TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 1802T007 – Informační technologie

Software pro simulaci blokových schémat

Software for simulation of blocks´ schemes

Diplomová práce

Autor: **Bc. Miroslav Vavroušek**
Vedoucí práce: doc. Ing. Jiřina Královcová, Ph.D
Konzultant: Ing. Miloš Hernych

V Liberci 21. 5. 2010

Anotace

Práce se zabývá návrhem a implementací simulačního prostředí pro popis kinematických a dynamických modelů. Navržené virtuální modely jsou dále zpracovány vytvořeným programem, který umožňuje simulaci modelu, jeho vizualizaci a následné zpracování vypočtených hodnot.

Návrhu simulačního prostředí je věnovaná teoretická část. Vytvořený simulační systém je složen z dvaceti druhů funkčních bloků, z kterých je sestaveno blokové schéma. Každý blok, podle svého typu a definovaných parametrů bloku, realizuje funkci mezi svými vstupními porty a výsledek této operace předává na výstupní porty bloku. Bloky popisují různé matematické operace, mezi které patří sčítaní vstupů, porovnávání vstupů nebo například generovaní obdélníkového signálu. Navržené simulační prostředí obsahuje i rozšiřující bloky pro realizaci vizualizace nebo bloky pro komunikaci se zařízeními typu *Programmable Logic Controller* (*PLC*). Bloky jsou propojeny vodiči informací. V teoretické části jsou také popsané použité nástroje při tvorbě programu, stručná charakteristika komunikačního protokolu *EPSNET* a popis syntaxe souborů pro vstup a výstup dat. Teoretickou část uzavírá informativní přehled jednotlivých typů bloků a jejich funkcí.

Praktická část práce se zabývá konkrétní realizací programu. Vytvořený software umožňuje simulaci blokových schémat. Program je navržen pro práci na platformě *Windows* a obsahuje nástroje pro vizualizaci procesů pomocí pohybu s 3D objekty v prostoru. Aplikace umožňuje znázornění průběhu sledovaných veličin ve formě grafu a uložení do souboru jako obrázek. Hodnoty lze také exportovat do textového souboru pro další zpracování. Program obsahuje rozhraní pro komunikaci s *PLC*. *PLC* je možné využít k řízení a sledování virtuálního modelu. Vestavěné rozhraní lze použít i k vzájemnému propojení virtuálního a reálného modelu a k jejich společné kooperaci pro co nejvěrnější nastavení virtuálního modelu. Program lze také využít k prezentaci animací konkrétních situací. Obsahem daného celku je i popis uživatelského rozhraní a nastínění možností programu. V další části následuje popis a návrh důležitých algoritmů. V posledním úseku praktické části jsou popsány dva ukázkové příklady pro usnadnění pochopení práce s programem. První příklad se zabývá pohybem dvou hmotných těles v soustavě pružin a tlumičů. Druhým příkladem je model křižovatky řízené semafory. Semafory je možné řídit pomocí *PLC*.

Na závěr je posouzena vhodnost programu při řešení typických úloh.

Annotation

This work deals with a proposal and an implementation of simulation environment for a description of kinematic and dynamic models. These proposed virtual models are then processed by created program which enables the model´s simulation, its visualization and consecutive calculated values´ processing.

The theoretical part is devoted to a proposal of simulation environment. A created simulation system consists of twenty sorts of function blocks which put together a block scheme. According to a block´s sort and defined parameters, each block implements a function among its input ports and then an outcome of this operation passes to the block´s output ports. The blocks describe various mathematical operations such as inputs´ addition, inputs´comparing or rectangular signal´s generating. The proposed simulation environment contains even some spreading blocks for visualization´s realization or blocks for communication with such kind of device as *Programmable Logic Controller* (*PLC*). Blocks are linked by an information cell bridge. In the theoretical part there are also described the instruments used for a creation of the program, brief characteristics of communications protocol *EPSNET* and there is also a description of file´s syntax for an input and an output of data. The theoretical part closures an informative view to particular sorts of blocks and to its functions.

The practical part deals with a concrete realization of the program. The created software enables a simulation of the blocks´ schemes. The program is designed to work at a Windows platform and it contains the instruments for a visualization of processes by means of moving with 3D objects in space. The application enables a visualization of a process of monitored quantities into a form of a graph and to save it as a picture to a file. It is also possible to export the values to a text file for some extra manipulation. The program contains interface for a communication with *PLC*. *PLC* might be used for controlling and monitoring of a virtual model. The inbuilt interface can be also used for an interconnection between a real and a virtual model and for their co-operation leading to the most devoted virtual model´s setting. It is also possible to use a program for a presentation of an animation of some concrete situations. A content of the given complex is a description of a user interface and an adumbration of program´s possibilities. In the following part there is a description and a proposal of the important algorithms. In the last part of the practical work there are described two examples for a simplification of understanding how to work with the program. The first example deals with a movement of two mass solids in a system

of springs and shock-absorbers. The second example is a model of cross-roads controlled by traffic lights. It is possible to control the traffic lights by *PLC*.

At the conclusion it is considered the convenience of the program for solving some typical cases.

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé diplomové práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé diplomové práce (prodej, zapůjčení apod.).

Jsem si vědom toho, že užít své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum 21. 5. 2010

Podpis

Poděkování

Na tomto místě bych chtěl především poděkovat vedoucí mé diplomové práce doc. Ing. Jiřině Královcové, Ph.D. za cenné rady, připomínky k práci a za odborné vedení při jejich tvorbě. Dále děkuji Bc. Radimu Kracíkovi za pomoc při návrhu grafických modelů a praktické rady k tématu. V neposlední řadě bych zde chtěl poděkovat své rodině za veškerou podporu během celého studia.

Úvod

Práce se zabývá návrhem simulačního prostředí. Prostředí je implementováno v aplikaci nazvané „*VirMo Viewer*“. Aplikace umožňuje simulaci blokových schémat, vizualizaci simulovaného systému a sledování průběhu zvolených veličin. K stavbě schémat je možné použít program „*VirMo Editor*“. Program „*VirMo Editor*“ byl vytvořen jako magisterský semestrální projekt.

Nejprve bylo nutné navrhnout simulační blokové prostředí a definovat postup výpočtu schématu. Vytvořené prostředí obsahuje dvacet typů bloků. Jednotlivé bloky jsou propojeny vodiči informací. Z důvodu standardizace bylo šestnáct typů bloku definováno shodně s vybranými bloky obsaženými v prostředí *MATLAB Simulink*. Čtyři doplňkové bloky obohacují základní výpočetní možnosti o další rozšiřující funkce. Prvním blokem je *VMObjekt*, který slouží pro realizaci vizualizace. Druhým je *VMTrajektorie*, který pracuje s tabulkou dat. Třetím blokem je *VMPLC*. Tento blok uskutečňuje komunikaci s *PLC*. Posledním blokem je *VMSchema*, který umožňuje vytváření nižších funkčních celků, z nichž je výsledné schéma složeno.

Poté bylo simulační prostředí implementováno do aplikace umožňující simulování sestaveného schéma. Pro tvorbu aplikace bylo nutné navrhnout a implementovat proces výpočtu schématu, vizualizace simulovaného modelu a zpracování vypočtených hodnot. Program umožňuje vypočtené hodnoty vykreslit ve formě grafu a uložit vypočtené hodnoty do souboru. Pro spolupráci se zařízením *PLC* bylo nutné vytvořit rozhraní pro komunikaci pomocí protokolu *EPSNET*. Program byl navrhnut v prostředí *Visual Studio* v jazyku *C++*. Pro demonstraci funkcí programu jsem vytvořil dva ukázkové příklady.

Teoretická část

V teoretické části je uveden popis použitých prostředků a přístupů, z kterých vycházím v následující praktické části. Teoretická část obsahuje popis knihovny *OpenGL a* použitých vývojových prostředí pro tvorbu programu. V kapitole je popsáno i komunikačního rozhraní *EPSNET*. V závěru této časti je deklarovaná syntaxe souborů pro vstup a výstup dat. Mezi ně patří soubory s definicí *3D* modelů, soubory pro vstup dat textur k obohacení modelů a také zde jsou popsány soubory pro uložení projektu a simulačního schématu. Následuje stručný popis uložení a definice schémat programu *MATLAB Simulink*. Poslední popsaný je soubor s tabulkou dat, který se používá pro vstup a výstup dat z prostředí programu, jenž je podporovaný mnoha tabulkovými procesory. Díky velmi snadné syntaxi není problém soubor dále zpracovat. Textové typy souborů jsou demonstrovány stručnými a zjednodušenými příklady.

1.1 Modelování systémů pomocí blokových schémat

Modelování systémů je velmi často používanou metodou, která se využívá v širokém spektru oborů. Modelování usnadňuje pochopení zkoumaného systému a nalezení řešení hledaných problémů. Vstupní parametry a model lze definovat, případně zpřesňovat experimenty. Každý model je zatížen nejistotami. Nepřesnosti mohou vzniknout ve struktuře modelů například zanedbáním některých faktorů ovlivňujících výsledek simulace. Přesnost výsledků je také ovlivněná množstvím a komplexností vstupních dat. Při realizaci numerického modelování pomocí počítače je výsledek také limitován přesností uložení čísla v paměti počítače. To je velmi důležité pro zhodnocení věrohodnosti výsledků.

Numerické modelování se zabývá řešením problémů pro konkrétní číselné hodnoty. Veličiny jsou popsané diskrétně ve formě vektorů. Numerické modelování lze vyžít ve velkém množství úloh a postihuje velmi širokou disciplínu. Řadu fyzikálních problémů lze popsat pomocí diferenciálních rovnic. Tyto rovnice lze snadno zdiskretizovat. Pro zkoumané situace jsou vytvořeny numerické modely počítané v definovaných krocích. Numerické modely jsou vhodné i ke zkoumání hypotetických situací. Dříve byly tyto výpočty velmi časově náročné, a proto se obvykle využívalo analytického řešení úloh, jež spočívá v hledání řešení pomocí známých vztahů a postupných úprav. To je pro některé typy úloh velmi obtížné a může být i nemožné. Vývoj počítačů přinesl numerickému

modelování velký rozmach. V dnešní době se velmi často provádí modelování pomocí počítačů. Díky vysokému výkonu lze k výpočtu použít i osobních počítačů. Matematický model řešeného problému je přepsán do tvaru programu. Výhodou virtuálních modelů bývá snadné přizpůsobení pro konkrétní úlohu.

Diferenční rovnice lze přepsat do podoby blokového schématu, které je složeno z funkčních bloků. Bloky jsou spojeny vodiči informací, které realizují přenos dat z výstupních portů na vstupní porty dalšího bloku. Celé schéma je cyklicky počítáno od počátečního do koncového času v hodnotách simulačního kroku. A tím jsou postupně vytvářeny vektory popisující průběhy veličin.

Funkční Blok, dále jen blok, si lze představit jako zapouzdření funkce do podoby samostatně funkčního celku. Každý blok podle svého typu realizuje operaci, která podle hodnot vstupních portů a nastavených parametrů bloku jednoznačně přiřadí hodnotu výstupním portům. Pomocí tohoto jednoduchého principu lze sestavovat i rozsáhlá řídicí schémata.

Vodič informace, také nazývaný *line*, slouží k propojení bloků pro přenos informace. Každý vodič informace má pouze jeden zdrojový blok a může být pomocí větvení připojen k více cílovým objektům. Přenesení dat může proběhnout až ve chvíli, kdy byl zdrojový blok zpracován.

1.2 Programovací jazyk

Programovací jazyk je formální jazyk pro popis algoritmů. Je zapsaný programátorem a realizovaný počítačovými technickými prostředky. Jazyk je nástrojem programátora a usnadňuje jeho práci. Je složen z příkazů, posloupnost příkazů se nazývá program. Existují jazyky nižší a vyšší úrovně. Programovací jazyk nižší úrovně je například jazyk symbolických adres, který je velmi podobný strojovému kódu. Pro usnadnění práce programátorů byly vytvořeny vyšší programovací jazyky, jako například *Pascal* nebo *C*. Jazyky lze také rozdělit na kompilované a interpretované. Kompilované jsou přeloženy přímo do strojového kódu a patří mezi ně jazyk *Pascal*. V interpretovaných jazycích je program přeložen do mezikódu a poté je mezikód přeložen až pro konkrétní platformu. Překlad mezikódu může probíhat i za běhu programu. Typickým příkladem interpretovaného jazyka je jazyk *JAVA*. Jazyky je také možné rozdělit podle způsobu zápisu kódu na imperativní (procedurální) a deklarativní. Imperativní se dělí na strukturované, jako je třeba jazyk *C*, a na objektově orientované jazyky, jako je

například *Object Pascal*. Deklarativní mohou být rozděleny na funkcionální, mezi které patří jazyk *Lisp,* a logické. Představitelem logických jazyků je například *Prolog*.

1.3 Vývojové prostředí

Vývojové prostředí je program, který obsahuje editor kódu, kompilátor pro zvolený jazyk a obvykle i *debugger* sloužící pro hledání chyb v programu. Prostředí rozhodujícím způsobem ulehčuje práci programátorům. Editor kódu může mít implementovány další pomocné funkce, jako je zvýraznění syntaxe nebo nástroje pro práci s textem. *Kompilátor* bývá propojený s editorem kódu a chyby při kompilaci jsou označeny přímo v textu. Obsažený *debugger* může podporovat krokování programu a vkládání zarážek označovaných také *breakpoints*. Zarážky slouží k zastavení programu na zvoleném místě v kódu. Moderní prostředí využívají při tvorbě aplikací systémy *RAD* (anglická zkratka *Rapid Application Development* – umožňující rychlý vývoj aplikací). Systém je založen na následujícím principu: prostředí automaticky generuje části kódu například pro uživatelské prostředí. Stavba uživatelského rozhraní je poté realizována pomocí komponent představujících jednotlivé ovládací prvky. Komponenty mohou mít vizuální charakter, jako třeba tlačítko, nebo mohou být nevizuální, například časovač, a realizovat tak potřebnou skrytou funkci. Nástroje pro objektově orientované jazyky obvykle obsahují prohlížeč objektů. Prostředí mohou obsahovat řadu pomocných knihoven pro realizaci standardních funkcí, ale i vyšších funkcí vhodných pro konkrétní účel. Prostředí jsou často určena pro jediný jazyk. Existují ale i nástroje pro práci s celou paletou jazyků. Další jazyk lze obvykle přidat pomocí zásuvných modulů.

1.4 Historie OpenGL

Grafické rozhraní *OpenGL* začalo vznikat v roce 1992 za podpory firmy *SGI* a dalších firem z důvodů standardizace práce s různými druhy grafických akcelerátorů a na různých platformách. Podpory vykreslování grafiky pod *OpenGL* využívá mnoho aplikací pro zpracování grafiky, modelování jako jsou *CAD systémy* a počítačové hry. Z nejzákladnějšího pohledu je *OpenGL* jen seznam funkcí s popisy, jak by se měly chovat. Implementaci dané funkce zajišťuje výrobce sám, a proto je možné, že výsledky na dvou grafických akcelerátorech od dvou různých výrobců se mohou mírně vizuálně lišit (geometrie musí být zachována). Různá je také hardwarová podpora rozhraní. Funkce, které nejsou hardwarově podporovány, jsou počítány softwarově.

V současné době prožívá *OpenGL* velmi dynamicky vývoj pokrývající nové technologie. Dospěl až k nejnovější verzi 3.2. Naprostá většina dnešních moderních grafických karet obsahuje přímou hardwarovou podporu pro zpracování výpočtů.

1.5 Základní principy OpenGL

OpenGL není závislé na systému, nad kterým pracuje. Proto neumí pracovat například se vstupními periferiemi, s okny, se zvukem nebo s vykreslováním textu. Díky tomu může fungovat na různých platformách (*Windows, Unix*) a z tohoto důvodu si například zavádí i své základní typy (např. *GLuint - unsigned integer*).

OpenGL se chová jako stavový automat a to v tom smyslu, že při startu jsou proměnné nastavené na defaultní hodnoty a s těmi bude *OpenGL* pracovat, dokud nedojde k explicitnímu změnění (změna stavu). Například pokud dojde ke změně barvy, kterou se bude kreslit, *OpenGL* bude kreslit touto barvou až do dalšího nastavení barvy. To umožňuje změnu pouze parametrů potřebných k požadovanému vykreslení scény a tím přispívá k jeho snadnému použití.

