkoliv ve výkresu. Jeho hlavní přednosti se projeví především při práci ve 3D. Při používání ACS je možné zobrazit souřadnicový kříž, ale i přímo si ho definovat. Ve 3D výkresu je možné si definovat tři různé typy ACS :

- pravoúhlý (kartézský). Zde jsou souřadnice vyjádřeny ve tvaru (x,y,z).
- válcový (cylindrický). Body v prostoru jsou vyjádřeny pomocí R, Z a úhlu α. Ve 2D se jedná o tzv. polární souřadnicový systém.
- kulový (sférický). Body v prostoru jsou vyjádřeny pomocí průvodiče R a dvou úhlů α a β.

6.4. Promítání

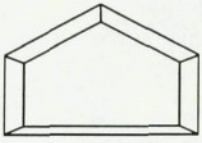
Aby bylo možné zobrazovat na 2D obrazovce 3D prvky, používá Microstation promítání. V systému Microstation lze používat tři druhy promítání - rovnoběžné, perspektivní a izometrické.



obr. 1

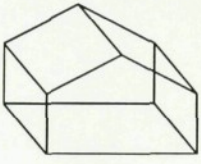
- v pohledu s rovnoběžným promítáním je každý prvek promítán na obrazovku rovnoběžně s osou **Z** pohledu. Toto promítání je z hlediska práce s pohledem

pochopitelnější a lehčí, ale není příliš realistické, protože prvky jsou stejně veliké nezávisle na tom, jak jsou daleko (myšleno do hloubky). *obr.1*



obr.2

- v pohledu perspektivním je každý prvek zobrazován tak, aby se jeho vzdálenější část zdála menší. Takovému pohledu se říká pohled s kamerou. *obr.2*

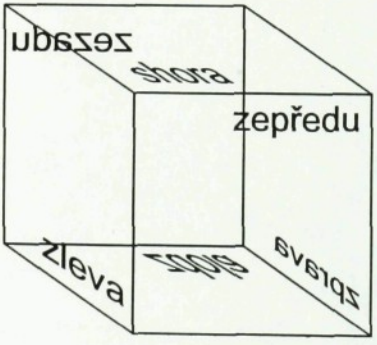


obr.3

- poslední pohled je izometrický, *obr.3*

6.5. Standardní pohledy

Prostředí 3D, protože pohled lze otočit podél tří os, má celkem šest navzájem na sebe kolmých či rovnoběžných orientací . Všechny tyto orientace je možné zvolit přímo v prostředí Microstation.



- shora (top)
- zdola (bottom)
- zleva (left)
- zprava (right)
- zepředu (front)
- zezadu (back)

Microstation má sedm standardních pohledů, jde o výše zmíněné pravoúhlé a Izo (izometrický) pohled. Izo pohled je orientován tak, aby všechny osy pracovní krychle svíraly s rovinou obrazovky stejný úhel.

6.6. Stínování pohledů (Rendering)

Všechny zakládací výkresy dodávané v systému Microstation, jsou v nejjednodušší podobě a to jako drátový model. V pohledu s drátovým modelem jsou plochy zobrazovány pomocí jejich hran, proto jsou prvky za nimi viditelné. Drátový model má tu výhodu, že zobrazuje a dovoluje manipulovat s prvky, které by jinak nebyly viditelné. Na druhou stranu je tato metoda nevýhodná, protože ve složitějších výkresech je už obtížná orientace. Jestliže chceme získat realistický náhled na výkres, můžeme použít osvětlení a rendering .

Jak už jsem se zmínil dříve, Microstation má sedm metod renderingu. Rád bych popsal jejich rozdíly. Jsou seřazeny vzestupně podle hlediska reálnosti.

Drátová síť - je podobně jako drátový model průhledná. V drátové síti jsou však zakřivené plochy reprezentovány polygonální sítí, která je daleko hustší než v obyčejném drátovém modelu.

Řez - zobrazí rovinný řez modelu. Pokud je aktivní nějaký ACS, řez je proveden rovinou. Pokud žádný ACS aktivní není, řez se provede rovinou rovnoběžnou s pohledem v aktivní hloubce.

Skryté hrany - v polygonovém zobrazení jsou viditelné pouze ty části prvků, které nejsou překryty jiným prvkem.

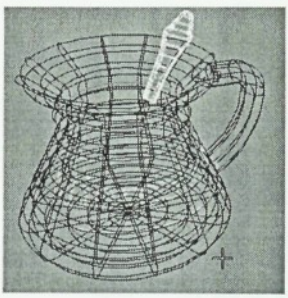
Vyplnění viditelných ploch - je tožné jako polygonové zobrazení, ale každá plocha je vybarvena barvou prvku.

Konstantní stínování - každý prvek je vykreslen jako jeden nebo více polygonů, které jsou vyplněny jednou konstantní barvou, podle charakteristiky materiálu a rozestavění světel.

Hladké stínování - barva polygonu není konstantní, ale je vypočítána na všech hranách polygonu zvlášť a uvnitř postupně přechází jedna v druhou. To vytváří dojem hladké plochy.

Phongovo stínování - zde počítána barva každého pixelu obrazovky. Phongovo stínování vytváří obrázky se skutečně velmi vysokou kvalitou. Je to jediné stínování, které přesně určuje polohu a rozložení světelných paprsků. Jeho překreslování je však velmi pomalé.

Pro srovnání bych rád uvedl nejjednodušší a nejsložitější rendering. První je drátová síť *(obr.1)* a na druhém obrázku je vidět Phongovo stínování *(obr.2)*.



obr.1

obr.2

Každý pohled má režim zobrazení, který určuje, zda a případně jak se má pohled renderovat po obnově nebo po změně objektu. Mezi režimy renderingu zobrazení má nejkratší dobu obnovy drátová síť.

6.7. Osvětlení

S vizualizací úzce souvisí také nastavení světel a charakteristiky materiálu. Plochy ve stínovaných pohledech jsou vidět proto, že je Microstation osvětluje. Prostředí Microstation používá šest druhů světel.

- blesk (světlo, které vychází z místa umístění kamery-podobně jako blesk u fotoaparátu)
- okolní světlo (osvětluje všechny body na ploše rovnoměrně)
- sluneční světlo (přináší dojem osvětlení modelu sluncem). Pro zvýšení přesnosti lze použít nastavení dne v roce, času a zeměpisné šířky a délky.
- bodové světlo (svítí z určitého bodu všemi směry, light point)
- vzdálené světlo (simuluje nekonečně daleko umístěný světelný zdroj)
- reflektor (simuluje kuželovitý svazek paprsků)

Tato světla doplňují okolní světlo a umožňují nastavit různé varianty osvětlených ploch rozmístěných v prostoru.

6.8. Materiálové charakteristiky

Materiálové charakteristiky mají vliv na to, jak se plocha stínuje. Atributy charakteristiky materiálu jsou tyto :

- barva
- odrazivost
- hladkost

Každý prvek se stejným číslem barvy a ve stejné vrstvě je ve stínovaných

pohledech zobrazován se stejnou charakteristikou materiálu. Charakteristiky materiálu jsou uloženy v ASCII souborech, které se nazývají materiálové tabulky. Na tyto soubory se Microstation odkazuje vždy při stínování.

7. Uvedení MDL aplikace

Původním úmyslem bylo vytvořit aplikaci, která bude určena pro archeologické účely. Tato aplikace umožňuje libovolné posouvání, rotaci a jednoduchou animaci 3D objektů.

Aplikace měla hlavně napomáhat při sestavování archeologických vykopávek, které je nutno zkusmo skládat a to není dost dobře možné nebo ne tak pohodlné jako za pomoci tohoto produktu. Postupně jsme zjistili, že najde využití i v jiných oblastech, ať už ve strojírenství nebo stavitelsví.

7.1. Konkrétní řešení v systému Microstation

Představme si, že máme několik objektů nebo řekněme střepů ve výkresu a chceme je poskládat dohromady v jeden celek pomocí posouvání a natáčení.

Nejdříve musíme vybrat element, se kterým budeme manipulovat. V našem případě se jedná po spuštění aplikace a nastavení příslušných parametrů v dialogovém okně (viz. další kapitola) pouze o kliknutí na elementu, ke kterému jsme vyzváni v hlavním menu.

Po identifikaci objektu jsme v pozici, kdy si můžeme rozmyslet jakým způsobem dovedeme element do konečné pozice. V případě, že je vše připraveno, klikneme kdekoliv na pracovní ploše, což nám zaktivuje vlastní manipulaci elementem a můžeme pracovat.

Jestliže je celý objekt složen, může být pomocí standardních funkcí Microstation renderován popřípadě ke zlepšení vizuální stránky i osvícen.

7.2. Řešení pomocí MDL aplikace

Spuštění MDL aplikace může probíhat několika způsoby :

- pomocí příkazové řádky a klávesnice příkazem *MDL load rotace*
- zavoláním přímo z dialogového okna MDL aplikací (MDL applications). Zobrazí se nám dialogové okno; vybereme příslušnou aplikaci a potvrdíme tlačítkem "Load".

7.3. Popis programu a některých použitých funkcí

Po otevření dialogového okna je nutno zadat vstupní údaje, které jsou popsány spolu s ovládáním dialogového okna podrobněji v další kapitole. Nyní bych se rád věnoval samotnému programu.

Celá aplikace začíná identifikací daného elementu. Toto nám umožňuje funkce *mdlLocate_init ()*, která změní kurzor v identifikační kříž a umožní pohodlné vybrání elementu z jakéhokoliv pohledu. V případě nepřesnosti je zajištěno opakované volání identifikace pomocí mnou vytvořené funkce *Znovu*. Jestliže proběhla identifikace v pořádku, musíme ještě jednou potvrdit k přesnému načtení do paměti. Nyní jsme se dostali do smyčky, která nám umožňuje pohodlné ovládání manipulace. Je zajištěna funkcí *Trigger_točení* obsahující pouze počítadlo, které nabývá hodnot 0, 1 podle fáze stisknutí tlačítka na myši. Je to vlastně přerušovač při točení nebo při posouvání prvkem.