OpenGL umožňuje vykreslovat základní standardní primitiva jako třeba body, úsečky, trojúhelníky a mnoho dalších. Tato primitiva jsou zadaná pomocí vrcholů a jejich skládáním se vytváří složitější objekty. Nejčastěji, a to hlavně v počítačových hrách, se vykreslují trojúhelníky pro svou univerzalitu, jasně definovanou normálu, a schopnost složit jakýkoli jiný tvar. Z tohoto důvodu mají grafické karty často optimalizované vykreslování právě trojúhelníků a při kreslení ostatních primitiv značně klesá výkon.

Vykreslovaný obraz je uložený a zobrazovaný z *framebufferu*. Proto, aby při vykreslování nedocházelo k blikání obrazu, jsou použity dva *framebuffery* a po každém snímku dochází k jejich záměně. Zdrojem informací pro tuto kapitolu je *OpenGL Programming Guide 1.23.2.1*

1.6 Jazyk C++

Programovací jazyk *C++* je objektovým programovacím jazykem a je následovníkem jazyka *C*. Jazyk *C* byl vybrán jako výchozí jazyk, protože je mnohostranný, stručný, relativně nízko úrovňový a vyhovuje většině úloh systémového programování. Jazyk *C* zůstává podmnožinou jazyka *C++*. Důležitým návrhovým kritériem byla jednoduchost. Byl zachován důraz na programovací prostředky pracující na nízké úrovni vhodné pro systémové programování. Objektové vlastnosti jazyka jsou inspirovány

jazykem *Simula67*. Schopnost přetěžovat operátory a možnost umístit deklaraci kamkoliv, kde se může umístit příkaz, připomíná *Algol68*. Jméno *C++* vyjadřuje evoluční charakter oproti jazyku *C*.

C++ nemá datové typy vyšší úrovně ani primitivní operace vyšší úrovně. Například neexistuje maticový typ s operátorem inverze nebo řetězcový typ s operací zřetězení. Přeje-li si uživatel takový typ, může ho definovat. Ve skutečnosti je definování nového obecně použitelného nebo aplikačně specifického typu nejzákladnější činností programátorů v *C++*. Dobře navržený uživatelský typ se liší od vestavěného typu pouze způsobem svého nadefinování a nikoliv způsobem použití.

Program lze větvit pomocí příkazu podmínky a příkazu pro opakování částí kódu. Každý program začíná spuštěním hlavní funkce s názvem *Main*. Program končí opuštěním hlavní funkce.

Jazyk *C++* byl navržen pro použití v tradičním prostředí pro kompilaci a běh programů, tj. vývojové prostředí pro *C* v systému *UNIX*. Jsou však dobré důvody pro používání *C++* v prostředí, kde je k dispozici výrazně větší podpora. Prostředky jako dynamické zavádění programu, inkrementální kompilace a databáze typových definicí se dají dobře použít, aniž se jazyk ovlivní. Údaje byly získány z knih: *Programujeme v Microsoft Visual C++ 1.23.2.1* a *Programovací jazyk C pro zelenáče 1.23.2.1*.

1.7 Prostředí Microsoft Visual Studio

Visual studio je vývojové prostředí vyvinuté společností *Microsoft*. Lze ho použít pro práci v mnoha různých jazycích. Mezi přímo podporované jazyky patří *Java*, *C#*, *C++*, *Visual Basic .NET*. Podporu dalších jazyků lze zajistit instalováním doplňků.

V prostředí lze vytvářet jak konsolové aplikace, tak i aplikace s grafickým uživatelským rozhraním pomocí *Windows Forms*. *Visual studio* je určeno pro vytváření aplikací na různých platformách společnosti *Microsoft* od platformy *Windows*, přes mobilní platformy *Windows Mobile* a *Windows CE* až po *platformu .NET*.

Editor zdrojových kódů pro usnadnění práce podporuje automatické doplňování textu, zvýraznění syntaxe kódu a zvýraznění párovosti závorek. V prostředí lze použít řadu standardních knihoven, například pro práci s textem, či pro dynamickou alokaci paměti a pro realizaci matematických funkcí. Vývojové prostředí má integrovaný *debugger* pracující na úrovni zdrojového i strojového kódu. *Debugger* podporuje krokování programů a vkládání *breakpoints*, bodů, na kterých je běh programu pozastaven.

Visual studio obsahuje integrovaný vizuální designér formulářů aplikací usnadňující tvorbu uživatelského rozhraní. Prostředí je dodáváno v několika verzích. Od verze Express, která je po zaregistrování ke stažení zdarma, ale obsahuje nejnižší paletu nástrojů, až po placené verze pro profesionální tvorbu aplikací s velkým důrazem kladeným na správu programu v týmu. Zdrojem informací pro tuto část je *Programujeme v Microsoft Visual C++1.23.2.1*.

1.8 Jazyk Object Pascal

Jazyk *Object Pascal* je objektovým nástupcem jazyka Pascal. Jeho autor Niklaus Wirth jazyk vyvinul pro firmu *Apple*. Později byl jazyk proslaven firmou *Borland*, která *Object Pascal* využívala ve svém integrovaném vývojovém prostředí *Turbo Pascal*. Jazyk existuje v několika mírně rozdílných implementacích.

Jazyk *Pascal* je procedurální programovací jazyk a byl navržen v roce 1970 v Curychu. Jazyk byl vytvořen pro osvojení základních technik začínajících programátorů. V roce 1971 byla vydána první reference jazyka, která byla po pozdějším zjištění nedostatků v roce 1974 novelizována. V roce 1981 byla vytvořena příslušná norma *ISO* a začaly vznikat komerční kompilátory. Jazyk umožňuje využití funkcí a procedur pro zapouzdření a snadné znovupoužití částí kódu programu. Dalšími možnostmi jazyka je nenáročné strukturování dat pomocí definování datových typů, záznamů, množin a ukazatelů. Pro řízení toku programu jsou k dispozici konstrukce podmíněných příkazů a konstrukce pro opakování úseku kódu.

Object Pascal přidává klasickému procedurálnímu programování možnost definování tříd objektu a poskytuje další vlastnosti usnadňující vývoj programu. Jazyk nerozlišuje velikost písmen ve zdrojových kódech (není *case-sensitive*) a umožňuje vytvářet přehledné a stručné programy. Funkce a procedury definované v jazyku jsou rozděleny na dvě části. V první deklarační části následující po klíčovém slovu *var* jsou zavedeny potřebné proměnné. V druhé části je implementován potřebný algoritmus pracující s deklarovanými proměnnými. Informace získány na webových stránkách *Pascal.webz.cz 1.23.2.1*.

1.9 Prostředí Borland Delphi

Prostředí *Delphi* je vývojové prostředí pro Jazyk *Object Pascal* od firmy *Borland*. Prostředí je uzpůsobeno pro vývoj aplikací pro operační systém Windows. Vizuální

vývojové prostředí podporuje systém RAD a skládá se ze tří základních komponent. Jsou jimi editor, ladící program a návrhář formulářů. Při návrhu aplikace všechny tyto komponenty spolupracují. Při práci s návrhářem formulářů generuje *Delphi* zdrojový kód komponent, jejichž instance jsou vývojářem umístěny na navrhovaném formuláři. *Delphi* má systém komponent hluboce implementován a pro snadné nastavení vlastností je k dispozici Inspektor objektů (*Object Inspector*). V editoru kódu lze doplnit dodatečný kód, který definuje požadované chování aplikace. Ve stejném editoru je také možné vznikající aplikaci ladit. K tomuto účelu slouží například zarážky kódu (*breakpoints*), funkce sledování obsahu proměnných apod. Dále editor obsahuje funkce zvýraznění syntaxe zdrojových textů. Integrovaný *debugger* a prostředí pro ladění programu usnadňuje hledání chyb v programech.

První verze byla vydaná v roce 1995 a od verze 2 byl kompilátor připraven pro *32 bitové Windows*. Po delší odmlce, během níž došlo ke stagnaci a zastarávání prostředí, byla vydána nová verze s názvem *Delphi 2010* podporující nové moderní technologie. Informace byly čerpány z publikací: *Mistrovství v Delphi 6 1.23.2.1*a *Mistrovství v Delphi 2 – pro Windows 95/NT 1.23.2.1.*

1.10 MATLAB Simulink

MATLAB Simulink je nadstavbou profesionálního výpočetního nástroje *MATLAB* pro návrh a simulaci blokových schémat od firmy *The MathWorks*. Pro stavbu schémat podporuje prostředí rozsáhlou množinu bloků z širokého množství oborů.

Je obzvláště vhodný pro úlohy řízení, ale prostředí je možné použít i pro mnoho dalších úloh. Zadání výpočtu je realizováno sestavením funkčních bloků v simulačním schématu. Toto schéma je numericky vypočteno s definovaným časovým krokem. Časový krok je nutný pro funkci některých bloků, mezi které patří například *Integrátor*. Prostředí obsahuje široké spektrum funkčních bloků, jež zastupuje různé obory numerické matematiky. Simulační prostředí je silně provázané s nástrojem *MATLAB*.

Schéma je uloženo v souboru zapsaném v kódování *ASCII*. V souboru jsou uloženy informace o nastavení systému, o blocích a o jejich propojení. Do souboru jsou zapsány pouze ty parametry bloku, které se liší od defaultního nastavení. Pramenem pro popis prostředí jsou webové stránky výrobce programu *1.23.2.1.*

1.11 PLC

PLC (*Programmable Logic Controller*) je programovatelný automat, který se využívá k číslicovému řízení procesů v průmyslovém prostředí. Pro řízení procesů obsahuje zařízení vstupy a výstupy. Mezi tyto procesy patří například řízení nejrůznějších manipulátorů a výrobních robotů. *PLC* jsou často realizovaná jako kompaktní zařízení pro malé, větší ale i rozsáhlé aplikace. Pro řízení rozsáhlých aplikací je *PLC* možné spojovat do komunikujících sítí. Zařízení je vyrobeno s důrazem na spolehlivost a je vybaveno řadou speciálních obvodů, jako jsou například obvody reálného času nebo obvody sledující, zda nedošlo k zacyklení běhu *PLC*.

Program *PLC* je zapsán jako posloupnost instrukcí v paměti uživatelského programu. Jednotlivé instrukce jsou postupně vykonávány v zápisníkové paměti. *PLC* má zápisníkovou paměť rozdělenou na čtyři části. První je registr X určený pro obraz vstupních portů. Druhou částí je registr Y, který obsahuje obraz výstupních portů. Tyto obrazy jsou aktualizovány pouze jednou v cyklu *PLC* (v části režie), což odstraňuje možnost vzniku hazardních stavů. Třetí částí je systémový registr a poslední částí je uživatelský registr.

PLC je možné pro nahrání programu nebo pro servisní zásahy připojit k PC pomocí sériového portu. Některé jednotky lze připojit také pomocí *USB* a *Ethernet*. Tato rozhraní slouží *PLC* i k vzájemné komunikaci při řízení složitějších procesů.

Program je uzpůsoben pro spolupráci s *PLC* firmy *TECO* a.s. Firma *TECO* a.s. je přední český výrobce průmyslových řídicích systémů kategorie *PLC*. Komunikace probíhá pomocí protokolu *EPSNET* po sériovém rozhraní. Informace čerpány z webových stránek *TECO a.s.1.23.2.1*.

1.12 Protokol EPSNET

Protokol slouží pro komunikaci po sériovém nebo *ethernetovém* rozhraní. Protokol umožňuje výměnu dat a případné řízení běhu *PLC*. Pracuje v režimu *master-slave*, kde *PLC* čeká, a odpovídá na dotazy. Protokol obsahuje popis dotazovacích a potvrzovacích zpráv pro komunikaci. Konfigurace sítě může být buď *monomaster*, což znamená, že na síti se nachází pouze jedna stanice typu master a jedna nebo více stanic pracujících v režimu *slave*. Komunikace může probíhat i v režimu *multimaster*. Výhodou protokolu je snadná implementace. Podkladová data pro tuto část jsou převzata z uživatelské příručky *Sériová komunikace programovatelných automatů TECOMAT1.23.2.1.*

1.13 Prostředí Mosaic

Mosaic je vývojové prostředí pro programování řídicích systémů. Prostředí bylo navrženo firmou *TECO* a.s. pro programování firmou vyráběných *PLC* pod platformou Windows. Prostředí umožňuje vytvářet řídicí program několika odlišnými způsoby, a tak zvolit pro daný problém nejvhodnější postup. Program *Mosaic* je dodáván od roku 2000. Prostředí je vyvíjeno ve shodě s mezinárodní normou *IEC EN-61131-3*, která definuje strukturu programů a programovací jazyky pro *PLC*. Zdrojem informací pro tuto kapitolu je uživatelská příručka *Začínáme v prostředí MOSAIC1.23.2.1*.

1.14 Použité formáty souborů

V následující kapitole jsou popsány podporované formáty souborů pro vstup a výstup dat z programu. Nastíněny jsou také nástroje a programy umožňující vytváření těchto datových struktur. Textové soubory jsou doplněny o příklad přibližující jejich syntaxi.

1.14.1 Formát souboru ASE

Formát *ASE* slouží pro uložení dat 3D modelu. Do tohoto formátu lze exportovat z mnoha komerčních i volně stažitelných grafických editorů a modelačních programů. Například z prostředí *Studio 3D MAX*, *Blender* a nebo *Cinema 4D*. Datové struktury jsou přehledně navrženy a zapsány v kódování *ASCII*, takže uložený soubor si lze snadno prohlédnout (např. v poznámkovém bloku).

V souboru jsou v první části definované materiály včetně odkazu na soubor s případnou texturou, barvy a dalších vlastností materiálů jako jsou údaje o nastavení textury definující počet jejích opakování a počáteční posunutí textury.

Následuje popis geometrie scény. Scéna je rozdělena na nižší celky s názvem *GeoObjekty*, ty jsou dále rozděleny na jednotlivé *Meshe*. Jednotlivé *Meshe* jsou popsané pomocí pole *vertexu* reprezentovaného třemi reálnými čísly a pole trojúhelníku definující propojení *vertexu* pomoci 3 indexů *vertexu*. Obdobně jsou definovány *texturové koordináty* pro mapování textur a normálové *vertexy* doplněné o normály ploch jednotlivých trojúhelníků.

1.14.2 Formát Souborů Targa (*.TGA)

Targa, označovaný také jako *TGA,* je formát určený pro popis rastrové grafiky. Byl navržen firmou *Truevision*.

Data grafiky mohou být zapsána třemi rozdílnými způsoby. V prvním je grafika popsaná omezenou barevnou paletou a každé barvě je přiřazen její index. Druhým způsobem je uložení obrázku ve stupních šedi. Posledním způsobem uložení dat je přímá definice barvy pixel. Pro každou ze tří základních barev je použito 8 bitů. Může být také použit *alpha* kanál představující hodnotu průhlednosti *pixelu*. *Alpha* kanál může být pouze jednobitová maska popisující průhledné oblasti obrázku anebo 8 bitový, který umožňuje použít různé stupně průhlednosti. Formát umožňuje ukládat data komprimovaná i nekomprimovaná.

Informace jsou v souborech typu *TGA* rozděleny do čtyř sekcí, přičemž pouze první sekce je povinná, ostatní sekce mohou či nemusí být použity. V první sekci umístěné na začátku souboru je uložena informační hlavička o velikosti rovné 18 bytům. V hlavičce souboru jsou informace o použitém formátu a rozlišení obrazu. Za informační hlavičkou může následovat identifikační pole obrázku, což je textový řetězec o maximální délce 255 znaků. Tato sekce je však nepovinná a málokdy se s ní v obrazových souborech setkáváme. Ve třetí sekci může být uložena paleta barev. Používá se pouze u některých obrázků s formátem 8 bitů na pixel. V poslední sekci jsou uložena vlastní rastrová data představovaná barvami jednotlivých *pixelů*. Posloupnost rastrových dat lze ve formátu *TGA* specifikovat přímo v hlavičce, je například možné obrázky ukládat od prvního řádku do řádku posledního či naopak. Rastrová data mohou být komprimována jednoduchým *RLE* algoritmem.

Díky své jednoduchosti a širokým možnostem se grafický formát *TGA* značně rozšířil a byl použit v mnoha aplikacích.

1.14.3 Soubor schémat MATLAB Simulink (*.MDL)

Soubor schémat výpočetního prostředí *MATLAB Simulink* je textový soubor, ve kterém je uloženo popisované schéma. Soubor ve své úvodní části poskytuje mnoho informací o nastavení systému. V druhé části jsou zde popsány údaje pro definici uložených bloků. Zapisovány jsou pouze změněné parametry od základního nastavení. V koncové části souboru jsou popsány vodiče informací nazvané *Line*, které propojují jednotlivé bloky.

1.14.4 Soubor VM projektu (*.VMP)

Soubor VM projektu je hlavní soubor projektu. Soubor obsahuje informace se jménem autora, datum vytvoření a název souboru hlavního schématu modelu. Hlavní schéma stojí nejvýše v simulačním stromu.

Soubor tvoří zapouzdření celého projektu. Stojí nejvýše v hierarchii vytvářeného projektu. Cesta k hlavnímu simulačnímu schématu, pokud je to možné, je zapsaná relativně od hlavního souboru projektu. Příklad syntaxe souboru je na obrázku 1.

```
Projekt {
  Info {
         Name "Projekt1"
         Autor "Miroslav vavrousek"
        }

         MainSchema "Projekt1_Data\SchemaA.vms"
}
```

1. Obrázek: Ukázka syntaxe

1.14.5 Soubor VM schématu (*.VMS)

Soubor popisuje *VMSchema*. Celý obsah souboru je zapsán v blocích s identifikátory *System* a *Model*. Bloky jsou ohraničeny otevíracími a uzavíracími složenými závorkami. Syntaxe souboru *VMSchema* byla navrhovaná s ohledem na zvýšení kompatibility se soubory *MATLAB Simulink*. Pro shodné vlastnosti mají bloky stejné identifikátory. Shodné jsou také principy zápisu *Line* a jejího větvení, včetně jejich grafické reprezentace do souboru.

V první části souboru jsou zapsané použité bloky. Každý blok je zapsán do sekce označené identifikátorem *Block* a otevírací složenou závorkou. Následují jednotlivé parametry bloku. Každý parametr se nachází na vlastním řádku a je uveden svým identifikátorem. Nejprve blok obsahuje čtyři povinné parametry. Těmito parametry je typ bloku (*identifikátor BlockType*), jméno bloku (*identifikátor Name*), třetím povinným parametrem je počet vstupních a výstupních portů zapsaných v hranatých závorkách (*identifikátor Ports*). Posledním povinným parametrem je parametr uvedený identifikátorem *Position* obsahující čtyři celá čísla definující obdélník pro zobrazení bloku při návrhu schématu.

V druhé části souboru jsou zapsány spojnice bloků, označované jako *Line*. Každá je uvedena identifikátorem *Line* a otevírací složenou závorkou. Následuje jméno zdrojového objektu uvedené identifikátorem *SrcBlock* a identifikátorem *SrcPort*

označujícím index výstupního portu bloku. Pokud není *Line* rozvětvená, je identifikátory *DstBlock* a *DstPort* definován cílový blok a jeho port. Pokud má *Line* více cílových bloků, znamená to, že je *Line* rozvětvená a jsou vytvořeny dva vnořené bloky s identifikátory *Branch*. Blok obsahuje definici bodu pro grafické znázornění *Line* uvedené identifikátorem *Points* a případně definici cílového bloku a portu nebo dalšího rozvětvení. To je opět reprezentováno dvěma bloky *Branch*.

Odkazy na další soubory s modely, tabulkami dat a vnořenými simulačními schématy, jsou, pokud to jejich umístění v systému souborů dovoluje, zapsány relativní cestou od souboru *VMSchema*. Na obrázku 2 je příklad souboru *VMSchema* zachycující schéma s jedním vstupním a jedním výstupním portem propojeným vodičem informací.

```
Model {
  System {
    Block {
      BlockType         InPort
      Name              "InPort1"
      Ports             [0, 1]
      Position          [458, 331, 508, 381]
    }
    Block {
      BlockType         OutPort
      Name              "OutPort1"
      Ports             [1, 0]
      Position          [579, 327, 629, 377]
    }
      Line {
        SrcBlock                InPort1
        SrcPort         1
        Points          [27, 0; 0, -4]
        DstBlock                OutPort1
        DstPort         1
      }
  }
}
```

2. Obrázek: VMSchéma

1.14.6 Soubor s tabulkou dat (*.TXT)

Jedná se o textový soubor s hodnotami oddělenými tabulátorem. Tento soubor slouží pro možnosti importu a exportu dat. S tímto typem souboru spolupracuje například *MS Excel* a tabulkové procesory. V programu je použit pro tabulku dat v bloku *VMTrajektorie* a pro export vypočtených dat. Na obrázku 3 je ukázka souboru s pěti sloupci hodnot. Hodnoty jsou oddělené tabulátory.

```
4       6       8       4       5
5       8       7       6       8
7       10      5       5       8
8       5       7       4       7
```

3. Obrázek: Ukázka souboru dat

1.15 Základní stavební bloky

Následuje výčet bloků se stručným popisem jejich funkce zpětně kompatibilních s programem *MATLAB Simulink*. Importovaná schémata z programu *MATLAB Simulink* musí být složená pouze z této množiny bloků, jinak program odmítne schéma importovat s odůvodněním, že schéma obsahuje neznámý blok.

In port

Blok má jediný výstup a slouží pro vstup hodnot z nadřazeného funkčního celku *(VMSchema)*.

Out port

Blok typu *Out port* slouží pro výstup hodnot z nadřazeného funkčního celku *(VMSchema)*.

Constant

Blok slouží pro zavedení konstant v simulačním schématu. Má jediný výstup, na kterém je hodnota uvedená v nastavení bloku.

Logical operator

Blok realizuje jednu z logických funkcí *NOT*, *AND* nebo *OR* na přivedených vstupech. Výsledek operace je uložen na jediný výstupní port. Při vyhodnocování je použita pozitivní logika.

Relational operator

Blok obsahuje dva vstupní porty, mezi kterými provádí porovnávací funkci podle nastaveného operátora. Výstupem bloku je booleovská proměnná. Hodnota jedna je na výstupu, pokud daná podmínka platí, a nulová, pokud neplatí.

Switch

Blok obsahuje tři vstupní porty a jeden výstupní. Slouží k přepínání mezi dvěma vstupy pomocí hodnoty přivedené na prostřední port. Pokud hodnota přivedená na prostřední vstup splňuje nastavenou podmínku, je na výstup přivedena hodnota druhého vstupu. Pokud podmínka neplatí, je na výstup přivedena hodnota prvního vstupu. Podmínka je definována hodnotou proměnné *Threshold* a operátorem.

Rounding

Blok provede na jediném vstupním portu vybranou zaokrouhlovací funkci a výsledek uloží na výstup. Lze zvolit jednu ze čtyř zaokrouhlovacích funkcí. První funkcí

je *floor*. Tato funkce zaokrouhlí reálné číslo dolů k menší absolutní hodnotě čísla. Druhá funkce je *ceil*. Na rozdíl od první funkce zaokrouhluje nahoru k nejbližší vyšší celé absolutní hodnotě. Třetí z funkcí je funkce *round*. Zaokrouhluje číslo obvyklým matematickým způsobem. Pokud je desetinná část čísla v absolutní hodnotě menší než 0.5, je číslo zaokrouhleno dolů. Poslední funkcí je funkce *fix*. Tato funkce přivede na výstup pouze celou část čísla na vstupu. V prostředí *Delphi* je reprezentována funkcí *Trunc*.

Sum

Tento typ bloku umožňuje sčítání a odčítaní. Každému vstupu je možné nastavit kladné nebo záporné znaménko. Všechny vstupy jsou poté s příslušným znaménkem sečteny a uloženy na výstup.

Product

Blok slouží k realizaci násobení. Na svůj výstup zapíše výsledek součinů všech svých vstupů.

Reference

Blok porovná vstup se zadanou konstantou a nastaveným operátorem. Pokud je daná podmínka splněna, je na výstup uložena hodnota 1, v opačném případě hodnota 0.

Discrete pulse generator

Blok má jediný výstup, na kterém generuje obdélníkový signál o nastavené amplitudě a se zadanou periodou. Lze také nastavit šířku pulsu, která se nastavuje v procentech periody.

Integrator

Blok provádí numerickou integraci vstupu lichoběžníkovou metodou. Lze nastavit vnitřní počáteční podmínku (*Initial Condition*) nebo použít počáteční podmínku externí přivedenou na příslušný vstup integrátoru. Lze použít i externí reset.

Gain

Blok vynásobí vstup konstantou uloženou v bloku a přivede ji na výstup.

Random number

Blok vygeneruje v každém kroku nové náhodné celé číslo.

MinMax

Blok umožňuje vybrat, zda hledat maximum či minimum. Podle zvolené funkce prověří všechny vstupy a na výstup předá minimální či maximální hodnotu vstupu.

Scope

Blok má pouze jeden vstupní port. Ukládá hodnoty z tohoto portu a vytváří tak vektor dat pro další zpracování.

1.16 Rozšiřující bloky

V následující kapitole jsou popsány čtyři bloky, které nemají ekvivalent v programu *MATLAB Simulink* a rozšiřují stavbu schématu o nové důležité funkce a vlastnosti.

1.16.1 VMObjekt

Blok *VMObjekt* slouží k realizaci vizualizace. Má přidělený model z knihovny modelů, který zobrazuje na určenou pozici a s definovanou rotací. V bloku lze definovat počáteční podmínky pro šest veličin, se kterými blok pracuje. Jsou to veličiny poloha, rychlost, zrychlení, rotace, rychlost rotace a zrychlení rotace.

Blok obsahuje pouze vstupní porty. Podle nastavení to mohou být porty pro definování všech šesti veličin. Pro souběžné definování závislých veličin jako jsou například poloha, rychlost a zrychlení, obsahuje ještě navíc *select* porty pro integračně vyšší veličiny. Aktivací *select* portu se rozumí přivedení nenulové hodnoty. Například pokud bude blok zvolený se vstupními porty poloha a rychlost a bude-li aktivován *select* port pro polohu, nebude na hodnoty přivedené na porty veličiny rychlosti brán zřetel. Jestliže nebude *select* port aktivován, znamená to, že na něj bude přivedena nulová hodnota, vypočte se poloha modelu z definované rychlosti a původní polohy modelu. Takto to analogicky funguje pro všechny veličiny. Díky tomu lze snadno definovat složitější modelované situace.

1.16.2 VMTrajektorie

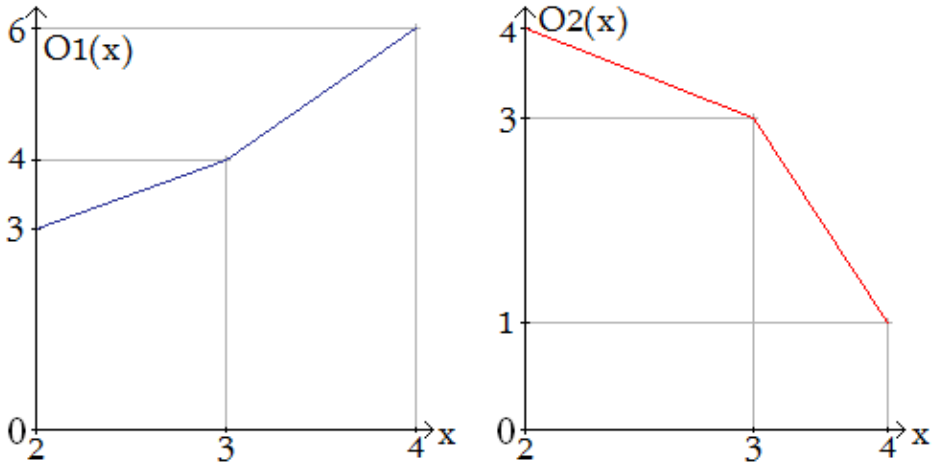
Blok má jeden vstupní port a nejméně jeden výstupní port. Výpočet hodnot na výstupních portech je proveden interpolací hodnot tabulky. První sloupec v tabulce je retenční. Další obsahují data pro interpolaci. Nastavením bloku lze zvolit soubor, ze kterého je načtena tabulka hodnot a je automaticky nastaven počet portů o jednu menší, než je počet sloupců tabulky. Z toho plyne, že tabulka musí mít nejméně 2 sloupce. Následuje ukázkový příklad nastiňující funkci bloku pro referenční hodnotu 2,5. Tabulka dat má tři sloupce a je zobrazena na obrázku 4 včetně hlavičky popisující jednotlivé sloupce jako referenční hodnoty a hodnoty první a druhé funkce v závislosti na referenční hodnotě.

Tabulka dat

x	O1(x)	O2(x)
2	3	4
3	4	3
4	6	1

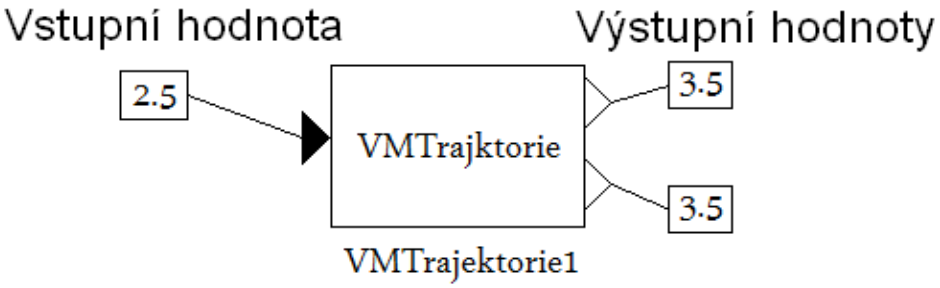
4. Obrázek: Tabulka

Na obrázku 5 jsou funkce O1(x) a O2(x) zobrazeny ve formě grafů včetně úseček vytvořených interpolací mezi definovanými hodnotami.



5. Obrázek: Grafy funkce O1(x) a O2(x)

Schematické znázornění funkce bloku včetně znázornění vstupní hodnoty a výstupních hodnot je na obrázku 6.



6. Obrázek: Schematické znázornění funkce bloku

1.16.3 VMPLC

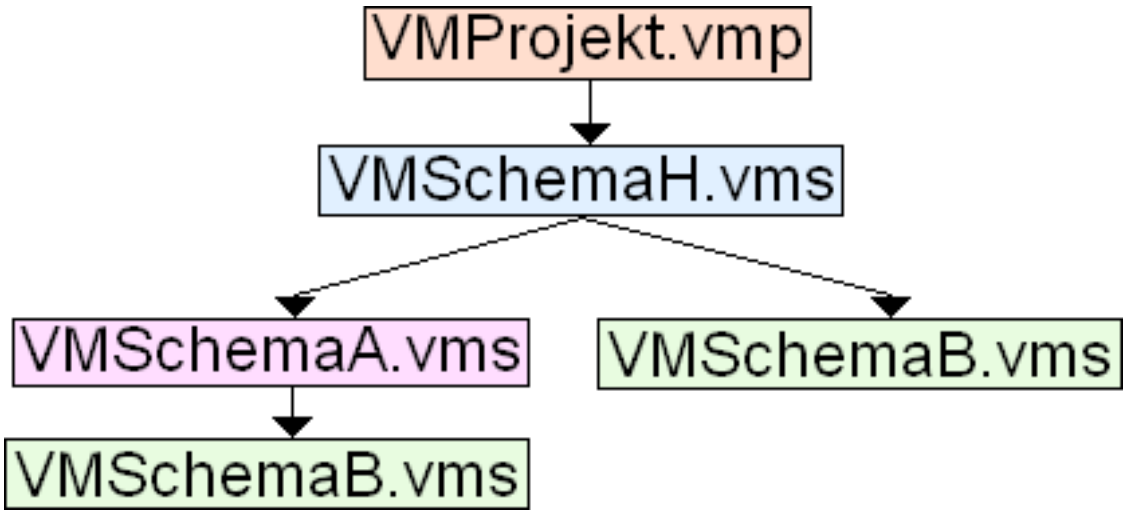
Blok realizuje komunikaci s *PLC* a jeho napojení do simulačního systému. Vstupní a výstupní porty bloku jsou napojeny na porty *PLC*. Komunikace probíhá po sériové lince *RS-232* pomocí protokolu *EPSNET*. V každém kroku jsou zapsány údaje na vstupní porty *PLC* a přečteny hodnoty z výstupních portů *PLC*. Práce s *PLC* je možná pouze v režimu simulace v reálném čase. V režimu uživatelsky definovaného kroku simulace je na všech výstupních portech zapsaná hodnota 0.