Další problém nastal, jestliže došlo v dialogovém okně k přepnutí tlačítka *Použít těžiště* na zapnuto. Při spuštění se automaticky volá vytvořená funkce *Transformace_profilu*, kde dochází k výpočtu těžiště. Protože v systému Microstation se pracuje buď s tělesy, povrchy nebo se spline plochami, pro každou

možnost se těžiště musí určit jiným způsobem. Pro tělesa a povrchy je možno použít vnitřní funkce Microstation. Jsou to *mdlMeasure_surfaceProperties ()* pro povrchy a *mdlMeasure_volumeProperties ()* pro tělesa. Těchto funkcí mimo výpočtu těžiště je možno použít při zjišťování ploch, silových momentů a momentů setrvačnosti. Ovšem při určování těžiště u spline plochy není k dispozici žádná vnitřní funkce, respektive by měla určit těžiště spline plochy funkce *mdlMeasure_surfaceProperties()* jako u povrchů, ale bohužel to nefungovalo a tak jsem byl nucen na úkor přesnosti použít čistě matematickou metodu. Zjistil jsem nejvyšší a nejnižší souřadnici dané plochy, potom rozměry a nakonec jsem určil přibližný střed.

Po vypočtení těžiště záleží na tom, s jakým elementem pracujeme. Při práci s *Celkem* dojde k volání vytvořené funkce *Točení_celku()* a případě práce s *Úlomkem* dojde k volání funkce *Točení_el()*. Tyto funkce se v podstatě od sebe neliší. V obou dochází nejdříve k transformaci souřadnic pomocí sledu následujících funkcí. *MdlCurrTrans_begin()* provádí inicializaci a nastartování transformace. *MdlCurrTrans_rotateByView()* zajišťuje nastavení souřadnic jako jsou v TOP pohledu, tzn. *x*-horizontální, *y*-vertikální a *z*-kolmá na plochu obrazovky. Touto transformací zajistíme dvě důležité věci. Při každém najetí kurzorem myši do jakéhokoliv pohledu vždy dojde k transformaci souřadnic a vzhledem k tomu, že je z-souřadnice kolmá k ploše obrazovky, bude výpočet úhlu natočení jednodušší.

Ve funkci *Točeni_el()* také dochází k vyhodnocování stavu přepínače *Metoda_pohybu* dialogového okna. Při přepnutí do polohy *Zapnuto* jsme nastaveni na metodu postupného narážení na element a postupného natáčení až do žádané polohy. Tato metoda je zajištěna vnitřní funkcí *mdlClip_isElemInside()*, která pracuje s oknem *fence* vytvořeného další vnitřní funkcí *mdlFence-_fromShape()*. Okno je vytvořené ze standardního elementu, v našem případě

je to klasický obdélník. Jeho velikost můžeme nastavit pomocí textové položky v dialogovém okně *Pásmo citlivosti.* Jestliže dojde k překrytí námi vytvořeného okna s daným elementem dochází k transformaci, v opačném případě nikoliv. V tomto místě došlo asi k největším problémům. Nemohl jsem zjistit z jakého důvodu dochází k protínání okna s elementem a následné transformaci pouze v TOP pohledu. V ostatních pohledech se aplikace chová jako kdyby byl element posunut do neznámé a hlavně neviditelné polohy.

Další část, kterou se zabývá funkce *Točení_el* je výpočet úhlu natočení, transformací translačních matic, potřebných k rotaci a posuvu elementu, a vlastního překreslování prvku. K tomuto účelu byly použity tyto funkce. Transformace matic byla provede za pomoci funkcí *mdlTMatrix_rotateByAngles()* a *mdlTMatrix_translate(),* k transformování a kreslení elementu *mdlElmdscr-_transform()* a *mdlElmdscr_display().*

V závěru funkce *Točení_el* ošetřuje přepínače dialogového okna. Jde o tlačítka *Fixace posuvu, Fixace rotace* a *přepínač Zapamatovat,* který slouží k nastavení pomocných proměnných pro zaznamenání transformace elementu pro pozdější animaci. Tuto animaci zajišťují vytvořené funkce *animace_tam* pro dopřednou animaci a *animace_zpět* pro zpětnou animaci.

Poslední důležitou funkcí je *exit_ven,* která zajišťuje regulerní opuštění systému pomocí vnitřní funkce *mdlSystem_exit().*

7.4. Popis práce s dialogovým oknem

V následujícím popisu dialogového okna bych rád vyvětlil všechny jeho funkce.

Po spuštění aplikace se nám otevře pro tento program vytvořené dialogové okno dle obrázku, ve kterém jsou tyto vstupní údaje :

- Tlačítka (PUSH BUTTONS) - Celek, Úlomek, Uvolni celek, Uvolni úlomek, Animace tam, Animace zpět, Přilep
- Přepínače (TOGGLE BUTTONS)- Fixace posuvu, Fixace rotace, Metoda pohybu, Použít těžiště, Zapamatovat, Trasovat, Přidat.
- Textové položky - Pásmo citlivosti, Tolerance, 2x textové hlášky.

Celé dialogové okno je rozděleno na čtyři logické části. První část s názvem "Výběr elementu", druhá s názvem "Nastavení manipulace" třetí "animační nastavení" a samostatná čtvrtá pouze s jedním tlačítkem "Přilep".

V první části (nahoře) se nacházejí čtyři tlačítka (PUSHBUTTON) a dvě textová hlášení. Při stisku horního tlačítka "Celek" nebo spodního tlačítka "Úlomek" se vlastně spouští proces identifikace elementu a tím i celá aplikace. Tlačítko *Celek* umožňuje

výběr elementu již složeného a naopak tlačítko *Úlomek* vybírá pouze jednoduchý element. Dvě tlačítka umístěná vpravo s názvem *Uvolni_celek a Uvolni_úlomek*, uvolní ze smyčky buď Celek nebo Úlomek. Umožňuje změnu manipulačního elementu a opětným stiskem datového tlačítka můžeme opět vybrat další objekt. Textové hlášky nám jen naznačují, jestli máme element vybrán nebo ne.

V druhém logickém okně (uprostřed) s názvem "Nastavení manipulace" se nacházejí tři dvoupolohová tlačítka (TOGGLEBUTTON - dále jen přepínače), která mají polohu vypnuto/zapnuto a dvě textové položky. *Pásmo citlivosti* a *Tolerance*.

Pásmo citlivosti udává rozmezí v jakém bude element reagovat na dotek kurzoru neboli jak velké se má vytvořit dotykové okno *Fence*, pomocí kterého zjišťuje průnik kurzoru s posouvaným elementem. Implicitní hodnota je 0,01, ale je velice důležité v jakém rozměru se pohybujeme.

Tolerance udává toleranci při výpočtu těžiště. Vždy je udána v základních jednotkách. Implicitní hodnota je 1,00 . Zde stejně jako v předchozím případě u *Pásma citlivosti* je velice důležitý rozměr objektů. Jestliže je špatně nastavená tolerance vzhledem k rozměrům, může dojít buď k nepřiměřeně dlouhému výpočtu nebo naopak k vypočtení nesprávné polohy těžiště.

Zbývající čtyři přepínací tlačítka "Fixace posuvu", "Fixace rotace", "Použít těžiště" a "Metoda posunu" slouží k přepínání základního nastavení manipulace. *Fixace posuvu* má implicitní nastavení vypnuto. Pokud přepneme do polohy zapnuto dojde v programu k zafixování translační matice posuvu elementu a v tom případě dochází pouze k rotaci. Obdobně je to s přepínačem *Fixace rotace*. Implicitní hodnota je vypnuto.

Tlačítko *Metoda posuvu* slouží k výběru, jakým způsobem bude docházet k manipulaci. Implicitně je nastavena manipulace bez doteku kurzoru. Pokud je přepnut na zapnuto a dojde ke stisku datového tlačítka myši (zapnutí manipulace), musíme narážet do objektu a teprve potom dochází k posuvu a rotaci. Jestliže je přepínač v poloze zapnuto a dojde ke stisku tlačítka myši (zapnutí manipulace), objekt se pohybuje aniž bychom do něho museli narážet.

Poslední tlačítko v tomto boxu je *Použít těžiště*. Implicitní nastavení je vypnuto tzn. že výpočet těžiště neprobíhá a program bude potřebovat o jeden bod více, Tento bod musíme zadat pomocí datového tlačítka myši a on pak určuje střed otáčení, popřípadě posuvu.

Třetí logická oblast obsahuje tři přepínače a dvě tlačítka. První přepínač s názvem *Zapamatovat* umožňuje výběr mezi zaznamenáním elementu pro pozdější animaci nebo nikoliv. Implicitní nastavení je vypnuto. Jestliže dojde ke změně na zapnuto, veškeré pohyby, které budou uskutečněny s elementem, budou zaznamenány do paměti. V každém okamžiku se zaznamená pohled, ve kterém dochází k transformaci, poloha transformační matice a zvětší se počítadlo jednotlivých elementů.

Jakým způsobem dojde k použití zaznamenaných elementů si můžeme vybrat ze zbývajících dvou přepínačů. První přepínač s názvem "Trasovat" nám dává na výběr jestli chceme jednotlivé elementy při animaci zobrazovat a potom mazat, ovšem bez přidání do výkresu, nebo ne. Implicitní hodnota je vypnuto a znamená, že k trasování nedojde.

Poslední přepínač má název *Přidat* a nabízí nám možnost přidávat nebo nepřidávat všechny zaznamenané polohy elementu do výkresu. Implicitní hodnota je nepřidávat. Zapneme-li toto tlačítko, dojde při animaci k přidávání

všech zaznamenaných poloh elementů do výkresu. Této funkce je možné použít, jestliže potřebujeme zjistit jakým způsobem dochází k transformaci během celého pohybu nebo určit rozdíly za sebou následujících poloh elementů a pomocí renderingu celého objektu, který se vytvoří, můžeme navíc určit nasvícení elementu v jednotlivých polohách, popřípadě zvolit nejlepší polohu pro umístění světla nebo více světel.