1.16.4 VMSchéma

Blok *VMSchema* umožňuje vložit schéma pro realizaci vnořené funkce. Poskytuje tím možnost stavby schémat z vyšších funkčních celků. To dovoluje snadnější opakované použití částí schémat a zvyšuje přehlednost schématu. Další výhodou je možnost tvorby knihoven rozvíjející funkce prostředí. Vstup a výstup dat je realizován pomocí bloku *InPort* a *OutPort*. Blok si po načtení automaticky upraví počet vstupních a výstupních portů. *VMSchéma* umožňuje importovat schéma z programu *MATLAB Simulink* sestavené z podporovaných bloků. umožnit

1.17 Strom projektu

Nejvýše ve stromu se nachází projekt reprezentovaný souborem projektu *(*.vmp)* a ten se odkazuje právě na jedno hlavní simulační schéma (*VMSchema*) reprezentované souborem *(*.vms).* Tato dvojice tvoří nejjednodušší projekt. Hlavní schéma může mít dále vnořena další schémata pomocí bloků *VMSchema*. Vnořená schémata mohou realizovat různé elementární funkce virtuálního modelu. Tato schémata mohou být složena ze základních bloků a mohou obsahovat další bloky typu *VMSchema*. Každé schéma je reprezentováno vlastním souborem. Na obrázku 7 je znázorněn strom hierarchie projektu reprezentovaný soubory projektu. Příklad je tvořen souborem projektu a hlavním schématem. Hlavní schéma obsahuje dvě vnořená schémata. První vnořené schéma obsahuje jako jeden ze stavebních bloků druhé vnořené schéma. Při návrhu projektu je nutné zajistit, aby hierarchie měla tvar stromu. To znamená, aby první schéma neobsahovalo schéma druhé a naopak. Takový simulační model by byl neřešitelný.



7. Obrázek: Strom projektu

Praktická část

Praktická část navazuje na semestrální magisterský projekt s názvem „*Tvorba nástroje pro návrh obecných virtuálních modelů*“*1.23.2.1.* Projekt se zabýval vývojem aplikace nazvané *VirMo Editor*, která umožňuje sestavovat simulační schéma a nastavovat blokům zvolené parametry. Tato schémata je možné uložit do souboru.

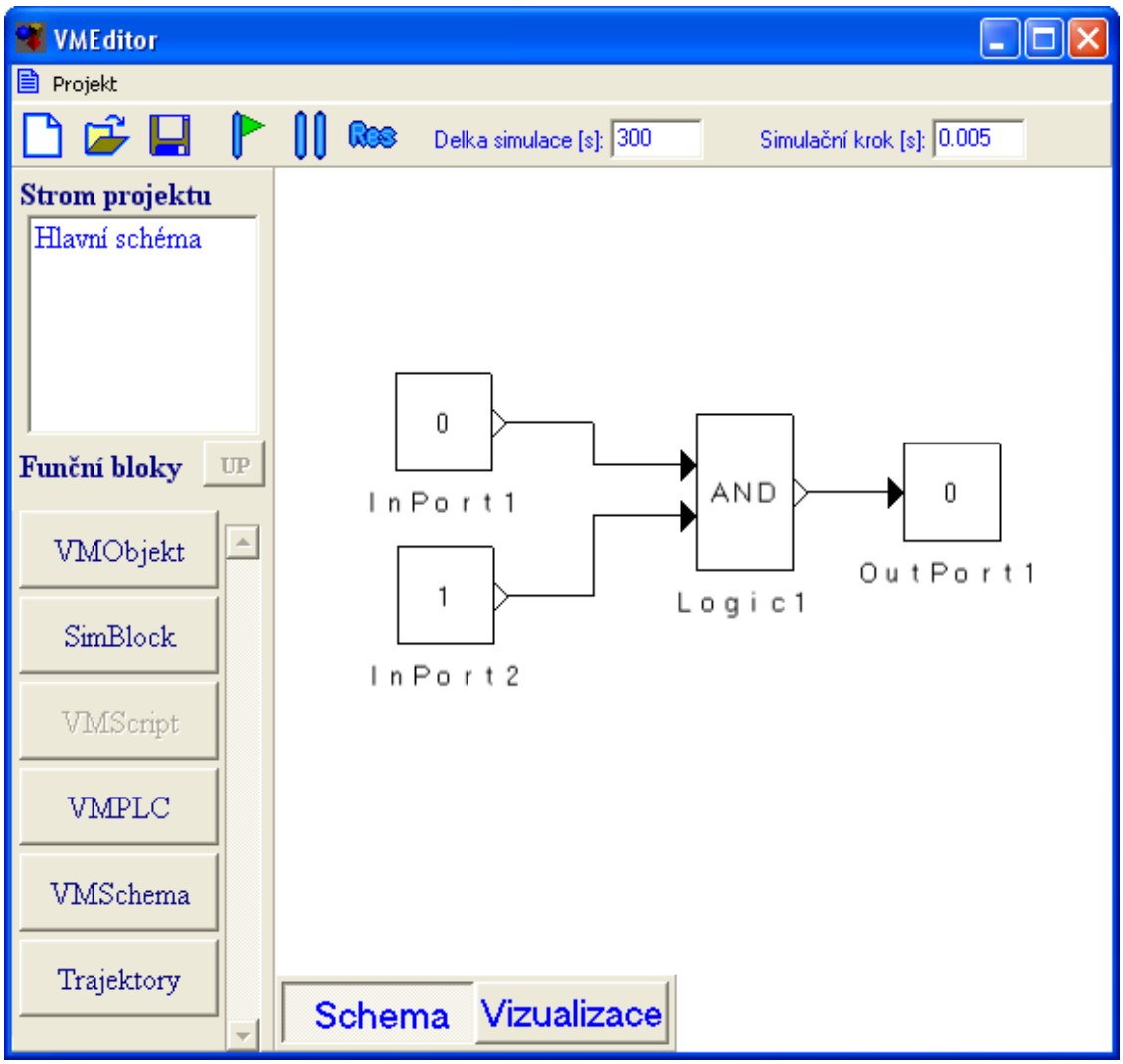
V diplomové práci se zabývám tvorbou aplikace pro simulaci a vizualizaci blokových schémat dále nazvaný jako program *VirMo Viewer*. Pro tvorbu aplikace bylo použito vývojové prostředí *Visual Studio*. Program byl vytvořen jako aplikace s grafickým uživatelským rozhraním. K tvorbě grafického rozhraní bylo použito grafické návrhové prostředí *Windows Form*.

Jelikož při práci na *VirMo Viewer* bylo nutné mírně upravit program *VirMo Editor,* v první časti je stručný popis programu a informace o provedených úpravách na programu *VirMo Editor*.

1.18 VirMo Editor

VirMo Editor je program pro stavbu blokových schémat vytvořený v prostředí *Delphi*. Program umožňuje navrhnout schéma, zkontrolovat, zda je výpočet navrženého schématu možné dokončit a poskytuje náhled vizualizace. Neobsahuje nástroje pro komunikaci s *PLC.* Pokud sestavené *VMSchema* obsahuje blok *VMPLC,* na všechny jeho výstupní porty je přivedena hodnota 0.

Program se také stará o správu souboru projektu. Pokud je vybrán soubor, který se nenachází v pracovním adresáři s daty modelu, je uživatel při ukládání projektu upozorněn a je mu nabídnuta možnost překopírovat soubor do adresáře s daty. Je možné určit volbu ponechání či překopírování pro všechny další soubory pomocí volby *Yes all* a *No all*. Soubor, který není uložen v pracovním adresáři, může narušit konzistenci modelu a jeho přenositelnost, protože cesta k němu je zapsaná absolutně. Také přibyl dotaz při uložení úprav vnořeného schématu. Uživatel je dotázán, zda se změny mají použít ve všech blocích osahující editované schéma, nebo zda má být schéma uloženo pod novým jménem. Rozšířeno a upraveno je i uživatelské rozhraní. Na levém bočním panelu s funkčními bloky přibyl strom projektu umožňující přesunout se z vnořeného schématu výše do některého z nadřazených schémat. Přibyla možnost vložit *blok VMPLC* a *VMTrajektorie*. Prostředí programu je zachyceno na obrázku 8.



8. Obrázek: Prostředí programu

1.19 VirMo Viewer

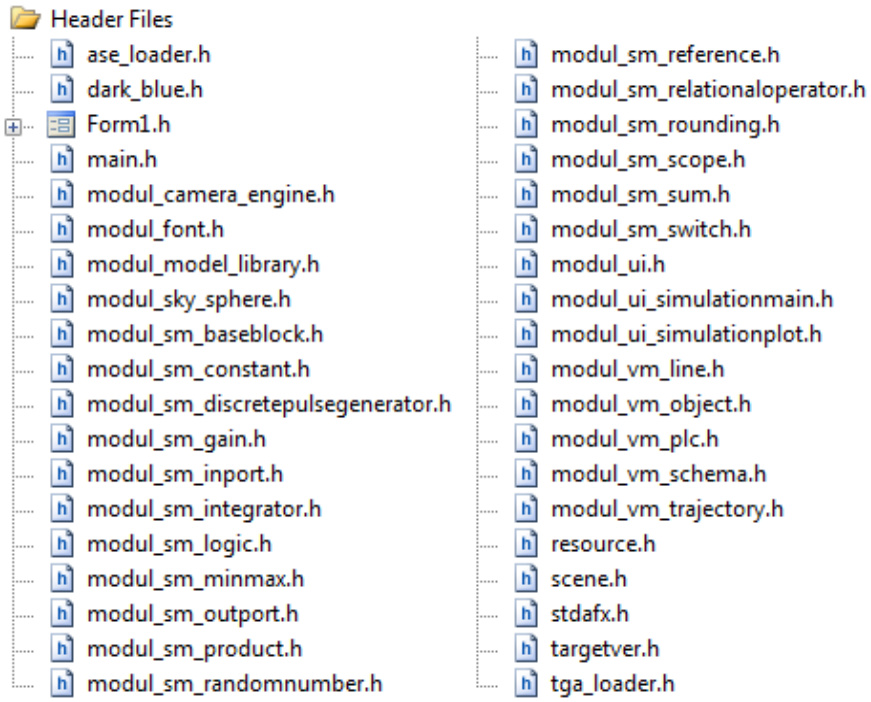
Aplikace má možnost otevřít soubor *VM projektu*, provést simulaci sestaveného schématu a výsledky lze vizualizovat pomocí 3D modelu. Parametry simulace lze číst a upravovat pomocí *PLC*. Prohlížeč umožňuje zobrazit grafy vypočtených hodnot v čase a to jak ve 2D, tak 3D zobrazení. V 2D je vybraná veličina ve zvolené ose funkcí času. V 3D je to tříprvkový vektor v závislosti na čase. Zobrazené grafy je možné exportovat jako obrázek a vypočtené hodnoty lze exportovat do textového formátu. Tento formát je podporovaný např. *MS Excelem*. Aplikace pracuje ve dvou základních režimech. Jsou to „Simulace se zvoleným krokem“ a „Simulace v reálném čase“.

1.20 Realizace a struktura programu

Program je napsán v jazyce *C++* ve vývojovém prostředí *MS Visual Studio* jako aplikace s vizuálním uživatelským rozhraním. Uživatelské rozhraní se skládá z hlavního okna a okna pro nastavení komunikace s *PLC*.

1.20.1 Uspořádání programu

Program je rozdělen do řady modulů v hiearchistické struktuře. Jednotlivé moduly realizují základní funkce jako například knihovnu modelů. Další moduly realizují funkce načítání souborů, funkce pro grafické rozhraní a vizualizaci výsledků. Pro každý blok je vytvořen vlastní modul realizující jeho funkci. Program využívá knihovny pro *OpenGL* a standardní knihovny obsažené ve vývojovém prostředí včetně knihovny *vector* pro dynamická pole. Přehled hlavičkových souborů je na obrázku 9.



9. Obrázek: Hlavičkové soubory

1.20.1.1 Hlavičkový soubor Form1.h

Obsahuje hlavní třídu *Form1* odvozenou od *System::Windows::Forms::Form*, která obsahuje standardní vlastnosti pro práci s formuláři. Tuto třídu dále rozšiřuje o vlastnosti a funkce definující chování vytvářené aplikace.

Třída *Form1* obsahuje deklaraci jednotlivých komponent, které vytváří grafické uživatelské rozhraní. V tomto souboru se také nachází popis rozmístění, velikostí a dalších

vlastností komponent. Je zde implementován *konstruktor třídy*, který vytváří další potřebné objekty a nastavuje některé počáteční podmínky, destruktor třídy a události jednotlivých komponent. Těmito událostmi může být například kliknutí na tlačítko nebo stisk klávesy. Jsou zde také popsány funkce pro nastavení vykreslování *OpenGL*.

Třída má deklarován objekt typu *Scene* a přilinkován soubor *scene.h*. Přes tento objekt probíhá veškerá další práce s jádrem programu.

Hlavní funkce *main* se nachází v souboru *form.cpp*. Jsou v ní povoleny vizuální styly. Poté je vytvořen a spuštěn hlavní formulář aplikace.

1.20.1.2 Hlavičkový soubor scene.h

Hlavičkový soubor *scene.h* reprezentuje nejvyšší rozhraní v projektu. Spojuje dohromady část pro realizaci vizualizace a část pro simulační činnost programu.

V souboru se nachází deklarace třídy *TSscene*, která je zastřešující funkcí jádra programu a nachází se v ní deklarace všech důležitých objektů. Jsou zde deklarovány proměnné pro definici současné polohy ve vizualizované scéně a proměnná pro uložení zvoleného režimu programu. Dále obsahuje deklarace funkce pro zprávu stisknutí tlačítka klávesnice a funkce pro události myši. Mezi události myši patří *MouseDown* pro stisk tlačítka myši, *MouseUp* pro uvolnění tlačítka myši a událost *MouseMove* volaná při pohybu myši.

Dále se v souboru nachází třída obsahující údaje pro práci s reálným časem nazvaná *FPS*, která mimo jiné slouží pro uložení počtu vykreslených snímků za sekundu a pro další pomocné parametry.

Všechny funkce mají svoji implementaci v souboru *scene.cpp*.

1.20.1.3 Simulační jádro

Simulační jádro slouží pro výpočet blokových schémat. Obsahuje definici funkcí jednotlivých bloků.

Simulační jádro je deklarováno v hlavičkových souborech pojmenovaných podle typu s prvním prefixem modul a druhým *sm* pro standardní bloky nebo *vm* pro rozšiřující bloky. Rozšiřující bloky nejsou zpětně kompatibilní s *MATLAB Simulink*.

V programu je pevně definován blok *VMSchema* v souboru *scene.h*, který může obsahovat kterýkoli jiný blok. Pro toto hlavní schéma jsou volány všechny funkce řízení simulace, které se dále distribuují podle potřeby podřízeným blokům.

Všechny třídy bloků jsou odvozeny od třídy *modul_sm_baseblock*. Třída odděluje společné vlastnosti všech bloků jako je třeba knihovna portů nebo jméno a deklaruje virtuální funkce, které jsou definovány ve zdrojových souborech jednotlivých bloků podle potřeby.

Všechny hlavičkové soubory mají implementaci deklarovaných funkcí obsaženou v souborech shodného jména.

1.20.1.4 Vizualizační jádro

Vizualizační jádro se používá pro vykreslování vizualizací simulace a pro vykreslení grafů zkoumaných objektů. Skládá se ze tří částí, z nichž každá realizuje jinou oblast vizualizace.

První část deklarovaná hlavičkovým souborem *modul_model_library.h* slouží pro práci s 3D modely. Modul je reprezentován knihovnou modelů. Tyto modely lze snadno načíst ze souboru a vložit do knihovny pod zvoleným jménem. Jsou podporovány soubory typu *ASE*. Načítání realizují funkce definované v hlavičkovém souboru *ase_loader.h.* Zde jsou také definovány funkce pro kompilaci načtených dat do podoby *Display Listu*. Pro načítání dat z *TGA* textur používá program modul definovaný v hlavičkovém souboru *tga_loader.h*. Každému modelu je přiřazeno jedinečné jméno, pod kterým s ním lze dále pracovat. Knihovna modelů obsahuje také funkci pro vykreslení modelů na zadanou pozici se zvolenou rotací a další pomocné funkce pro práci s modely.

Druhá část vizualizačního jádra je reprezentována hlavičkovým souborem *modul_camera_engine.h*. V tomto souboru jsou definovány funkce pro práci s kamerou. Tyto funkce slouží pro vhodné umístění kamery ve scéně pomocí různých volacích parametrů. Těmito parametry mohou být ve vhodné kombinaci pozice kamery, pozice cíle nebo vertikální a horizontální úhly natočení kamery.

V třetí části je deklarovaná v hlavičkovém souboru nazvaném *modul_font.h* třída pro práci s textem. Třída obsahuje knihovnu fontů, v níž jsou uloženy modely jednotlivých znaků textu. Tyto modely lze vytvářet pomocí vestavěné funkce s fonty znaků podporovaných operačním systémem Windows pomocí jeho systémového jména. *Fontový engine* se chová jako stavový automat. Nastavení zobrazení textu jsou platná až do jejich změny. Ve třídě jsou definovány funkce pro nastavení velikosti, fontu a barvy vykreslovaného textu. Dále se zde nachází funkce pro vykreslení textu. První funkce nazvaná *DrawText3D* kreslí text v prostoru na zadané pozici a při zvolené rotaci. Druhá

funkce jménem *DrawText2D* je pro vykreslování textu na pozici definované přímo v *pixelech*.

1.20.2 Výpočet schématu

Schéma je vypočítáváno v simulačních krocích. V prvním výpočtu je nutné určit posloupnost výpočtů jednotlivých bloků. Tato posloupnost je pro urychlení výpočtu uložena. Nejprve jsou zpracovány bloky bez vstupních portů. Poté bloky typu *Integrator* rozvedou svou počáteční podmínku do navazujících bloků. Tento krok je nutný pro zpracování smyček s vloženými integrátory. Dále jsou cyklicky prohledány všechny *Line*. Jestliže má blok aktualizované všechny vstupní porty, je zpracován. Pokud je zdrojový blok *Line* vyhodnocen, jsou přeneseny hodnoty na cílové bloky.

Výpočet schématu lze dokončit, pokud mají všechny bloky kromě *VMPLC* zapojené všechny vstupní porty a schéma neobsahuje smyčky, v kterých se nenachází integrátory.

1.21 Režimy programu

Program umožňuje práci ve dvou režimech. V prvním režimu, nazvaném „Simulace se zvoleným krokem“, je pevně definovaná délka simulace a velikost zvoleného simulačního kroku. V druhém režimu, pojmenovaném „Simulace v reálném čase“, probíhá simulace cyklicky, což umožňuje komunikaci s *PLC* a simulační krok je nejmenší dosažitelný.

1.21.1 Režim Simulace se zvoleným krokem

V režimu „Simulace se zvoleným krokem“ je zadán simulační krok a délka simulace. Výpočet je spuštěn a celý nejprve přepočítán do paměti. Jelikož výpočet může být časově velmi náročný, je indikován ukazatelem stavu průběhu výpočtu.

Po dokončení výpočtu je možné vizualizaci plně ovládat pomocí panelu přehrávače. Přehrávač obsahuje funkce pro krokování simulace dopředu i vzad nebo lze simulaci plynule přehrát ve zvolené rychlosti, pozastavit, případně se pomocí posuvníku, či po simulačních krocích v čase simulace posouvat.

Stisknutím příslušného tlačítka lze program přepnout do okna pro sledování grafů zkoumaných veličin. Hodnoty jsou zachyceny v celé délce simulace a grafy lze posouvat, případně vytvářet výřezy.

1.21.2 Režim Simulace v reálném čase

Schéma je cyklicky počítáno. Simulace probíhá v reálném čase, proto je simulační krok minimální, možný v závislosti na rychlosti výpočetního systému, na kterém simulace probíhá. Průběh je možné pozastavit a restartovat do počátečních podmínek. Lze zobrazit grafy průběhů veličin, které se cyklicky doplňují ve dvou režimech. V prvním režimu se hodnoty uchovávají od startu simulace a v druhém režimu pouze zvolenou dobu.

V režimu „Simulace v reálném čase“ je možné realizovat komunikaci s *PLC*. Nejprve je nutné nastavit komunikační parametry a sériové parametry pro jednotlivé *PLC* bloky pomocí rozhraní programu. Poté je možné komunikaci zahájit. Schéma je počítáno cyklicky a při výpočtu bloku *VMPLC* jsou zapsány a přečteny požadované porty. Jedna sekunda reálného času je jedna sekunda v simulaci. Simulační krok je minimální možný vzhledem k výkonu výpočetního systému. Vizualizace je okamžitě vykreslována a celý proces se snaží maximálně využít možný výkon. Vyšší simulační kroky mohou vést k nestabilitě systému.

1.22 Uživatelské rozhraní

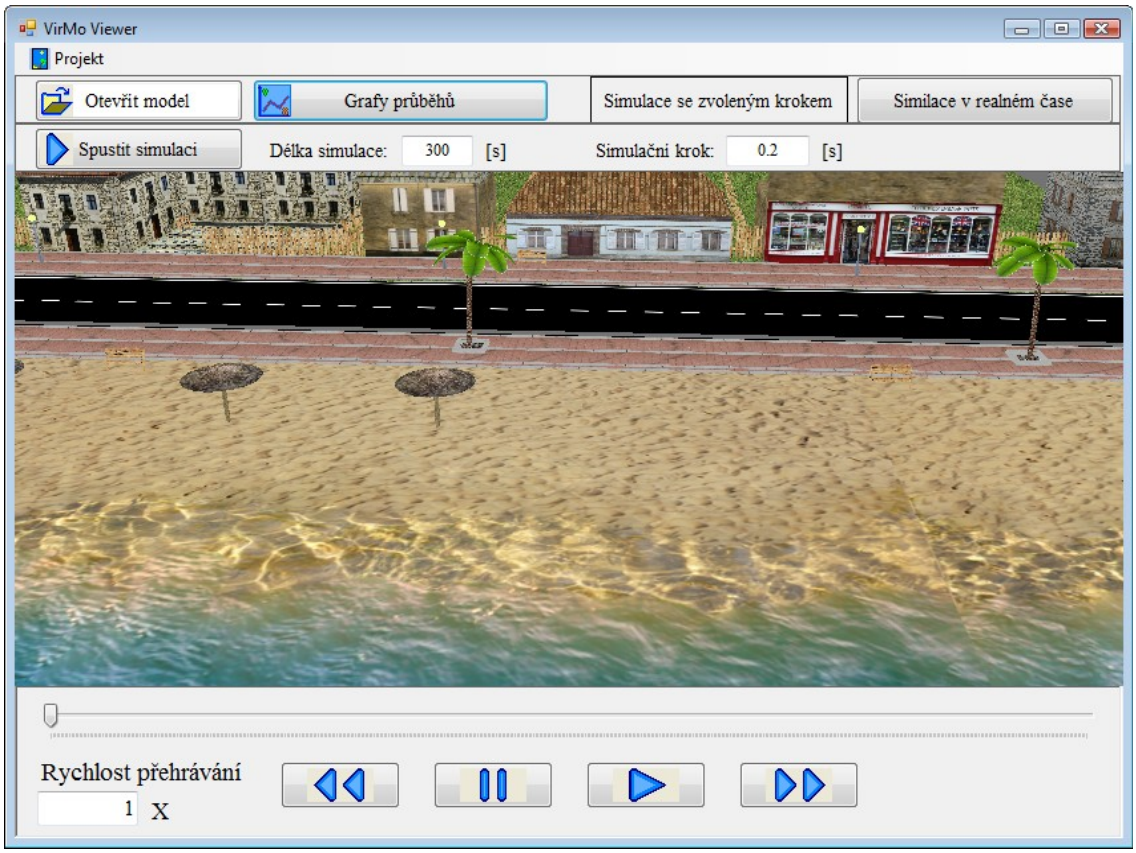
Tato část obsahuje popis práce s programem *VMView*, nastínění uživatelského rozhraní pro ovládání vizualizace a zpracování výsledků. Uživatelské rozhraní je tvořeno hlavním oknem, které mění svou podobu podle zvoleného režimu programu a okna pro nastavení komunikačních parametrů pro *PLC*. Následuje popis jednotlivých částí rozhraní.

1.22.1 Hlavní okno v režimu Simulace se zvoleným krokem

Hlavní okno je zobrazené na obrázku 10. V horní části se nalézá hlavní menu s nabídkou pro načtení projektu a ukončení programu.

Pod tímto menu se nachází lišty tlačítek. V první liště jsou umístěna tlačítka pro otevření projektu a pro volbu „Simulace se zvoleným krokem“ nebo „Simulace v reálném čase“. Dále je zde možnost pro přepnutí do *plotru* vykreslujícího grafy funkcí zvolených veličin. V režimu výpočtu „Simulace se zvoleným krokem“ se na druhé liště tlačítek zobrazuje tlačítko pro spuštění simulace. Textové pole slouží pro zadání délky simulace a simulačního kroku. Při zapnutí výpočtu modelu se ve středu obrazovky zobrazí panel s posuvným ukazatelem průběhu. Po vypočtení simulace se v dolní části obrazovky

objeví panel s posuvníkem průběhu simulace a tlačítka pro přehrávání simulace. V pořadí z leva doprava jsou to tlačítka pro přesunutí průběhu simulace o jeden krok zpět, tlačítko pro pozastavení simulace, spuštění simulace a posunutí simulace o krok dopředu. Na spodním panelu se nachází také textové pole pro nastavení rychlosti simulace.

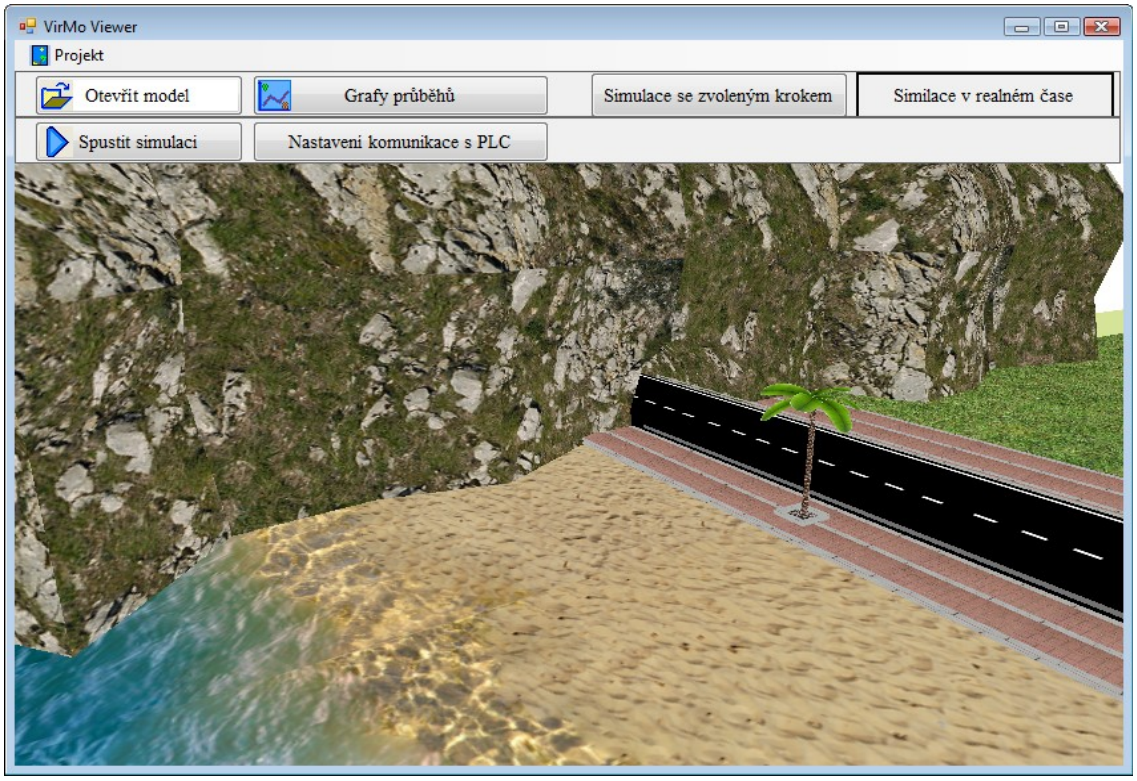


10. Obrázek: Okno simulace

1.22.2 Hlavní okno v režimu Simulace v reálném čase

Při běhu programu v režimu „Simulace v reálném čase“ je horní část okna shodná s horní částí okna v režimu „Simulace se zvoleným krokem“ a nachází se v ní tlačítka pro volbu režimu programu.

Okno se liší od druhé tlačítkové lišty, kde se nachází tlačítko pro spuštění a pozastavení simulace a tlačítko vyvolávající okno pro nastavení komunikačních parametrů s *PLC*. Zobrazení okna se nachází na obrázku 11.



11. Obrázek: Okno simulace v reálném čase

1.22.3 Pohyb ve vizualizaci

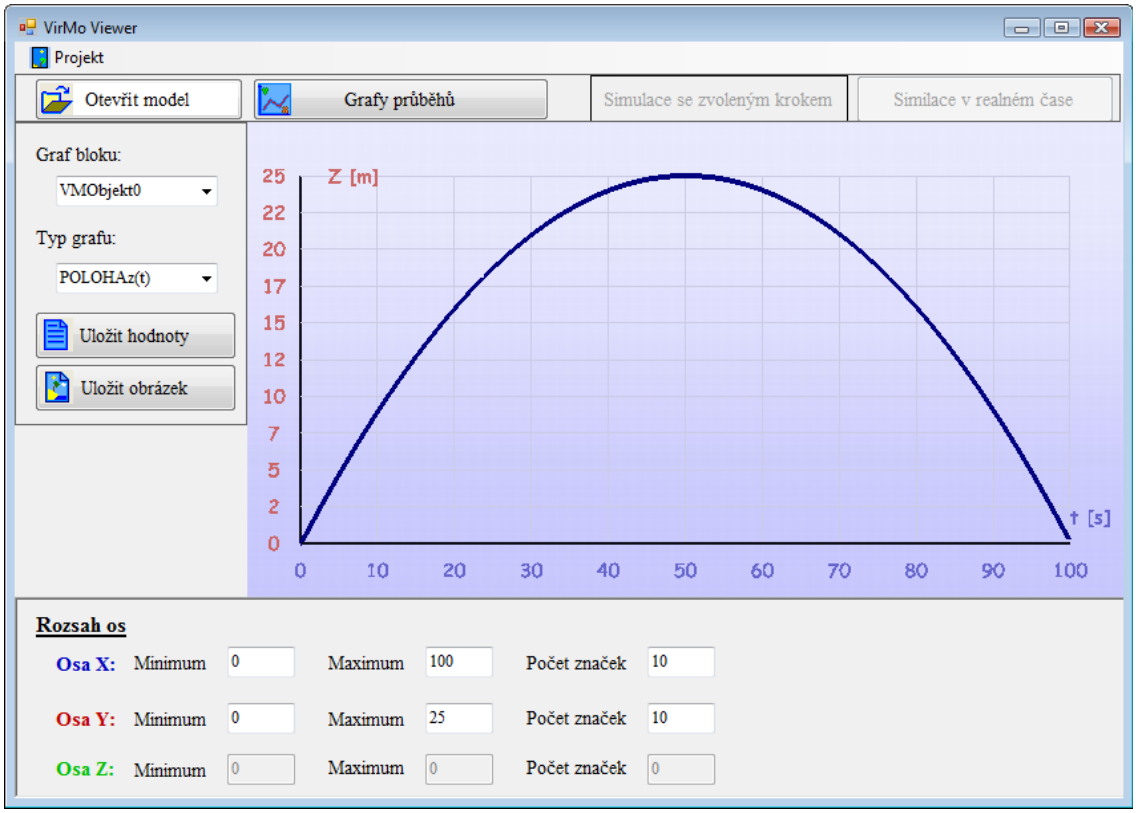
Pohyb v prostoru vizualizace je shodný pro oba režimy. Pohyb lze ovládat dvěma způsoby, které lze vzájemně kombinovat. Kameru lze posouvat pomocí kurzorových šipek v osách X a Y a tlačítek *Page Up* a *Page Down* v ose Z. Také je možné se stisknutím tlačítek W nebo S pohybovat dopředu, případně dozadu ve směru osy pohledu. Tlačítky A a D je možné pohybovat se do stran a tlačítky Q a E nahoru a dolů. Při stisku pravého tlačítka myši je možné měnit úhel natočení kamery.