Poslední dva přepínače úzce souvisí se dvěmi napravo umístěnými tlačítky, umožňujícími jednoduchou animaci. Jejich názvy jsou *Animace tam* a *Animace zpět* . Jak je již z názvu patrné, stiskem tlačítka *Animace tam* dojde k animaci z původní do konečné polohy elmentu a při stisku druhého tlačítka dojde ke zpětnému chodu. Tyto pohyby jsou ovšem podmíněny předchozím zaznamenáním nebo naopak nezaznamenáním jednotlivých poloh elementu.

Poslední oblastí je vlastně samostatně umístěné tlačítko "Přilepit". Jestliže si myslíme, že náš právě transformovaný element je v konečné poloze a stiskneme toto tlačítko, dojde k přidání tohoto objektu k celku skládajícího se již z několika elementů. V tomto stavu máme k dispozici animaci celé transformační dráhy i s reverzním pohybem.

Tento program má tu nevýhodu, že není možné po přilepení několika elementů k celku pomocí tlačítka *Přilepit*, odtrhnout kterýkoliv element, ale je možné odtrhnout pouze ten poslední. Tato nevýhoda je dána prací tohoto systému s elementy. Je možné ji odstranit ukládáním jednotlivých elementů do databáze. Ale vzhledem k tomu, že tato aplikace má sloužit pouze k manipulaci, není tato závada natolik závažná. Nehledě na to, že zavedení databáze do programu nepatří mezi jednoduché operace a zabrala by pravděpodobně více času než samotná aplikace.

8. Popis jednotlivých programů MDL aplikace "rotace"

Aplikace "rotace", která byla vytvořena v této diplomové práci se skládá ze šesti souborů. Tyto soubory bych chtěl nyní zběžně popsat a později se jim věnovat o něco více.

- rotace.h : obsahuje definice globálních proměnných a symbolických konstant, které jsou použity v dialogovém okně i ve zdrojovém textu
- rottxt.h : obsahuje definice návěstí v dialogovém okně
- rotace.r : definuje strukturu dialogového okna s konkrétními deklaracemi všech jejích položek
- rottyp.mp : obsahuje definici globální struktury
- rotace.mc : je to zdrojový text, který obsahuje definice hlavní (main) funkce a deklarace všech ostatních použitých funkcí
- rotace.mke : je to dávkový soubor pro ovládání kompilátoru (bmake) a jeho utilit, potřebných k vytvoření spustitelného programu v Microstation rotace.ma

Pro přehlednost jsem všechny soubory rozdělil poznámkami, na které se obracím při popisu souborů.

8.1. Soubor "rotace.h"

Tento soubor s příponou *.h (header), můžeme rozdělit do třech logických částí :

1. část - zde jsou definovány konstanty, které jsou použity v dialogovém okně pro zdrojové ID čísla. Každá položka v této části souboru rotace.h má svoje ID

číslo, které je jedinečné. Z toho vyplývá, že při definici jednotlivých položek jednoho typu (jako jsou například DItem_TextRsc, DItem_ToggleButtonRsc, DItem_PushButtonRsc) dialogového okna nesmí nastat to , že by dvě stejné položky měli stejné ID číslo.

2.část- obsahuje definici symbolických konstant pro hákové funkce(HOOK FUNCTIONS). Pro tyto konstanty platí stejná pravidla jako v předchozí části, nesmí se vyskytnout konstanty se shodným ID číslem.

3.část- v závěru tohoto souboru je definována celá struktura položek dialogového okna s názvem "globals".

Tento soubor, obsahující definice globální struktury a symbolických konstant je použit v souborech "rotace.r", "rotace.mc" a v souboru "rottyp.mp".

8.2. Soubor "rottxt.h"

V souboru jsou obsaženy definice textů, které jsou umístěny v dialogovém okně. Tento program byl vytvořen pomocí prostředí MDE. Toto prostředí bylo popsáno již dříve. V případě, že bychom chtěli provést nějaké změny, stačí je provést v tomto souboru.

8.3. Soubor "rotace.r"

1.část - obsahuje definici hlavičkových souborů, ve kterých jsou obsaženy struktury dialogového okna.

2.část - obsahuje úplnou definici dialogového okna DIALOGID_Dialog_Box se

všemi parametry včetně rozměrů. Dialogové okno obsahuje celkem 17 položek. Každá položka má dané přesně své rozměry, umístění v dialogovém okně a parametr jestli má být zobrazena (ON) či nikoliv (HIDEN).

3.část - Tato oblast obsahuje definice PUSH BUTTON položek obsažených v dialogovém okně.

Jsou to : PUSHBUTTONID_Přilepit
PUSHBUTTONID_Uvolnit_celek
PUSHBUTTONID_Uvolnit_úlomek
PUSHBUTTONID_Opakovat_tam
PUSHBUTTONID_Opakovat_zpět
PUSHBUTTONID_Celek
PUSHBUTTONID_Úlomek

4.část - obsahuje definice textových položek (struktura DItem_TextRsc). V mém dialogovém okně jsou dva druhy textových položek. V horní části dialogového okna jsou pouze hláškové texty (READ ONLY), které nám hlásí zda jsme v transformační smyčce nebo ne. Ve střední části dialogového okna jsou umístěny textové položky, které udávají toleranci a pásmo citlivosti.

5.část-Tato oblast obsahuje definice TOGGLE BUTTON položek obsažených v dialogovém okně. Tyto přepínače nabývají hodnot pouze (-1-zapnuto, 0-vypnuto)

Jsou to : TOGGLEID_Fixace_posuvu
TOGGLEID_Fixace_rotace
TOGGLEID_Použít_těžiště
TOGGLEID_Zapamatovat
TOGGLEID_Trasování
TOGGLEID_Přidávat
TOGGLEID_Metoda_pohybu

8.4. Soubor "rottyp.mt"

V tomto souboru jsou definovány použité hlavičkové soubory a definuje strukturu *globals_tag*, která obsahuje hodnoty proměnných, které se zadávají v dialogovém okně.

8.5. Soubor "rotace.mc"

Tento soubor je vlastně hlavní program (zdrojový), který obsahuje základní funkci main. Dále obsahuje ostatní funkce, které obsluhují vlastní manipulaci. Je rozdělen na 9 částí.

1.část - definuje hlavičkové soubory, globální funkce a proměnné a počáteční naplnění struktury globals.

2.část - definuje vztahy hákových funkcí

3. část - obsahuje hlavní funkci main, ve které je zveřejnění globálních struktur, proměnných, hákových funkcí a zde se také otevírá dialogové okno DIALOGID_Dialog_Box.

4.část - definice hákových funkcí, obsahuje položky *create, init* a *destroy*

5.část - obsahuje definice funkcí na identifikaci elementu, jeho kontrolu a v případě neúspěšné identifikace funkci na znovuspuštění identifikace.

Jsou to funkce : Ident_Elementu ( )
Kontrola ( )
Znovu ( )

6. část - zde jsou definovány vlastní funkce, které umožňují manipulaci, otáčení a posuv libovolným elementem, určení jeho těžiště apod.

Obsahuje tyto funkce : Transformace_profilu ( )

Točeni_el ( )

Transformace_profilu_CD ( )

Točeni_celku ( )

7. část - obsahuje funkce, které umožňují jednoduchou animaci. Tyto funkce jsou závislé na globálních proměnných globals.zapamatovat, globals.trasovat, globals.přidat. Tyto proměnné jsou spojené s položkami TOGGLEID_Zapamatovat, TOGGLEID_Trasovat, TOGGLEID_Přidat, jejichž definice jsou umístěny v souboru rotace.r.

8. část - definice jednoduché funkce, která umožňuje libovolné přepínání skoku do transformační smyčky a opět z ní zase ven.

9. část - definuje funkci, která umožňuje přilepení elementu

8.6. Soubor "rotace.mke"

"Rotace.mke" je prováděcí soubor, který používá utilitu bmake a ostatní utility k automatickému vytvoření aplikačního souboru rotace.ma. Podrobnější popis najdeme v příloze.

9. Závěr

V této diplomové práci jsem měl možnost seznámit se s integrovaným grafickým systémem Microstation verze 5.00. Protože mým úkolem bylo hlavně vytvořit aplikaci v programovacím jazyku MDL, zabýval jsem se především tímto problémem.

Výsledným produktem mé diplomové práce je, jak jsem již řekl, aplikační program ukazující hlavní zásady při programování CAD systému, jakým je Microstation. Tento program může být nápomocen v různých oblastech lidské činnosti. Přestože byl vytvořen hlavně pro archeologické účely, najde pravděpodobně využití i ve strojírenství, stavitelství nebo dokonce v architektuře.

Poukazuje na možnosti, které nabízí tento systém prostřednictvím programovacího jazyka MDL. Protože žádný grafický systém není uzpůsobený pro řešení konkrétních problémů, je možno pomocí MDL vytvořit konkrétní aplikaci potřebnou k řešení daného problému.

Na zadaném konkrétním příkladě manipulace s 3D prvky v prostoru jsem chtěl ukázat základní kroky při tvorbě MDL aplikace. Téměř ke každé aplikaci patří grafické dialogové okno, které bylo v tomto případě vytvořeno pomocí prostředí MDE. Slouží k zadávání vstupních parametrů a nastavení základních voleb nutných pro chod programu.

Přednosti navržené aplikace oproti řešení tohoto problému pomocí základních funkcí v integrovaném prostředí Microstation jsou v pohodlnosti zadávání vstupních údajů, popřípadě jejich změny, ale především v jednoduchosti manipulace s danými objekty.

Aplikace totiž dovoluje pomocí dialogového okna jednoduché přepínání druhu pohybu, ať už se jedná o rotační nebo posuvný. A navíc poskytuje pomocí dopředné a zpětné animace jejich přehrání.

Tento grafický systém může poskytnout mnoho nových možností, jak v kreslení, projektování nebo v programování a může být velkým přínosem při poznávání nových a výkonných CAD systémů.

Je ovšem pravda, že v oblasti programování v MDL je tato situace poněkud složitější. Tento jazyk potřebuje pro své zvládnutí poněkud delší čas, protože se jedná o velice rozsáhlý a složitý systém.

10. Použitá literatura

(1) Bentley Systems, Inc. and Intergraph Corporation, Huntsville : Microstation PC Customization Guide, 1991

(2) Bentley Systems, Inc. and Intergraph Corporation, Huntsville : Microstation PC User's Guide, 1991

(3) Bentley Systems, Inc. and Intergraph Corporation, Huntsville : Microstation PC Reference Guide, 1991

(4) Bentley Systems, Inc. and Intergraph Corporation, Huntsville : Microstation PC MDL manual, 1991

(5) GRANÁ, L. - SECHOVSKÝ, H. : Počítačová grafika. Praha 1980, SNTL

11. Seznam příloh

Příloha č.1 : Výpis zdrojových souborů aplikace rotace v jazyce MDL

Příloha č. 2 : Disketa, obsahující všechny potřebné zdrojové texty a spustitelný soubor "rotace.ma"

```
/*-------------------------------------------------------------+
|                                                             |
| rotace.h - definuje konstanty a typy, které jsou použity v této aplikaci
|                                                             |
+-------------------------------------------------------------*/


/*-------------------------------------------------------------+
|                                                             |
| 1.část - definice zdrojových  ID čísel použitých položek    |
|                                                             |
+-------------------------------------------------------------*/

#define DIALOGID_Dialog_Box                     1

#define HOOKDIALOGID_Dialog_Box                 1

#define PUSHBUTTONID_Prilepit                   1
#define PUSHBUTTONID_Uvolnit_celek              2
#define PUSHBUTTONID_Opakovat_tam               3
#define PUSHBUTTONID_Opakovat_zpet              4
#define PUSHBUTTONID_Celek                      5
#define PUSHBUTTONID_Ulomek                     6
#define PUSHBUTTONID_Uvolnit_ulomek             7


#define TOGGLEID_Fixace_posuvu                  4
#define TOGGLEID_Fixace_rotace                  5
#define TOGGLEID_Pouzit_teziste                 6
#define TOGGLEID_Zapamatovat                    7
#define TOGGLEID_Trasovani                      8
#define TOGGLEID_Pridavat                       9
#define TOGGLEID_Metoda_pohybu                  10


#define TEXTID_Vybran_celek                     7
#define TEXTID_Pasmo_citlivosti                 8
#define TEXTID_Tolerance                        9
#define TEXTID_Vybrán_ulomek                    10


/*-------------------------------------------------------------+
|                                                             |
| 2.část - definice zdrojových ID čísel pro závěsené příkazy dialog box
|                                                             |
+-------------------------------------------------------------*/

#define CMD_identifikace_ulomek                 1
#define CMD_animace_tam                         2
#define CMD_animace_zpet                        3
#define CMD_zmena_teziste                       4
#define CMD_nahravani_zacátek                   5
#define CMD_uvolni_celek                        6
#define CMD_uvolni_ulomek                       7
#define CMD_identifikace_celek                  8
#define CMD_prilepit                            9
```

```
/*-------------------------------------------------------------+
|
| 3.část - definice typu globální struktury
|
+-------------------------------------------------------------*/

typedef struct globals_tag
{
                        double          tolerance;
                        double          q;
                        int             metoda_pohybu;
                        int             trasovat;
                        int             pridávat;
                        int             fixace_posuvu;
                        int             fixace_rotace;
                        int             pouzit_teziste;
                        int             zapamatovat;
                        int             pridat;
                        char            vybrano_c(4);
                        char            vybrano_u(4);
} Globals;
```

```
/*-------------------------------------------------------------------+
|
| rottxt.h - obsahuje definice navěstí v dialogovém okně
|
+-------------------------------------------------------------------*/

#define TXT_dBox0_Rotaceelementu                "Rotace elementu"
#define TXT_gBox0_Nastavenipohybu               "Nastaveni pohybu"
#define TXT_gBox0_Vybranielementu               "Vybrani elementu"
#define TXT_gBox0_Animacepohybu                 "Animace pohybu"
#define TXT_pBtn3_Opakovattam                   "Opakovat tam"
#define TXT_pBtn4_Opakovatzpet                  "Opakovat zpet"
#define TXT_pBtn6_Celek                         "Celek"
#define TXT_pBtn7_Ulomek                        "Ulomek"
#define TXT_pBtn8_Uvolnit                       "Uvolnit"
#define TXT_dTxt7_Toler                         "Toler."
#define TXT_tBtn15_Pridavat                     "Pridavat"
#define TXT_dTxt8_Pasmocitlivosti               "Pasmo citlivosti"
#define TXT_tBtn16_Metodapohybu                 "Metoda pohybu"
#define TXT_pBtn1_Prilepit                      "Prilepit"
#define TXT_pBtn2_Uvolnit                       "Uvolnit"
#define TXT_dTxt9_Vybrano                       "Vybrano:"
#define TXT_dTxt10_Vybrano                      "Vybrano:"
#define TXT_tBtn4_Fixaceposuvu                  "Fixace posuvu"
#define TXT_tBtn5_Fixacerotace                  "Fixace rotace"
#define TXT_tBtn6_Pouzitteziste                 "Pouzit teziste"
#define TXT_tBtn13_Zapamatovat                  "Zapamatovat"
#define TXT_tBtn14_Trasovani                    "Trasovani"
```

```
/*-------------------------------------------------------------------+
|
| rotace.r - obsahuje konstanty a typy, které se používají v této aplikaci
|
+-------------------------------------------------------------------*/


/*-------------------------------------------------------------------+
|
| 1.část - definice hlavičkových souborů
|
+-------------------------------------------------------------------*/

#include <rscdefs.h>
#include <cmdclass.h>
#include <dlogbox.h>
#include <dlogids.h>
#include <keys.h>

#include "rotace.h"
#include "rottxt.h"


/*-------------------------------------------------------------------+
|
| 2.část - definice dialogového okna
|
+-------------------------------------------------------------------*/

DialogBoxRsc DIALOGID_Dialog_Box =
   {
    DIALOGATTR_DEFAULT,
    47 * XC, 25 * YC + 10,
    NOHELP,   MHELP,
    HOOKDIALOGID_Dialog_Box,
    NOPARENTID,
    TXT_dBox0_Rotaceelementu,
{
{{11*XC, 22*YC+15, 25*XC, 0},PushButton, PUSHBUTTONID_Přilepit, ON, 0, TXT_pBtn1_Prilepit, ""}
{{33*XC+4, 1*YC+9, 10*XC, 0},PushButton, PUSHBUTTONID_Uvolnit_celek, ON, 0, TXT_pBtn2_Uvolnit,
""},
{{21*XC+3, 17*YC+6, 22*XC, 0},      PushButton, PUSHBUTTONID_Opakovat_tam, ON, 0,
TXT_pBtn3_Opakovattam, ""},
{{21*XC+3, 19*YC+8, 22*XC, 0},      PushButton, PUSHBUTTONID_Opakovat_zpet, ON, 0,
TXT_pBtn4_Opakovatzpet, ""},
{{3*XC+1, 1*YC+9, 12*XC, 0},        PushButton, PUSHBUTTONID_Celek, ON, 0, TXT_pBtn6_Celek,
""},
{{3*XC+1, 4*YC+4, 12*XC, 0},        PushButton, PUSHBUTTONID_Ulomek, ON, 0,
TXT_pBtn7_Ulomek, ""},
{{33*XC+3, 4*YC+4, 10*XC, 0},       PushButton, PUSHBUTTONID_Uvolnit_ulomek, ON, 0,
TXT_pBtn8_Uvolnit, ""},

{{3*XC+2, 9*YC+2, 0, 0},            ToggleButton, TOGGLEID_Fixace_posuvu, ON, 0,
TXT_tBtn4_Fixaceposuvu, ""},
{{3*XC+2, 11*YC+1, 0, 0},           ToggleButton, TOGGLEID_Fixace_rotace, ON, 0,
TXT_tBtn5_Fixacerotace, ""},
{{27*XC+4, 10*YC+10, 0, 0},         ToggleButton, TOGGLEID_Použít_teziste, ON, 0,
TXT_tBtn6_Pouzitteziste, ""},
{{3*XC+2, 17*YC+6, 0, 0},           ToggleButton, TOGGLEID_Zapamatovat, ON, 0,
TXT_tBtn13_Zapamatovat, ""},
{{3*XC+2, 19*YC, 0, 0},             ToggleButton, TOGGLEID_Trasovani, ON, 0,
```

```
TXT_tBtn14_Trasovani, ""},
{{3*XC+2, 20*YC+7, 0, 0},          ToggleButton, TOGGLEID_Pridavat, ON, 0,
TXT_tBtn15_Pridavat, ""},
{{27*XC+4, 9*YC+1, 0, 0},          ToggleButton, TOGGLEID_Metoda_pohybu, ON, 0,
TXT_tBtn16_Metodapohybu, ""},
{{16*XC+1, 12*YC+11, 9*XC, 0},     Text, TEXTID_Pasmo_citlivosti, ON, 0, TXT_dTxt8_Pasmocitlivosti,
""},
{{23*XC+11, 1*YC+14, 9*XC, 0},     Text, TEXTID_Vybran_celek, ON, 0, TXT_dTxt9_Vybrano, ""},
{{23*XC+11, 4*YC+9, 9*XC, 0},      Text, TEXTID_Vybran_úlomek, ON, 0, TXT_dTxt10_Vybrano, ""},
{{34*XC+1, 12*YC+11, 9*XC, 0},     Text, TEXTID_Tolerance, ON, 0, TXT_dTxt7_Toler, ""},

{{1*XC+3, 8*YC, 44*XC, 7*YC},      GroupBox, 0, ON, 0, TXT_gBox0_Nastavenipohybu, ""},
{{1*XC+3, 9, 44*XC, 6*YC},         GroupBox, 0, ON, 0, TXT_gBox0_Vybranielementu, ""},
{{1*XC+3, 16*YC+5, 44*XC, 6*YC},   GroupBox, 0, ON, 0, TXT_gBox0_Animacepohybu, ""},
}
  };


/*-------------------------------------------------------------------------+
|
| 3.část - zdroje pro definice PUSH BUTTON položek
|