1.22.4 Hlavní okno v režimu prohlížení výsledků

Hlavní menu a první tlačítková lišta je opět shodná s již zmíněnými dvěma režimy. Na této liště se nachází tlačítka pro volbu režimu programu.

Pod hlavní lištou se nachází panel pro volbu bloku, ze kterého se mají zobrazovat výsledky. Grafy lze zobrazit z bloku *VMObjekt* a *Scope*. Dále je možné zvolit vnořené bloky *VMSchema* a vybrat bloky *VMObjekt* a *Scope* z nich. Pro návrat do nadřazeného *VMSchema* slouží nejvyšší položka z menu označená "..". Při výběru bloku typu *VMObjekt* se zobrazí ještě jeden výsuvný list, ze kterého je možné zvolit si typ grafu, jenž se má zobrazit.

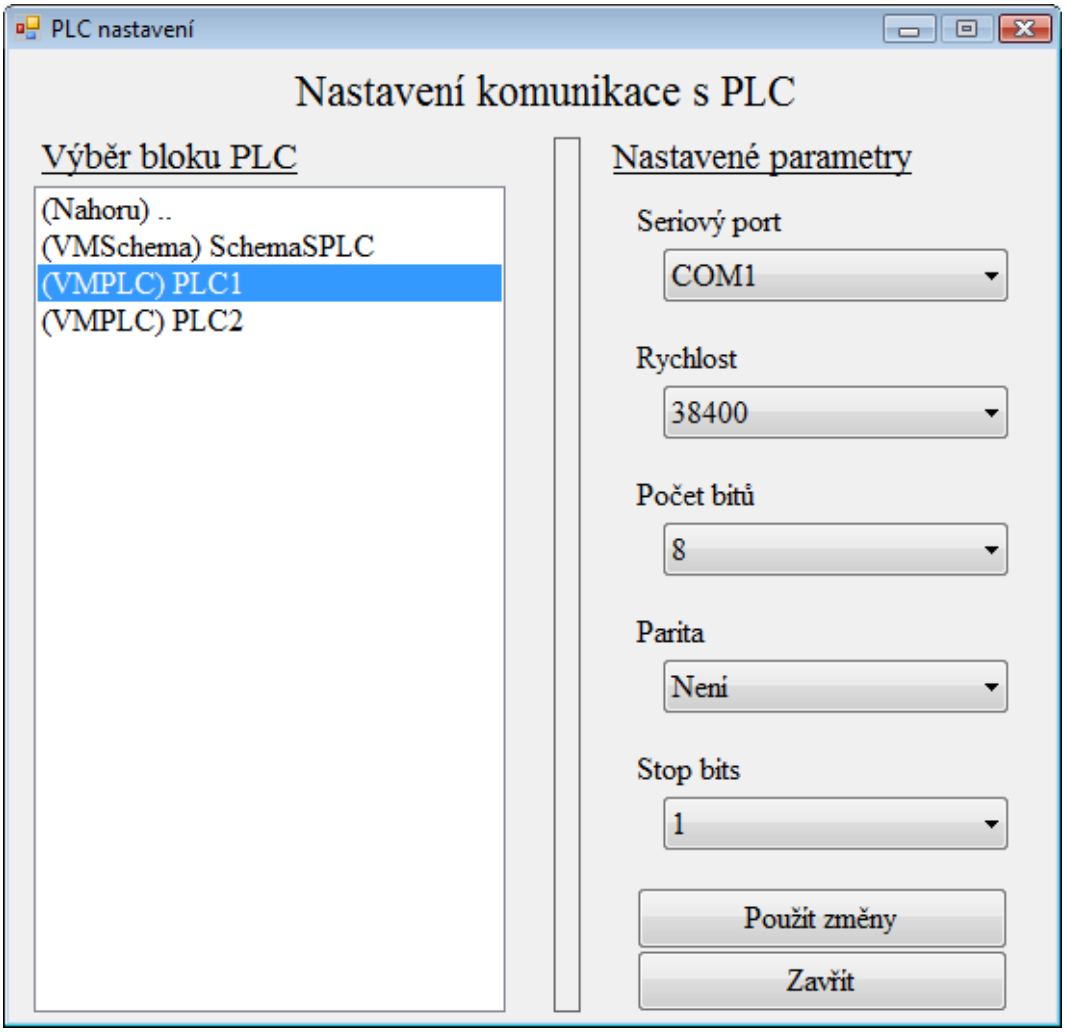
Ve spodní části okna se nachází panel pro nastavení minimální a maximální hodnoty zobrazené na oknech a také počet značek zobrazených na oknech. Celkové uspořádání je na obrázku 12.



12. Obrázek: Prohlížení výsledků

1.22.5 Dialogové okno pro nastavení komunikace s PLC

Dialogové okno umožňuje podle potřeby nastavit jednotlivým blokům *VMPLC* komunikační parametry. V levé části okna se nachází průzkumník schématu se zobrazenými bloky *VMPLC* a *VMSchema*. Bloky *VMSchema* jsou zobrazeny pro nastavení parametrů vnořených bloků *VMPLC*. Při vnoření do zvoleného schématu se jako první v seznamu objektů objeví symbol "..", který slouží pro návrat do nadřazeného schématu. V pravé části okna se nachází ovládací prvky pro zvolení komunikačních parametrů s konkrétním *PLC*. Těmito parametry jsou jméno portu, rychlost komunikace, počet bitů, parita a definice tvaru *Stop bitů*. Obrázek okna pro nastavení komunikace s *PLC* je na obrázku 13.



13. Obrázek: Okno komunikace s PLC

1.23 Ukázkové příklady

Ukázkové příklady demonstrují možnosti programu a nastiňují principy stavby virtuálních modelů včetně prezentace výsledků simulace.

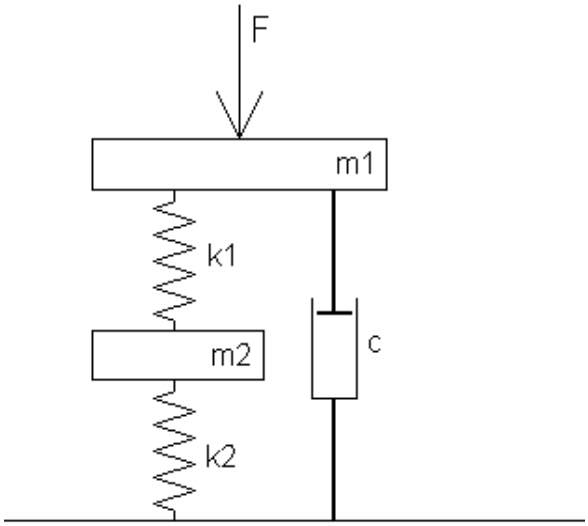
První ukázkový model se zabývá simulací dvou hmotných těles v soustavě pružin a tlumičů. Stejný ukázkový příklad byl použit v semestrálním magisterském projektu a snahou je demonstrovat rozdíly.

Druhý ukázkový model nazvaný Křižovatka reprezentuje rozsáhlý virtuální model simulující provoz a řízení světelné křižovatky. Pro řízení světel semaforu je možné použít *PLC*.

1.23.1 Pohyb dvou hmotných těles v soustavě pružin a tlumičů

Ukázkový příklad, znázorněný na obrázku 14, demonstruje simulaci pohybu dvou hmotných těles v systému pružin a tlumičů. Nejsou uvažovány žádné ztráty. Tělesa se pohybují pouze ve svislém směru a nemůžou se vychýlit do stran. Je uvažována gravitační síla.

1.23.1.1 Zadání



F= 40 N
m_1= 10 kg
m_2= 150 kg
k_1= 1200 N/m
k_2= 900 N/m
c= 50 kg/s

14. Obrázek: Schéma soustavy

1.23.1.2 Matematicko-fyzikální popis

Souřadný systém

Počátek osy y_1 položíme do těžiště hmotného bodu m_1, nepůsobí-li na něj gravitační síla a nejsou-li pružiny předepnuty.

Počátek osy y_2 do těžiště m_2 za stejných předpokladů.

Obě osy jsou orientovány nahoru.

Těleso soustavy 1

$$F_{g1} = -m_1 \cdot g$$

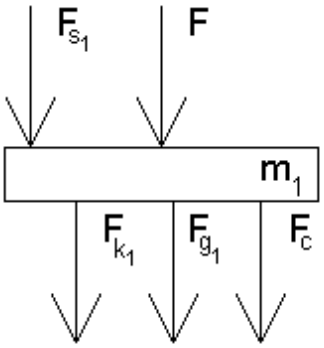
$$F_{k_1} = -k_1 \cdot (y_1 - y_2)$$

$$F_{s1} = -m_1 \cdot \ddot{y}_1$$

$$F_c = -c \cdot \dot{y}_1$$

$$-F + F_{k1} + F_s + F_{g1} + F_c = 0$$

$$F + k_1 \cdot (y_1 - y_2) + m_1 \cdot \ddot{y}_1 + m_1 \cdot g + c \cdot \dot{y}_1 = 0$$



15. Obrázek: Silové schéma prvního tělesa

Těleso soustavy 2

$$F_{g2} = -m_2 \cdot g$$

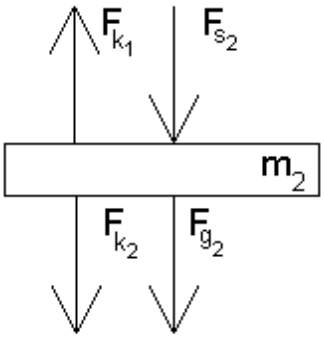
$$F_{k1} = k_1 \cdot (y_1 - y_2)$$

$$F_{k2} = -k_2 \cdot y_2$$

$$F_{s2} = -m_2 \cdot \ddot{y}_2$$

$$F_{k1} + F_{k2} + F_{s2} - F_{g2} = 0$$

$$k_1 \cdot (y_1 - y_2) - k_2 \cdot y_2 - m_2 \cdot \ddot{y}_2 - m_2 \cdot g = 0$$



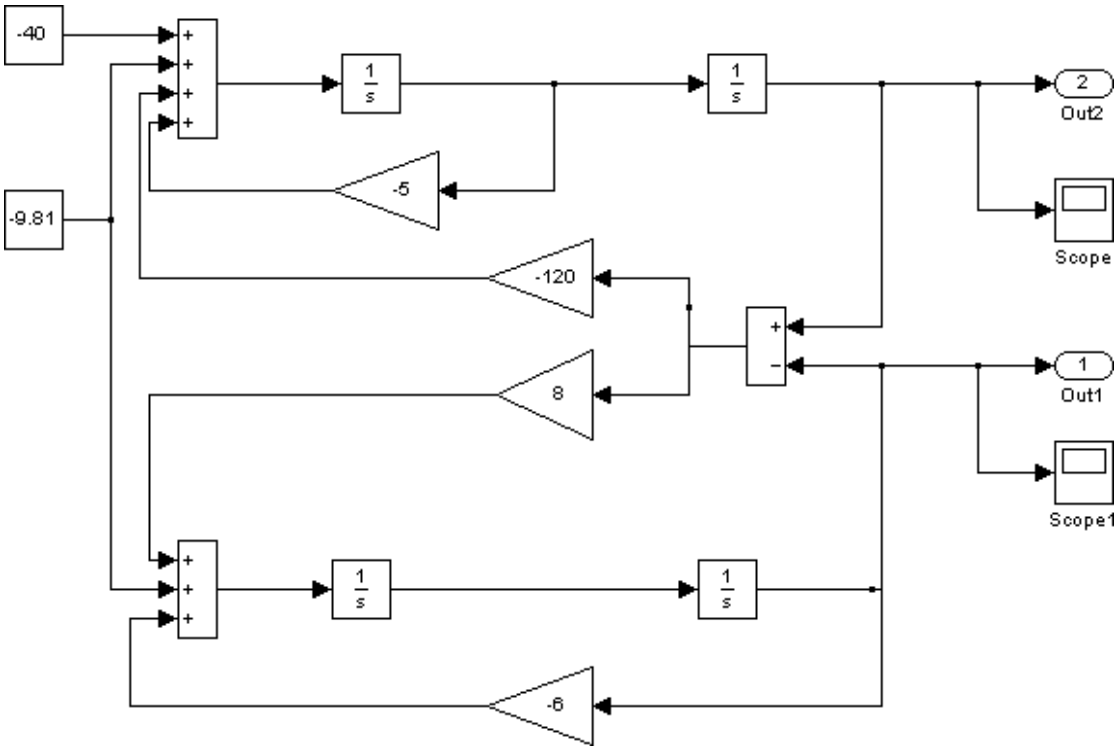
16. Obrázek: Silové schéma druhého tělesa

1.23.1.3 Simulační schéma

Z odvozených rovnic v matematicko-fyzikálním popisu vyjádříme druhé derivace polohy a navrhneme schéma v programu *MATLAB Simulink*. Simulační schéma je zobrazeno na obrázku 17.

$$\ddot{y}_1 = -\frac{c}{m_1}\dot{y}_1 - \frac{k_1}{m_1}(y_1 - y_2) - g - \frac{F}{m_1}$$

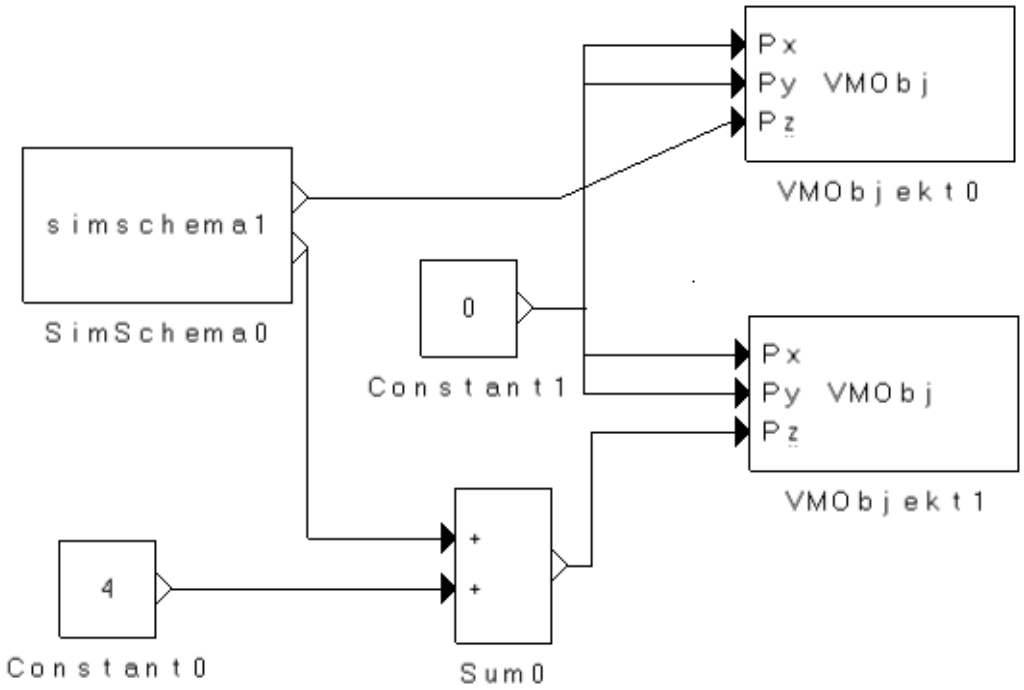
$$\ddot{y}_2 = \frac{k_1}{m_2}(y_1 - y_2) - \frac{k_2}{m_2}y_2 - g$$