+-------------------------------------------------------------------------*/

DItem_PushButtonRsc PUSHBUTTONID_Prilepit =
  {
  NOT_DEFAULT_BUTTON,
  NOHELP,
  LHELP,
  NOHOOK,
  NOARG,
  CMD_prilepit,
  LCMD,
  "",
  TXT_pBtn1_Prilepit
  };

DItem_PushButtonRsc PUSHBUTTONID_Uvolnit_celek =
  {
  NOT_DEFAULT_BUTTON,
  NOHELP,
  LHELP,
  NOHOOK,
  NOARG,
  CMD_uvolni_celek,
  LCMD,
  "",
  TXT_pBtn2_Uvolnit
  };


DItem_PushButtonRsc PUSHBUTTONID_Opakovat_tam =
  {
  NOT_DEFAULT_BUTTON,
  NOHELP,
  LHELP,
  NOHOOK,
  NOARG,
  CMD_animace_tam,
  LCMD,
  "",
  TXT_pBtn3_Opakovattam
  };
```

```
DItem_PushButtonRsc PUSHBUTTONID_Opakovat_zpet =
   {
   NOT_DEFAULT_BUTTON,
   NOHELP,
   LHELP,
   NOHOOK,
   NOARG,
   CMD_animace_zpet,
   LCMD,
   "",
   TXT_pBtn4_Opakovatzpet
   };


DItem_PushButtonRsc PUSHBUTTONID_Celek =
   {
   NOT_DEFAULT_BUTTON,
   NOHELP,
   LHELP,
   NOHOOK,
   NOARG,
   CMD_identifikace_celek,
   LCMD,
   "",
   TXT_pBtn6_Celek
   };


DItem_PushButtonRsc PUSHBUTTONID_Ulomek =
   {
   NOT_DEFAULT_BUTTON,
   NOHELP,
   LHELP,
   NOHOOK,
   NOARG,
   CMD_identifikace_ulomek,
   LCMD,
   "",
   TXT_pBtn7_Ulomek
   };


DItem_PushButtonRsc PUSHBUTTONID_Uvolnit_ulomek =
   {
   NOT_DEFAULT_BUTTON,
   NOHELP,
   LHELP,
   NOHOOK,
   NOARG,
   CMD_uvolni_ulomek,
   LCMD,
   "",
   TXT_pBtn8_Uvolnit
   };
```

```
/*+-------------------------------------------------------------------------+
  |                                                                         |
  | 4.část - zdroje pro definice zdroju textových položek                   |
  |                                                                         |
  +-------------------------------------------------------------------------+*/

DItem_TextRsc TEXTID_Tolerance =
   {
   NOCMD,
   LCMD,
   NOSYNONYM,
   NOHELP,
   LHELP,
   NOHOOK,
   NOARG,
   20,
   "%f",
   "%f",
    "0.001",
   "",
   NOMASK,
   TEXT_NOCONCAT,
   TXT_dTxt7_Toler,
   "globals.tolerance"
   };


DItem_TextRsc TEXTID_Pasmo_citlivosti =
   {
   NOCMD,
   LCMD,
   NOSYNONYM,
   NOHELP,
   LHELP,
   NOHOOK,
   NOARG,
   20,
   "%f",
   "%f",
    "0.01",
   "",
   NOMASK,
   TEXT_NOCONCAT,
   TXT_dTxt8_Pasmocitlivosti,
  "globals.q"
   };


DItem_TextRsc TEXTID_Vybran_celek =
   {
   NOCMD,
   LCMD,
   NOSYNONYM,
   NOHELP,
   LHELP,
   NOHOOK,
   NOARG,
    3,
   "%s",
   "%s",
    "",
   "",
```

```
    NOMASK,
    TEXT_READONLY,
    TXT_dTxt9_Vybrano,
    "globals.vybrano_c"
    ""
    };


DItem_TextRsc TEXTID_Vybran_ulomek =
    {
    NOCMD,
    LCMD,
    NOSYNONYM,
    NOHELP,
    LHELP,
    NOHOOK,
    NOARG,
     3,
    "%s",
    "%s",
    "",
    "",
    NOMASK,
    TEXT_READONLY,
    TXT_dTxt10_Vybrano,
    "globals.vybrano_u"
    };

/*-------------------------------------------------------------------------+
|
| 5.část - zdroje pro definice TOGGLE BUTTON položek
|
+-------------------------------------------------------------------------*/

DItem_ToggleButtonRsc TOGGLEID_Fixace_posuvu =
    {
    NOCMD,
    LCMD,
    NOSYNONYM,
    NOHELP,
    LHELP,
    NOHOOK,
    NOARG,
    NOMASK,
    NOINVERT,
    TXT_tBtn4_Fixaceposuvu,
    "globals.fixace_posuvu"
    };


DItem_ToggleButtonRsc TOGGLEID_Fixace_rotace =
    {
    NOCMD,
    LCMD,
    NOSYNONYM,
    NOHELP,
    LHELP,
    NOHOOK,
    NOARG,
    NOMASK,
    NOINVERT,
```

```
    TXT_tBtn5_Fixacerotace,
    "globals.fixace_rotace"
    };


DItem_ToggleButtonRsc TOGGLEID_Pouzit_teziste=
    {
    CMD_zmena_teziste,
    LCMD,
    NOSYNONYM,
    NOHELP,
    LHELP,
    NOHOOK,
    NOARG,
    NOMASK,
    NOINVERT,
    TXT_tBtn6_Pouzitteziste,
    "globals.pouzit_teziste"
    };


DItem_ToggleButtonRsc TOGGLEID_Zapamatovat =
    {
    CMD_nahravani_zacatek,
    LCMD,
    NOSYNONYM,
    NOHELP,
    LHELP,
    NOHOOK,
    NOARG,
    NOMASK,
    NOINVERT,
    TXT_tBtn13_Zapamatovat,
    "globals.zapamatovat"
    };


DItem_ToggleButtonRsc TOGGLEID_Trasovani =
    {
    NOCMD,
    LCMD,
    NOSYNONYM,
    NOHELP,
    LHELP,
    NOHOOK,
    NOARG,
    NOMASK,
    NOINVERT,
    TXT_tBtn14_Trasovani,
    "globals.trasovat"
    };


DItem_ToggleButtonRsc TOGGLEID_Pridavat =
    {
    NOCMD,
    LCMD,
    NOSYNONYM,
    NOHELP,
    LHELP,
    NOHOOK,
    NOARG,
```

```
    NOMASK,
    NOINVERT,
    TXT_tBtn15_Pridavat,
    "globals.pridavat"
    };


DItem_ToggleButtonRsc TOGGLEID_Metoda_pohybu =
    {
    NOCMD,
    LCMD,
    NOSYNONYM,
    NOHELP,
    LHELP,
    NOHOOK,
    NOARG,
    NOMASK,
    NOINVERT,
    TXT_tBtn16_Metodapohybu,
    "globals.metoda_pohybu"
    };
```

```
/*-------------------------------------------------------------------+
|
| rottyp.mp - obsahuje funkci pro generování globalní proměnné
|
+-------------------------------------------------------------------*/

#include "a1.h"
#include "a1txt.h"

publishStructures (globals_tag);
```

```
/*-------------------------------------------------------------------+
|
| rotace.mc - obsahuje hlavni funkci main a ostatni potrebne funkce
|
+-------------------------------------------------------------------*/

/*-------------------------------------------------------------------+
|                                                                    |
| 1.cast - definice hlavickovych souboru a globalnich promennych     |
|                                                                    |
+-------------------------------------------------------------------*/

#include        <tcb.h>
#include        <userfnc.h>
#include        <cexpr.h>
#include        <mdl.h>
#include        <dlogitem.h>
#include        <cexpr.h>
#include        <cmdlist.h>
#include        <dlogman.fdf>
#include        <mselems.h>
#include        <msinputq.h>

#include        "rotace.h"

/* definovani "complex dynamics" pro dynamickr vykraslovani elementu */
#define         complexDynamics
#define         MASTER_FILE  0

/* pocatecni naplneni globalni struktury */
Private Globals  globals = {1.0,0.01};  /* tolerance = 1.0 */
Inputq_element iq;

/* definice globalnich promennych */
MSElementDescr      *element_ED = NULL ;
MSElementDescr      *element_CD = NULL ;
ULong               fpos, fposc ;
int                 zadani ;
double              Uhel_oz;
double              Uhel_z ;
Dpoint3d            pt_teziste;
Dpoint3d            bod1,pbod ;
int                 poc_anim ;
int                 toc_to;
nt                  frame=1;
int                 t;
int                 obr_cnt;
MSElementDescr      *startelem_ED ;

typedef struct anim_tag
{
int                 view;
Dpoint3d            pt;
}   animT;

animT               *obr;
DialogBox           *dbP;

/* definice funkci obsazenych v hlavnim progranu */
Private void Transformace_profilu_CD( Dpoint3d *pt) ;
Public void Ident_Elementu();
Private void trigger_toceni( Dpoint3d *pt, int view, int drawMode ) ;
```

```
Private void Znovu1();
Private void exit_ven() ;
Private void Kontrola1() ;
Private void Toceni_celku( Dpoint3d *pt, int view, int drawMode ) ;
Private void Toceni_el( Dpoint3d *pt, int view, int drawMode ) ;
Private void Transformace_profilu( Dpoint3d *pt) ;
Private int test_clipComplexElement(MSElementDescr *inputEdP,void *clipP,int view);
Private void Znovu2(void) ;
Private void Kontrola2(void) ;
Public void Ident_celek(void) ;
Public void prilepit(void) ;


/*---------------------------------------------------------------------+
|                                                                      |
| 2.cast - definice vztahu u hakovych funkci                           |
|                                                                      |
+---------------------------------------------------------------------*/

Private dialog_hook(DialogMessage *dmP);

Private DialogHookInfo uHooks[]=
{
    {HOOKDIALOGID_Dialog_Box,dialog_hook},
};


/*---------------------------------------------------------------------+
|                                                                      |
| 3.cast - definice funkce main                                        |
|                                                                      |
+---------------------------------------------------------------------*/

Public main
(
int   argc,
char *argv[]
)
{
    char                *setP;
   RscFileHandle        rscFileH;
    DialogBox           *dbP;

   mdlResource_openFile (&rscFileH, NULL, 0);

   setP = mdlCExpression_initializeSet (VISIBILITY_DIALOG_BOX, 0, FALSE);

    mdlDialog_publishComplexVariable (setP, "globals_tag",
                                      "globals", &globals);


    mdlDialog_hookPublish (sizeof(uHooks)/sizeof(DialogHookInfo), uHooks);


  /* otevreni dialogoveho okna */
    if ( (dbP = mdlDialog_open (NULL, DIALOGID_Dialog_Box)) == NULL)
          mdlOutput_error( "Nelze otevrit hlavni dialog" );
}
```

```
/*-------------------------------------------------------------------+
|
| 4.cast - definice hakovych funkci
|
+-------------------------------------------------------------------*/

Private void dialog_hook
(
DialogMessage *dmP
)
{
    if(dmP->dialogId!=DIALOGID_Dialog_Box)
            return;
   dmP->msgUnderstood = TRUE;
   switch (dmP->messageType)
    {

    case DIALOG_MESSAGE_CREATE:
            strcpy(globals.vybrano_c,"ne");
            strcpy(globals.vybrano_u,"ne");
            break;

    case DIALOG_MESSAGE_INIT:
            bod1.x=bod1.y=bod1.z=0.0;
            pt_teziste.x=pt_teziste.y=pt_teziste.z=0.0;
            break;

    case DIALOG_MESSAGE_DESTROY:
            exit_ven();
    mdlDialog_cmdNumberQueue(FALSE,CMD_MDL_UNLOAD,mdlSystem_getCurrTaskID(),TRUE);
            break;

    default:
            dmP->msgUnderstood = FALSE;

      break;
    }
}



/*-------------------------------------------------------------------+
|                                                                    |
| 5.cast - definice funkci na inicializaci elementu a jeho kontrolu  |
|                                                                    |
+-------------------------------------------------------------------*/

Public cmdName void Ident_celek(void)
cmdNumber CMD_identifikace_celek
{

    cnt=0;
   mdlLocate_init();


    /*      spusteni procesu identifikace    */
    mdlState_startModifyCommand( Ident_celek, Kontrola2, NULL, NULL, NULL,
                        NULL, NULL, TRUE, 1 );
    mdlOutput_prompt( "Identifikuj celek" );
}
```

```
Private void Kontrola2
(
void
)
{
   int     status;

    /* nacteni elementu do pameti a do souboru */
    fposc = mdlElement_getFilePos( FILEPOS_CURRENT, 0 ) ;
    if( ( status = mdlElmdscr_read( &element_CD, fposc, 0, 0, NULL ) ) != 0 )
    {
    if ( globals.pouzit_teziste == -1 )
                    mdlState_setFunction( STATE_COMPLEX_DYNAMICS , Transformace_profilu_CD ) ;
            else
                    mdlState_setFunction( STATE_DATAPOINT , Transformace_profilu_CD ) ;
    }
    strcpy(globals.vybrano_c,"ano");
    dbP=mdlDialog_find(DIALOGID_Dialog_Box,NULL);
    if (dbP!=NULL)
            mdlDialog_itemsSynch(dbP);
}

Private void Znovu2
(
void
)
   {
    /* smycka pri spatnem zadani elementu */
    mdlOutput_error( "Zadej prvni profil znovu    <RESET>" );
    mdlState_setFunction( STATE_RESET, Ident_celek );
   }


Public cmdName void uvolni_celek(void)
cmdNumber CMD_uvolni_celek
{


    /* smazani stareho a vykresleni noveho celku */
    mdlElmdscr_undoableDelete(element_CD, fposc, TRUE ) ;
    mdlElmdscr_display( element_CD, 0,XORDRAW         ) ;
    mdlElmdscr_add( element_CD ) ;

    element_CD = NULL ;
    strcpy(globals.vybrano_c,"ne");
    dbP=mdlDialog_find(DIALOGID_Dialog_Box,NULL);
    if (dbP!=NULL)
            mdlDialog_itemsSynch(dbP);
    mdlState_startDefaultCommand() ;
}


Public cmdName void uvolni_ulomek(void)
cmdNumber CMD_uvolni_ulomek
{

    /* smazani stareho a vykresleni noveho celku */
    mdlElmdscr_undoableDelete(element_ED, fpos, TRUE ) ;
    mdlElmdscr_display( element_ED, 0,XORDRAW         ) ;
    mdlElmdscr_add( element_ED ) ;

    element_ED = NULL ;
```

```
    strcpy(globals.vybrano_u,"ne");
    dbP=mdlDialog_find(DIALOGID_Dialog_Box,NULL);
    if (dbP!=NULL)
            mdlDialog_itemsSynch(dbP);
    mdlState_startDefaultCommand() ;
}


Public cmdName void Ident_Elementu(void)
cmdNumber CMD_identifikace_ulomek
{
    /* nastavebi pocitadla na nulu */
    cnt=0;
   mdlLocate_init();
    /*      spusteni procesu identifikace     */
    mdlState_startModifyCommand( Ident_Elementu, Kontrola1, NULL, NULL, NULL,
                        NULL, NULL, TRUE, 1 );
   mdlOutput_prompt( "Identifikuj profil  c.1 " );
}


Private void Kontrola1
(
void
)
{
   int     status;

    /* nacteni elementu do pameti a do souboru */
    fpos = mdlElement_getFilePos( FILEPOS_CURRENT, 0 ) ;
    if( ( status = mdlElmdscr_read( &element_ED, fpos, 0, 0, NULL ) )    != 0 )
    {
      mdlOutput_error("");
      /*  nastaveni "zavesene" fce. na udalost, podle nastaveni globalni
       promenne globals.pouzit_teziste.  */

          if ( globals.pouzit_teziste == -1 )
                mdlState_setFunction( STATE_COMPLEX_DYNAMICS , Transformace_profilu ) ;
          else
                mdlState_setFunction( STATE_DATAPOINT , Transformace_profilu ) ;
    }
    strcpy(globals.vybrano_u,"ano");
    dbP=mdlDialog_find(DIALOGID_Dialog_Box,NULL);
    if (dbP!=NULL)

          mdlDialog_itemsSynch(dbP);
}


Private void Znovu1
(
void
)
   {
   mdlOutput_error( "Zadej prvni profil znovu    <RESET>" );
    mdlState_setFunction( STATE_RESET, Ident_Elementu );
   }
```

```
/*-------------------------------------------------------------------+
|                                                                    |
| 6.cast - definice vlastnich transformacnich funkci                 |
|                                                                    |
+-------------------------------------------------------------------*/

Private void Transformace_profilu( Dpoint3d *pt)
{
    int     status ;

    if (element_ED==NULL) return;
    if ( globals.pouzit_teziste == -1 )
    {
        /* funkce na vypocet teziste */
        switch (element_ED->el.hdr.ehdr.type)
        {
            case SOLID_ELM:
                mdlOutput_prompt( "SOLID_ELM" ) ;
                if ( status = mdlMeasure_volumeProperties( NULL, NULL, NULL, &bod1, NULL,
                                NULL, NULL, NULL, NULL, NULL, element_ED,
                            globals.tolerance, NULL ) != SUCCESS )
                {
                    memcpy(&bod1,pt,sizeof(Dpoint3d));
                    printf(" BAD ");
                }
                break;

            case SURFACE_ELM:
                mdlOutput_prompt( "SURFACE_ELM" ) ;
                if ( status = mdlMeasure_surfaceProperties( NULL, &bod1, NULL,
                                NULL, NULL, NULL, NULL, NULL, element_ED,
                            globals.tolerance, NULL ) != SUCCESS )
                {
                    memcpy(&bod1,pt,sizeof(Dpoint3d));
                    printf(" BAD ");
                }
                break;

            default :
                {
                    Dpoint3d plow;
                    Dpoint3d phigh;
                    mdlOutput_prompt( "BSPLINE_SURFACE_ELM" ) ;
                    plow.x = mdlCnv_fromScanFormat(element_ED->el.hdr.ehdr.xlow);
                    plow.y = mdlCnv_fromScanFormat(element_ED->el.hdr.ehdr.ylow);
                    plow.z = mdlCnv_fromScanFormat(element_ED->el.hdr.ehdr.zlow);
                    phigh.x = mdlCnv_fromScanFormat(element_ED->el.hdr.ehdr.xhigh);
                    phigh.y = mdlCnv_fromScanFormat(element_ED->el.hdr.ehdr.yhigh);
                    phigh.z = mdlCnv_fromScanFormat(element_ED->el.hdr.ehdr.zhigh);

                    mdlVec_subtractPoint(&bod1,&phigh,&plow);
                    mdlVec_scale(&bod1,&bod1,0.5);
                    mdlVec_addPoint(&bod1,&bod1,&plow);

                    break;
                }       }

}
    else
        memcpy(&bod1,pt,sizeof(Dpoint3d));
        mdlWindow_cursorTurnOff() ;
```

```
    /* nastaveni zavesnych funkci na udalost */
    mdlState_setFunction( STATE_COMPLEX_DYNAMICS, Toceni_el ) ;
    mdlState_setFunction( STATE_DATAPOINT, trigger_toceni     ) ;
}


Private void Toceni_el
(
Dpoint3d *pt,
int                 view,
int                 drawMode )
{
    static Dpoint3d     oldpt;
    Dpoint3d            delta;
    RotMatrix           rMatrix ;
    Transform           tMatrix ;
    static int          oldview ;
    int                     stat ;
    void                *clipP;
    MSElementUnion      shape ;
    int                 status;
    MSElementDescr      *oldElement_ED ;
    MSElementUnion      elem ;
    Dpoint3d            p2(2),n;
    Dpoint3d            ptc=*pt;
    int                 vymaz;

    if ( element_ED == NULL )
    {
            return ;
    }
    if (toc_to==0) {cnt=0;oldview=-1; return;}
    if (oldview != view ) cnt = 0 ;

    oldview = view ;

    if (globals.metoda_pohybu == -1)
    {
    int over;
    /* transformace souradnic */
            mdlCurrTrans_begin();
            mdlCurrTrans_masterUnitsIdentity (FALSE);
            mdlCurrTrans_rotateByView (view);
            mdlCurrTrans_translateOriginWorld (pt);

            p2(0).x=-globals.q;         p2(2).x= globals.q;
            p2(0).y=-globals.q;         p2(2).y= globals.q;
            p2(0).z= 0.0;               p2(2).z= 0.0;
            p2(1).x= globals.q;         p2(3).x=-globals.q;
            p2(1).y=-globals.q;         p2(3).y= globals.q;
            p2(1).z= 0.0;               p2(3).z= 0.0;
            p2(4).x=-globals.q;
            p2(4).y=-globals.q;
            p2(4).z= 0.0;