17. Obrázek: Simulační schéma v MATLAB Simulink

1.23.1.4 VMSchema

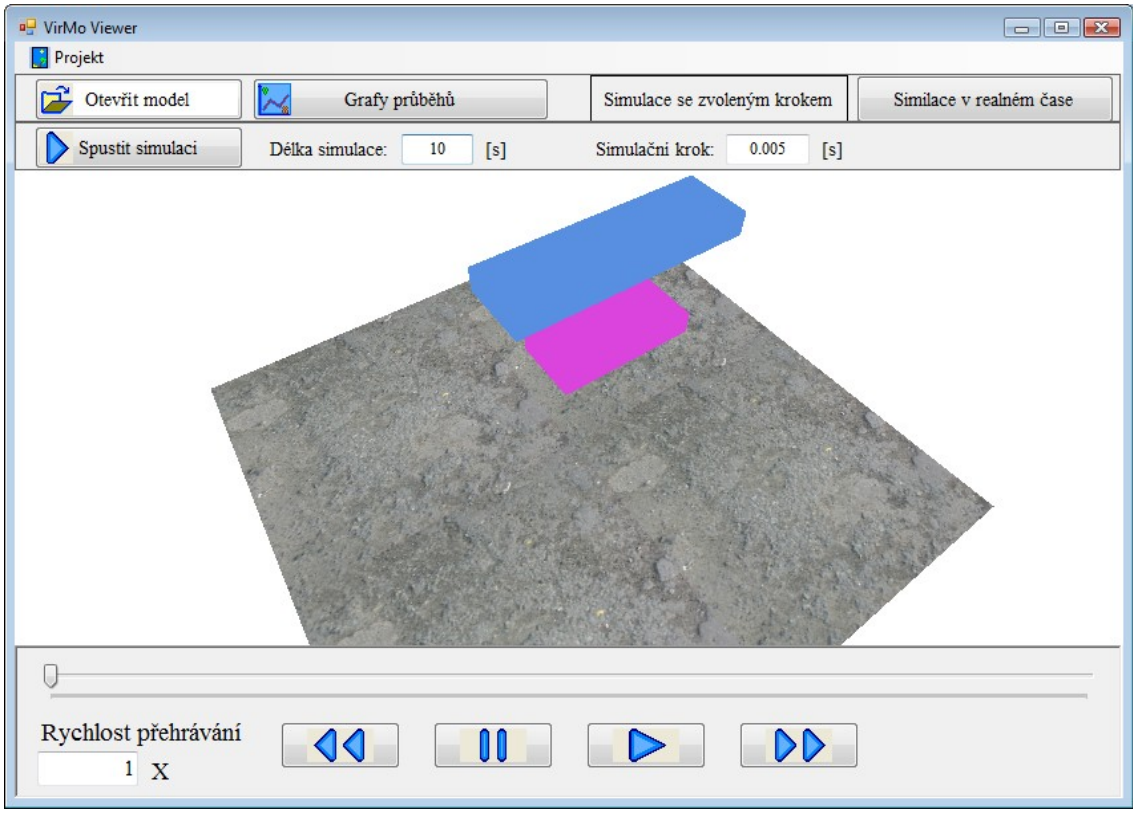
Vytvořené simulační schéma je vloženo do programu *VMEditor* jako funkční blok. Dále jsou přidány dva bloky *VMObjekt* pro vizualizaci obou hmotných těles. Přičítaná konstanta zavádí posunutí souřadného systému prvního tělesa. Vyobrazeno na obrázku 18.



18. Obrázek: Blokové schéma pro vizualizaci

1.23.1.5 Vizualizace simulace

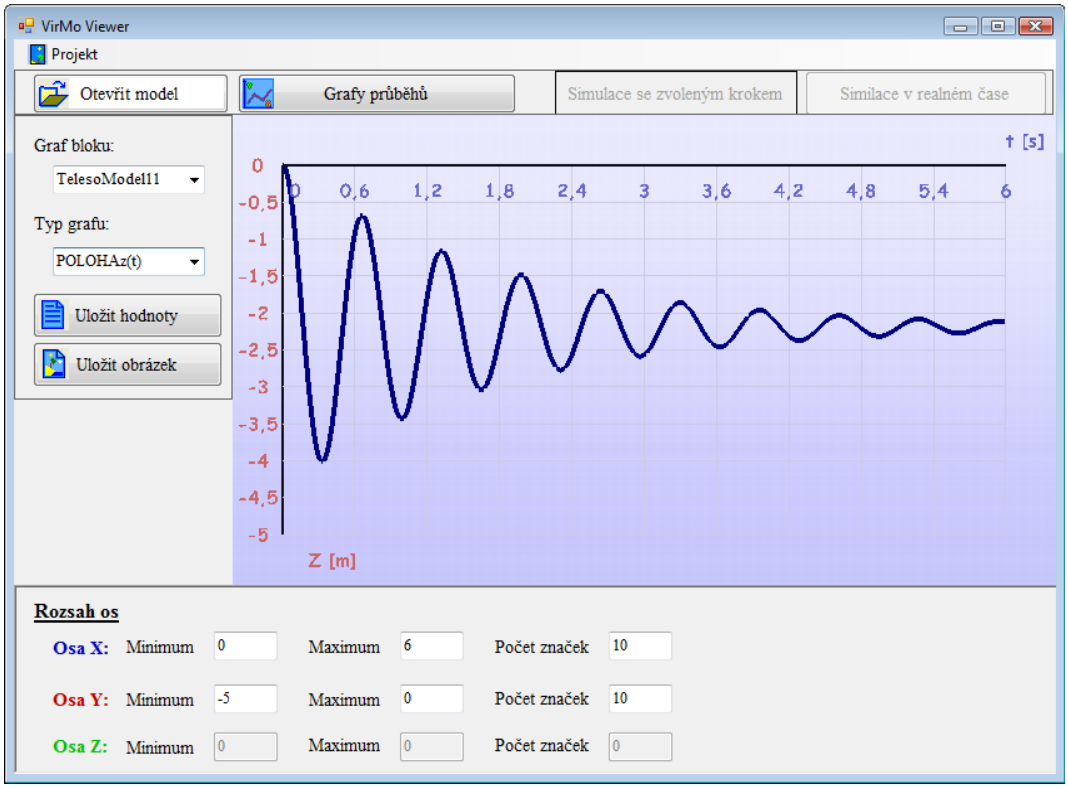
Po načtení projektu a výpočtu simulace je možné prohlédnout si vizualizaci procesu. Ve vizualizaci je vidět pohyb obou těles v prostoru vzhledem k nehybné podlaze. Ve vizualizaci lze kamerou volně pohybovat pomocí směrových šipek a otáčet pomocí myši. Na obrázku 19 je ukázka výsledku simulace prezentované programem.



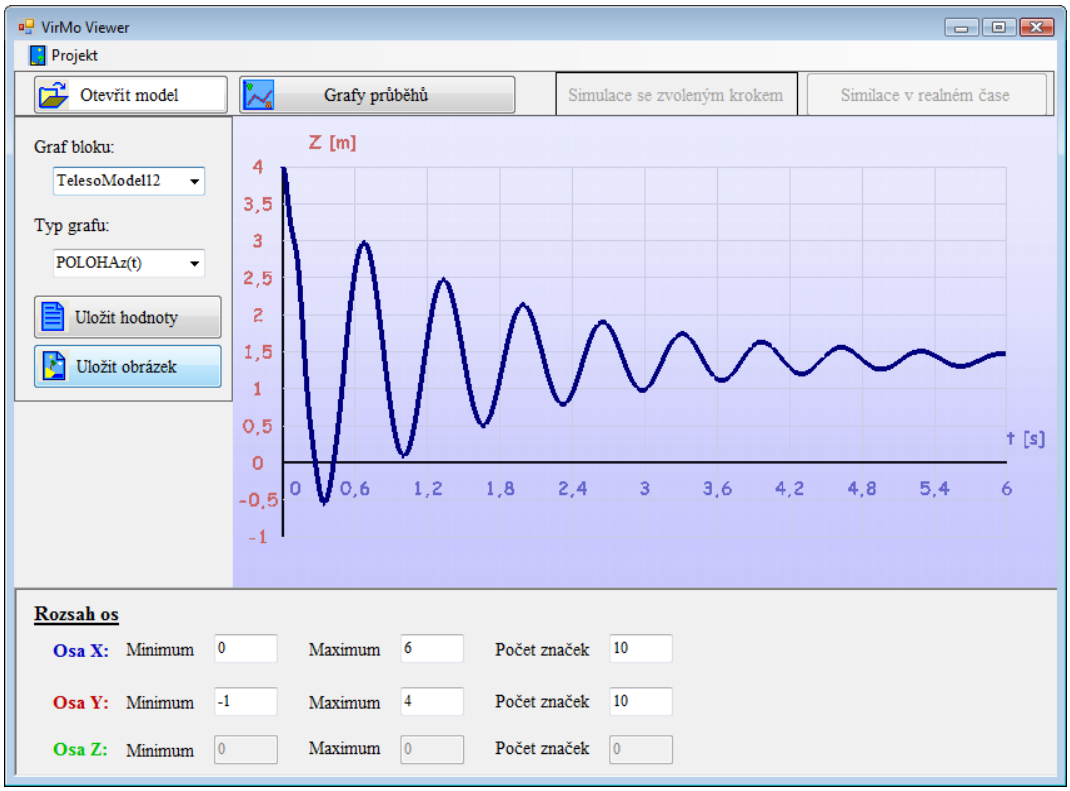
19. Obrázek: Vizualizace ukázkového příkladu

1.23.1.6 Zhodnocení výsledků simulace a průběhy sledovaných veličin

Důležitými veličinami v tomto ukázkovém příkladu jsou poloha hmotných těles v průběhu času, případně derivace polohy představující rychlost a druhá derivace popisující zrychlení pohybu těles. Z hlediska orientace třírozměrného prostoru jsou sledované veličiny zaznamenány v ose Z. Průběh polohy v ose Z prvního tělesa je vykreslen v programu na obrázku 20. Na obrázku 21 je zobrazen pohyb druhého tělesa.



20. Obrázek: Průběh polohy prvního tělesa v ose Z



21. Obrázek: Průběh polohy druhého tělesa v ose Z

1.23.2 Virtuální model Křižovatka

Virtuální model *Křižovatka* byl vytvořen pro srovnání s už existujícím modelem uvedeným v mé bakalářské práci. Model demonstruje stavbu rozsáhlého návrhu složeného z velkého množství základních stavebních celků. Jednotlivé stavební celky mají za úkol řídit samostatné části modelu jako je realizace semaforů nebo řízení pohybujících se aut. Příklad také demonstruje komunikaci s PLC, do kterého je nahrán příslušný program pro řízení semaforů. Na obrázku 22 je ukázka vizualizace virtuálního modelu.

22. Obrázek: Vizualizace ukázkového příkladu Virtuální model Křižovatka

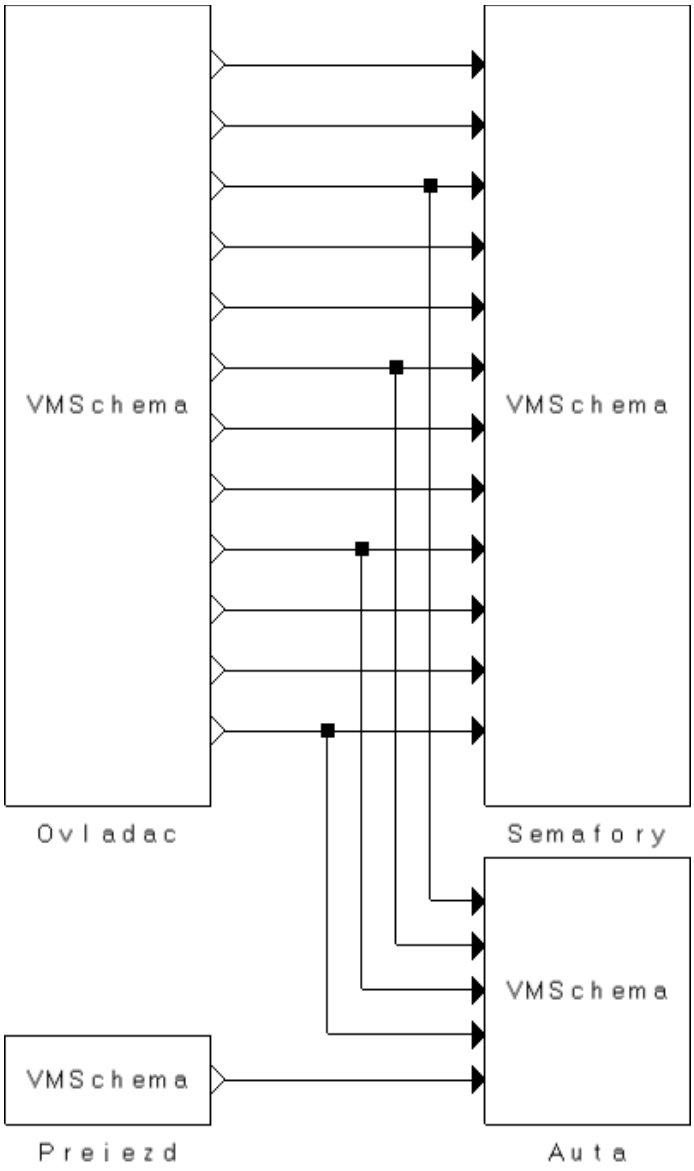
1.23.2.1 Popis modelu

Model simuluje řízení semaforů na křižovatce tvaru kříže. Semafory mohou být řízeny pomocí *PLC* anebo schématem. Po křižovatce se pohybuje sedm aut tří typů, která vjedou do křižovatky pouze na zelené světlo semaforu z příslušného směru. Na jedné

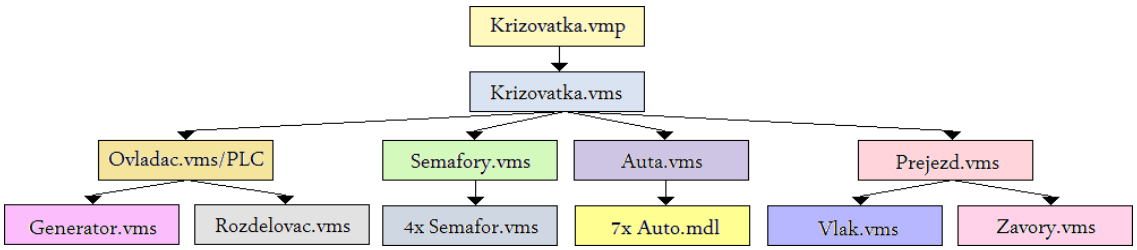
příjezdové silnici je umístěn železniční přejezd, kde v pravidelných intervalech projíždí vlak. Přejezd má funkční výstražná světla a závory. Auta dávají přednost projíždějícímu vlaku a zastavují před přejezdem.

Hlavní schéma

Hlavní schéma realizuje propojení čtyř nižších celků. První blok se stará o řízení semaforů. Model byl vytvořen ve dvou verzích. V první verzi jsou aktivovány semafory pomocí schéma. V druhé verzi pak prostřednictvím *PLC*. Druhý blok se stará o vizualizaci semaforů podle aktivovaných vstupů světel. Třetí blok slouží pro řízení a vizualizaci aut. Poslední blok zajišťuje řízení vlaku a přejezdu. Na obrázku 23 je vyobrazeno hlavní schéma a na obrázku 24 strom projektu zachycující hierarchii jednotlivých částí.