            /* vytvoreni okna fence */
            mdlShape_create(&shape,NULL,p2,5,1);
            mdlElement_display(&shape,XORDRAW);
            mdlFence_fromShape(&shape);
            mdlClip_getFence(&clipP);

            if (mdlClip_isElemInside(&over, element_ED, clipP, view, TRUE))
```

```
            {
                mdlClip_free(&clipP);
            }
            else
            {
                mdlCurrTrans_end();
                mdlClip_free(&clipP);
                cnt=0;
                return;
            }
            /* konec transformace */
            mdlCurrTrans_end();
    }
    /* nastaveni hlavni transformace */
mdlCurrTrans_begin();
mdlCurrTrans_masterUnitsIdentity (FALSE);
mdlCurrTrans_rotateByView (view);
mdlCurrTrans_translateOriginWorld (&bod1);

mdlTMatrix_getIdentity( &tMatrix ) ;

    mdlCurrTrans_invtransPointArray(&ptc,&ptc,1);
    mdlCurrTrans_invtransPointArray(&bod1,&bod1,1);

    /* vypocet uhlu natoceni */
    if (ptc.x - bod1.x == 0)
            Uhel_z = fc_piover2;
   else
            Uhel_z = atan ((ptc.y - bod1.y) / (ptc.x - bod1.x));
    if (ptc.x - bod1.x < 0)
            Uhel_z += fc_pi;
    if (cnt++<3)
    {
            if (cnt==1)
                vymaz=ERASE;
            else
                vymaz=XORDRAW;
            oldpt=ptc;
            Uhel_oz=Uhel_z;
    }
    mdlOutput_printf(MSG_PROMPT,"%3.2f",Uhel_z/fc_2pi*360.0);

    if (globals.fixace_rotace != -1)
            mdlTMatrix_rotateByAngles( &tMatrix, &tMatrix, NULL, NULL, Uhel_z-Uhel_oz) ;
    Uhel_oz=Uhel_z;
    mdlVec_subtractPoint(&delta,&ptc,&oldpt);
    if (globals.fixace_posuvu != -1)
    {
            mdlTMatrix_translate(&tMatrix,&tMatrix,delta.x,delta.y,0);
            mdlVec_addPoint(&bod1,&bod1,&delta) ;
    }
    mdlCurrTrans_transformPointArray(&bod1,&bod1,1);

    /* transformace elementu */
    mdlElmdscr_display( element_ED, 0, vymaz);

    if ( stat = mdlElmdscr_transform( element_ED, &tMatrix ) == SUCCESS )
    {
            mdlElmdscr_display( element_ED, 0, XORDRAW ) ;
            if (globals.pridavat==-1)
                mdlElmdscr_add( element_ED );
    }
```

```
    /* podminka pro zapamatovani */
    if (globals.zapamatovat == -1)
            if (t++%frame==0)
            {
                    if (obr_cnt==0)
                    {
                            mdlElmdscr_duplicate (&startelem_ED, element_ED);
                    }
                    if ((obr=(animT*)realloc(obr,(obr_cnt+1)*sizeof(animT)))!=NULL)
                    {
                            obr[obr_cnt].view=view;
                            obr[obr_cnt].pt=bod1;
                            obr[obr_cnt].tMatrix=tMatrix;
                            obr_cnt++;
                    }
            }
    mdlCurrTrans_end();
}


Private void Transformace_profilu_CD( Dpoint3d *pt)
{
    int     status ;
    mdlWindow_cursorTurnOff() ;

    if (element_CD==NULL) return;
    if ( globals.pouzit_teziste == -1 )
    {
            switch (element_CD->el.hdr.ehdr.type)
            {
                    case SOLID_ELM:
                            mdlOutput_prompt( "SOLID_ELM" ) ;
                            if ( status = mdlMeasure_volumeProperties( NULL, NULL, NULL, &bod1, NULL,
                                            NULL, NULL, NULL, NULL, NULL, element_CD,
                                    globals.tolerance, NULL ) != SUCCESS )
                            {
                                    memcpy(&bod1,pt,sizeof(Dpoint3d));
                                    printf(" BAD ");
                            }
                            break;
                    case SURFACE_ELM:
                            mdlOutput_prompt( "SURFACE_ELM" ) ;
                            if ( status = mdlMeasure_surfaceProperties( NULL, &bod1, NULL,
                                            NULL, NULL, NULL, NULL, NULL, element_CD,
                                    globals.tolerance, NULL ) != SUCCESS )
                            {
                                    memcpy(&bod1,pt,sizeof(Dpoint3d));
                                    printf(" BAD ");
                            }
                            break;
                    default :
                            {
                                    Dpoint3d plow;
                                    Dpoint3d phigh;

                                    mdlOutput_prompt( "BSPLINE_SURFACE_ELM" ) ;

                                    plow.x = mdlCnv_fromScanFormat(element_CD->el.hdr.ehdr.xlow);
                                    plow.y = mdlCnv_fromScanFormat(element_CD->el.hdr.ehdr.ylow);
                                    plow.z = mdlCnv_fromScanFormat(element_CD->el.hdr.ehdr.zlow);
                                    phigh.x = mdlCnv_fromScanFormat(element_CD->el.hdr.ehdr.xhigh);
                                    phigh.y = mdlCnv_fromScanFormat(element_CD->el.hdr.ehdr.yhigh);
```

```
                                phigh.z = mdlCnv_fromScanFormat(element_CD->el.hdr.ehdr.zhigh);

                                mdlVec_subtractPoint(&bod1,&phigh,&plow);
                                mdlVec_scale(&bod1,&bod1,0.5);
                                mdlVec_addPoint(&bod1,&bod1,&plow);

                                break;
                        }
                }

        }
        else
        memcpy(&bod1,pt,sizeof(Dpoint3d));

        mdlState_setFunction( STATE_COMPLEX_DYNAMICS, Toceni_celku ) ;
        mdlState_setFunction( STATE_DATAPOINT, trigger_toceni      ) ;
}


Private void Toceni_celku
(
Dpoint3d *pt,
int             view,
int             drawMode )
{
    static Dpoint3d     oldpt;
    Dpoint3d            delta;
    double                  Str_x, Str_y;
    Transform           tMatrix ;
    static int          oldview ;
    int                     stat ;
    void                *clipP;
    MSElementUnion      shape ;
    int                 status;
    Dpoint3d            p2(2),n;
    Dpoint3d            ptc=*pt;
    int                 vymaz;

    if ( element_CD == NULL )
    {
            return ;
    }
    if (toc_to==0) {cnt=0;oldview=-1; return;}
    if (oldview != view ) cnt = 0 ;
    oldview = view ;
    if (globals.metoda_pohybu == -1)
    {
            int over;
            mdlCurrTrans_begin();
            mdlCurrTrans_masterUnitsIdentity (FALSE);
            mdlCurrTrans_rotateByView (view);
            mdlCurrTrans_translateOriginWorld (pt);

            mdlCurrTrans_invtransPointArray(&bod1,&bod1,1);

            p2(0).x=-globals.q;         p2(2).x= globals.q;
            p2(0).y=-globals.q;         p2(2).y= globals.q;
            p2(0).z= bod1.z;            p2(2).z= bod1.z;
            p2(1).x= globals.q;         p2(3).x=-globals.q;
            p2(1).y=-globals.q;         p2(3).y= globals.q;
            p2(1).z= bod1.z;            p2(3).z= bod1.z;
            p2(4).x=-globals.q;
```

```
            p2[4].y=-globals.q;
            p2[4].z= bod1.z;

            mdlCurrTrans_transformPointArray(&bod1,&bod1,1);
            mdlShape_create(&shape,NULL,p2,5,1);
            mdlElement_display(&shape,XORDRAW);
            mdlFence_fromShape(&shape);
            mdlClip_getFence(&clipP);

            if (status=mdlClip_isElemInside(&over, element_CD, clipP, view, TRUE))
            {
                    mdlClip_free(&clipP);
                    mdlCurrTrans_end();
            }
            else
            {
                    mdlCurrTrans_end();
                    mdlClip_free(&clipP);
                    cnt = 0 ;
                    return;
            }
    }
    mdlCurrTrans_clear();

mdlCurrTrans_begin();
mdlCurrTrans_masterUnitsIdentity (FALSE);
mdlCurrTrans_rotateByView (view);
mdlCurrTrans_translateOriginWorld (&bod1);

    mdlTMatrix_getIdentity( &tMatrix ) ;

    mdlCurrTrans_invtransPointArray(&ptc,&ptc,1);
    mdlCurrTrans_invtransPointArray(&bod1,&bod1,1);

if (ptc.x - bod1.x == 0)
            Uhel_z = fc_piover2;
   else
            Uhel_z = atan ((ptc.y - bod1.y) / (ptc.x - bod1.x));
            if (ptc.x - bod1.x < 0)
            Uhel_z += fc_pi;
    if (cnt++<3)
    {               if (cnt==1)
                    vymaz=ERASE;
            else
                    vymaz=XORDRAW;
                    oldpt=ptc;
                    Uhel_oz=Uhel_z;
    }
    mdlOutput_printf(MSG_STATUS,"%f",Uhel_z/fc_2pi*360.0);

    if (globals.fixace_rotace != -1)
            mdlTMatrix_rotateByAngles( &tMatrix, &tMatrix, NULL, NULL, Uhel_z-Uhel_oz) ;
    Uhel_oz=Uhel_z;
    mdlVec_subtractPoint(&delta,&ptc,&oldpt);
    if (globals.fixace_posuvu != -1)
    {
            mdlTMatrix_translate(&tMatrix,&tMatrix,delta.x,delta.y,0);
            mdlVec_addPoint(&bod1,&bod1,&delta) ;
    }
    mdlCurrTrans_transformPointArray(&bod1,&bod1,1);
    mdlElmdscr_display( element_CD, 0, vymaz ) ;
```

```
    if ( stat = mdlElmdscr_transform( element_CD, &tMatrix ) == SUCCESS )
    {
            mdlElmdscr_display( element_CD, 0, XORDRAW ) ;
            if (globals.pridavat==-1)
                    mdlElmdscr_add( element_CD );
    }
    mdlCurrTrans_end();
}


/* funkce, ktera reaguje na zmenu teziste */
Public cmdName void zmena_teziste ( void )
cmdNumber CMD_zmena_teziste
{
    if (globals.pouzit_teziste==-1)
    {
            pt_teziste=bod1;
            Transformace_profilu(&pt_teziste);
    }
    else
            bod1=pt_teziste;
}

/*-------------------------------------------------------------------------+
|                                                                          |
| 7.cast - definice funkci pro jednoduchou animaci                         |
|                                                                          |
+-------------------------------------------------------------------------*/

/* funkce pro doprednou animaci */

Public cmdName void Animace_dopredna ( void )
cmdNumber CMD_animace_tam
{
    int i;
    MSElementDescr    *edP=NULL;
    mdlElmdscr_duplicate (&edP, startelem_ED);

    if ( startelem_ED == NULL )
            return ;

    for (i=0;i<obr_cnt;i++)
    {
            if (globals.trasovat==0)
            if (edP!=NULL) mdlElmdscr_display( edP, 0, XORDRAW ) ;

            mdlCurrTrans_begin();
            mdlCurrTrans_masterUnitsIdentity (FALSE);
            mdlCurrTrans_rotateByView (obr[i].view);
            mdlCurrTrans_translateOriginWorld (&obr[i].pt);
            mdlElmdscr_display( edP, 0, XORDRAW ) ;

            if (mdlElmdscr_transform( edP, &obr[i].tMatrix ) == SUCCESS )
            {
                    mdlElmdscr_display( edP, 0, XORDRAW ) ;
                    if (globals.pridavat==-1)
                            mdlElmdscr_add( edP );
            }
            mdlCurrTrans_end();
    }
    mdlElmdscr_freeAll( &edP )
```

```
/* funkce pro zpetnou animaci */

Public cmdName void Animace_zpetna ( void )
cmdNumber CMD_animace_zpet
{
    int i,j;
    MSElementDescr    *edP=NULL;
    MSElementDescr    *oldedP=NULL;
    Transform         btMatrix;

    for (j=obr_cnt-1;j>=0;j—)
    {
        mdlElmdscr_duplicate (&edP, startelem_ED);
        for (i=0;i<obr_cnt;i++)
        {
            mdlCurrTrans_begin();
            mdlCurrTrans_masterUnitsIdentity (FALSE);
            mdlCurrTrans_rotateByView (obr(i).view);
            mdlCurrTrans_translateOriginWorld (&obr(i).pt);
            if (i==0) mdlTMatrix_getIdentity (&btMatrix ) ;
            mdlTMatrix_multiply(&btMatrix,&btMatrix,&obr(i).tMatrix);
            if (i==j)
            {
                if (globals.trasovat==0)
                if (oldedP!=NULL) mdlElmdscr_display( oldedP, 0, XORDRAW ) ;
                if (mdlElmdscr_transform( edP, &btMatrix ) == SUCCESS )
                    mdlElmdscr_display( edP, 0, XORDRAW ) ;
                if (globals.pridavat==-1)
                    mdlElmdscr_add( edP );
                oldedP=edP;
                mdlCurrTrans_end();
                break;
            }
            mdlCurrTrans_end();
        }
    }
    mdlElmdscr_freeAll( &edP ) ;
}


/*—————————————————————————————————————————————————————————+
|                                                             |
| 8.cast - definice funkce umoznujici vyskoceni s transformacni smycky                    |
|                                                             |
+—————————————————————————————————————————————————————————*/

Private void trigger_toceni
(
Dpoint3d *pt,
int              view,
int              drawMode )
{
    /* nastaveni prepinace */
    toc_to=++toc_to%2;
}


Private void exit_ven( void )
{
    mdlElmdscr_add( element_ED );
    mdlElmdscr_freeAll( &element_ED ) ;
}
```

```
/*-------------------------------------------------------------------+
|                                                                    |
| 9.cast - definice funkce umoznujici prilepeni elementu             |
|                                                                    |
+-------------------------------------------------------------------*/

Public cmdName void prilepit(void)
cmdNumber CMD_prilepit
{
    MSElementDescr    *edP,*outEdP,*origEdP;
    MSElement         header;

    mdlComplexChain_createHeader (&header, FALSE, -1);
    mdlElmdscr_new (&outEdP, NULL, &header);
    mdlElmdscr_duplicate (&origEdP, element_CD);
    mdlElmdscr_undoableDelete( element_CD, fposc, FALSE ) ;

    /* postupne rozebrani element descriptoru
    a spojovani dvou elementu k sobe*/
    if (origEdP->h.firstElem==NULL)
    {
      mdlElmdscr_new (&edP, NULL, &origEdP->el);
            element_ED->h.isValid=FALSE;
            mdlElmdscr_validate(outEdP,MASTER_FILE);
            mdlElmdscr_appendDscr (outEdP, edP);
    }
    else
            for (origEdP=origEdP->h.firstElem; origEdP; origEdP=origEdP->h.next)
            {
                    mdlElmdscr_new (&edP, NULL, &origEdP->el);
                    edP->el.ehdr.complex = 0;
                    element_ED->h.isValid=FALSE;
                    mdlElmdscr_validate(outEdP,MASTER_FILE);
                    mdlElmdscr_appendDscr (outEdP, edP);
            }
    mdlElmdscr_duplicate (&origEdP, element_ED);
    mdlElmdscr_undoableDelete( element_ED, fpos, FALSE ) ;

    if (origEdP->h.firstElem==NULL)
    {
      mdlElmdscr_new (&edP, NULL, &origEdP->el);
            element_ED->h.isValid=FALSE;
            mdlElmdscr_validate(outEdP,MASTER_FILE);
            mdlElmdscr_appendDscr (outEdP, edP);
    }
    else
            for (origEdP=origEdP->h.firstElem; origEdP; origEdP=origEdP->h.next)
            {
                    mdlElmdscr_new (&edP, NULL, &origEdP->el);
                    edP->el.ehdr.complex = 0;
                    element_ED->h.isValid=FALSE;
                    mdlElmdscr_validate(outEdP,MASTER_FILE);
                    mdlElmdscr_appendDscr (outEdP, edP);
            }
    mdlElmdscr_add (outEdP);
}
```

```
#-------------------------------------------------------------------------
#
# rotace.mke
#
#-------------------------------------------------------------------------

#-------------------------------------------------------------------------
# definuje specialni makra pro tento priklad
#-------------------------------------------------------------------------

baseDir    = e:/dpl/src/
objectDir  = e:/dpl/objects/
privateInc = $(baseDir)

#-------------------------------------------------------------------------
# definice maker, ktere jsou zahrnuty v linkovaci
#-------------------------------------------------------------------------

Objs  = $(objectDir)rotace.mo \
        $(mdlLibs)ditemlib.ml

Rscs  =  $(objectDir)rotace.rsc \
         $(objectDir)rottyp.mp  \
         $(objectDir)rottyp.rsc


#-------------------------------------------------------------------------
# mdl.mki - definuje implicitni pravidla pro vytvoreni *.rsc, *.mo
#-------------------------------------------------------------------------

%ifdef BSI
%include $(MS)/include/publish/mdl.mki
%else
%include $(MS)/mdl/include/mdl.mki
%endif


#-------------------------------------------------------------------------
# compilaci aplikacniho typu vytvori zdrojovy soubor
#-------------------------------------------------------------------------

$(objectDir)rottyp.r        : $(baseDir)rottyp.mt $(baseDir)rotace.h $(baseDir)rottxt.h

$(objectDir)rottyp.rsc   : $(objectDir)rottyp.r $(baseDir)rotace.h $(baseDir)rottxt.h


#-------------------------------------------------------------------------
# kompiluje dialogovy soubor pomoci rcomp
#-------------------------------------------------------------------------

$(objectDir)rotace.rsc      : $(baseDir)rotace.r $(baseDir)rotace.h $(baseDir)rottxt.h


#-------------------------------------------------------------------------
# kompiluje MDL zdrojovy soubor pomoci mcomp
#-------------------------------------------------------------------------

$(objectDir)rotace.mo       : $(baseDir)rotace.mc $(baseDir)rotace.h
```

```
#-------------------------------------------------------------------------
# linkuje MDL programovy soubor z posun.mo na ditem.lib pomoci rlink
#-------------------------------------------------------------------------

$(objectDir)rotace.mp          : $(Objs)
    $(msg)
    > $(objectDir)temp.cmd
    -a$□
    -s6000
    $(linkOpts)
    $(Objs)
    <
    $(linkCmd) □$(objectDir)temp.cmd
    ~time


#-------------------------------------------------------------------------
# vytvori zdrojovy MDL program pomoci rlib
#-------------------------------------------------------------------------

$(MDLAPPS)rotace.ma            : $(Rscs)
    $(msg)
    > $(objectDir)temp.cmd
    -o$□
      $(Rscs)
    <
    $(rscLibCmd) □$(objectDir)temp.cmd
    ~time
```