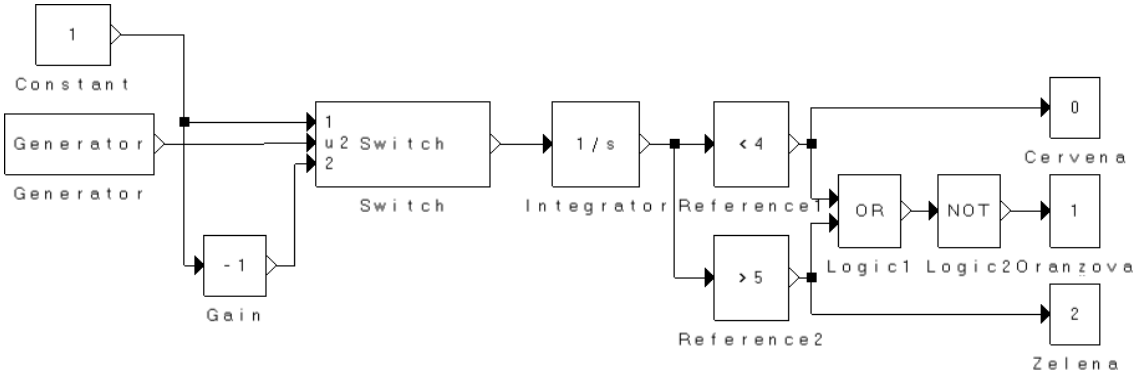
23. Obrázek: Hlavní schéma



24. Obrázek: Strom projektu

Schéma Ovladac.vms

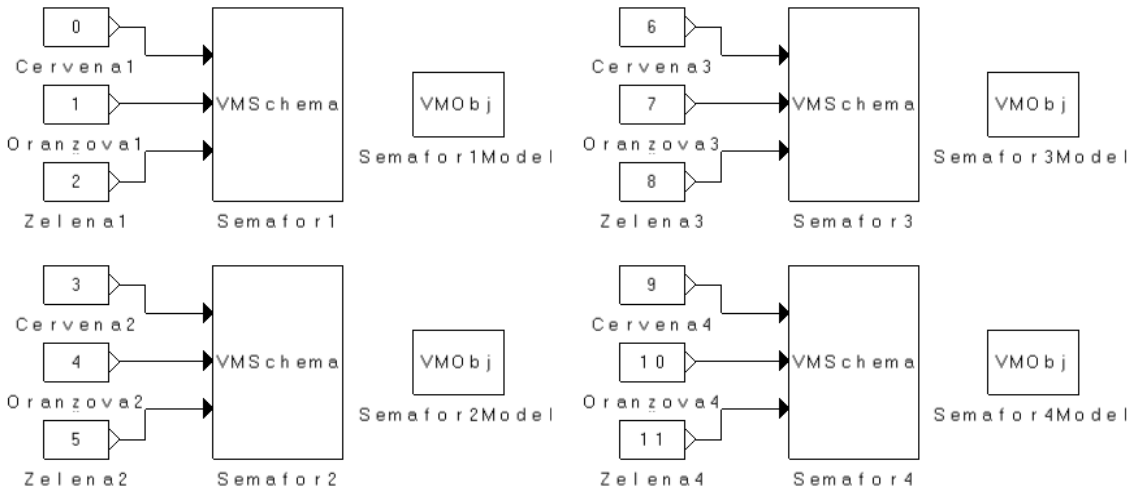
Řídící schéma lze použit na místo *PLC* a stará se o aktivaci jednotlivých semaforů. Skládá se z generátoru a rozdělovače. Generátor má tři výstupní porty, jejich aktivace se v průběhu času střídá. Oranžové světlo má nejkratší časový interval. Schéma generátoru je na obrázku 25. Rozdělovač distribuuje signály ze tří výstupu generátoru na 12 výstupů pro jednotlivá světla. V jednom směru křižovatky jsou výsledky aplikovány přímo a v druhém směru je aktivace červených a zelených světel obrácena.



25. Obrázek: Schéma generátoru

Schéma Semafory.vms

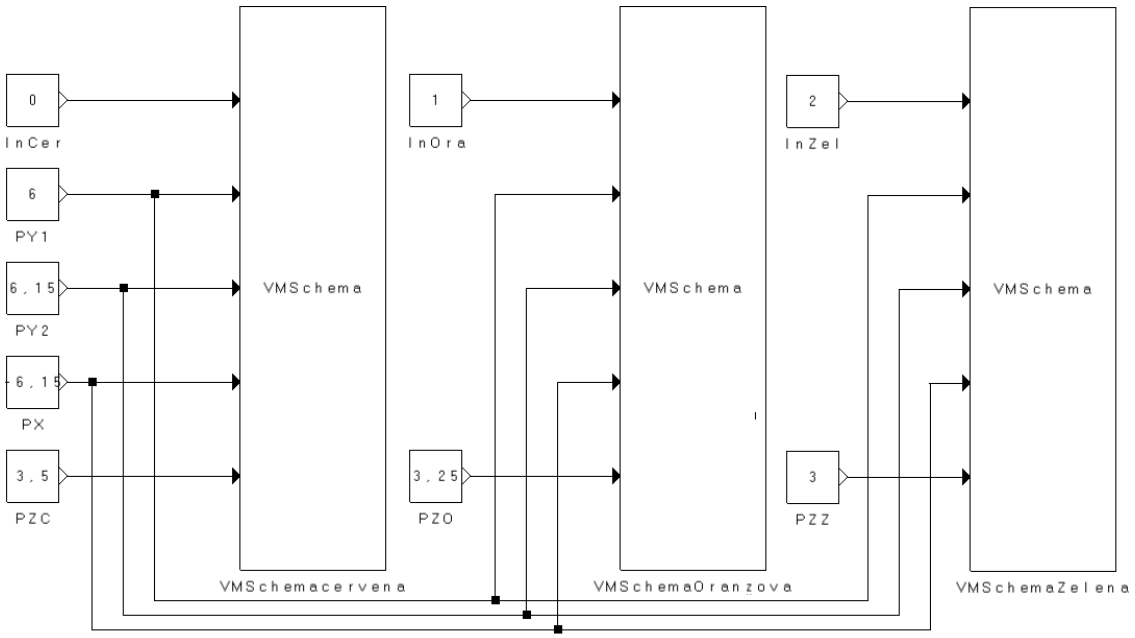
Schéma je použito k zapouzdření logiky semaforů. Obsahuje dvanáct vstupních portů pro aktivaci jednotlivých světel a bloky *VMObjekt* použité k zobrazení modelu sloupu semaforu. Posunutí modelů je realizováno nastavením počátečních podmínek bloků. Schéma je zachyceno na obrázku 26.



26. Obrázek: Schéma Semafory.vms

Schéma Semafor.vms

K realizaci světel je použito schéma jednotlivých semaforů. Ve schématu se nachází tři vstupní porty pro aktivaci červeného, oranžového nebo zeleného světla. Rozsvěcení semaforu je realizováno posunutím modelu tvaru koule většího průměru a světlejší barvy tak, aby byla viditelná. Při zhasnutém světle je větší světlejší koule uschována v sloupu semaforu a je vidět koule tmavší barvy. Bloky *VMSchema* realizují samotné posouvaní s modely. Schéma se nachází na obrázku 27.



27. Obrázek: Schéma Semafor.vms

Schéma Auta.vms

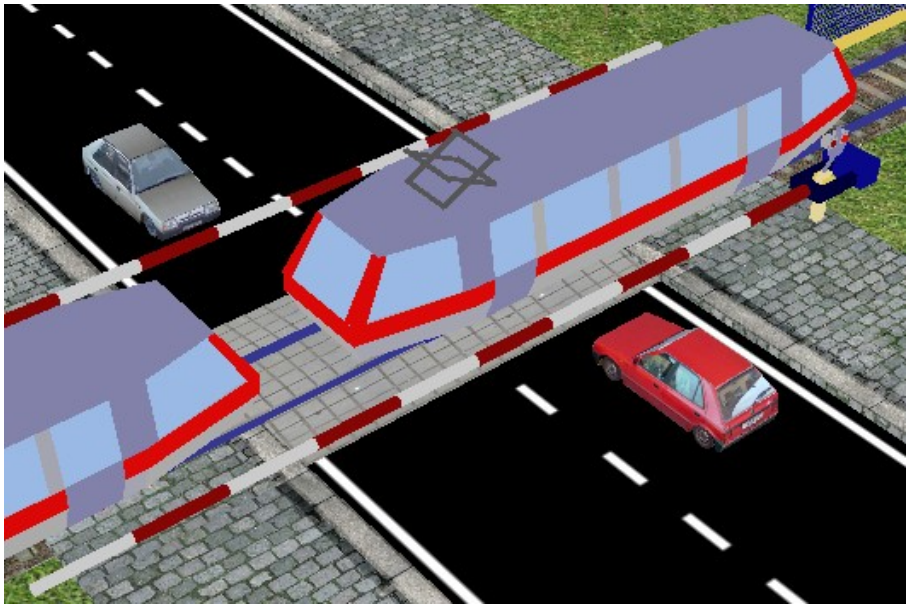
Schéma se stará o simulaci pohybu aut a jejich vizualizaci. Ve schématu je vnořeno sedm bloků obsahujících schéma Auto.mdl. Bloky mají navzájem propojeny porty. Porty slouží k předání informace o poloze v definované trajektorii pro zastavení za jiným autem. Bloky mají také přiveden signál o aktivaci zelených světel a přejezdu. Bloky *VMTrajektrorie* převádí postup v trajektorii na polohu auta ve scéně.

Schéma Auto.mdl

Schéma řídí jízdu auta po předem zadané trajektorii. Pokud se před autem v brzdné vzdálenosti nenachází překážka, auto zrychluje až na svou maximální rychlost. Je-li zaregistrována překážka, auto začne plynule zpomalovat. Překážkou rozumíme nerozsvícené zelené světlo, jiné auto v cestě nebo aktivovaný přejezd. Brzdná vzdálenost je počítána pro aktuální rychlost. Schéma je rozsáhlé, a proto je vynecháno jeho zobrazení, které by bylo značně nepřehledné. Obsahuje jedenáct vstupních portů a jeden výstupní. Čtyři porty slouží k přivedení informace o aktivaci zelených světel semaforů. Schéma také obsahuje vstupní porty pro polohu ostatních aut a informaci, zda je aktivován železniční přejezd. Na jediném výstupním portu je uvedena aktualizovaná poloha daného auta.

Schéma Prejez.vms

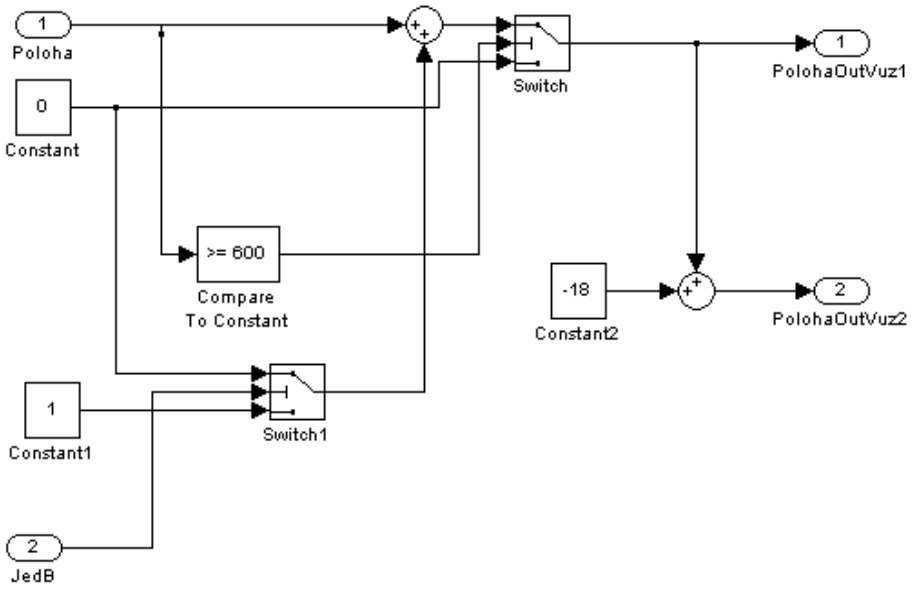
Schéma definuje chování přejezdu a jízdu vlaku. Vlak tvořený dvěma vozy se pohybuje po předdefinované trajektorii konstantní rychlostí. Na obrázku 28 je vyobrazena vizualizace oblasti přejezdu.



28. Obrázek: Ukázka oblasti přejezdu

Schéma Vlak.vms

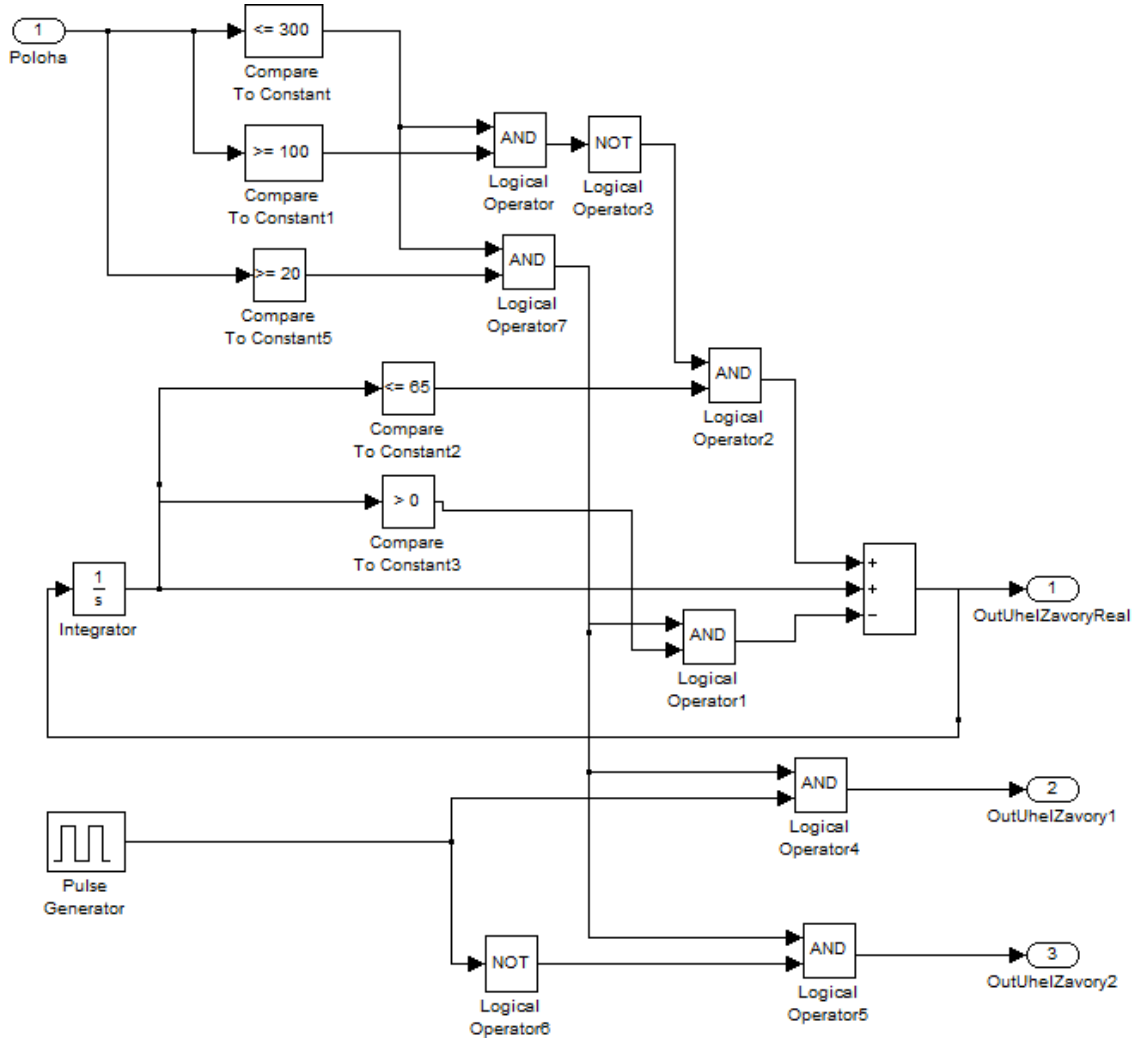
Schéma pro řízení vlaku je vyobrazeno na obrázku 29. Obsahuje vstupní port pro povolení jízdy vlaku. Na dvou výstupních portech se nachází aktualizované pozice obou vozů vlaku.



29. Obrázek: Simulační schéma pro řízení vlaku

Schéma Zavory.vms

K realizaci funkce přejezdu slouží schéma *Prejezd.vms*. Přejezd je tvořen pohyblivými závorami a funkčním výstražným zařízením. Pokud je vlak v nastavené vzdálenosti od přejezdu, jsou aktivovaná výstražná světla a poté sklopeny bezpečnostní závory. Po průjezdu vlaku je přejezd deaktivován. Na obrázku 30 se nachází schéma pro řízení přejezdu.



30. Obrázek: Schéma pro řízení přejezdu

Závěr

Diplomová práce navazuje na magisterský semestrální projekt „Software pro návrh a vizualizaci blokových schémat“. Práce je rozšířena o návrh simulačního prostředí a vývoj softwaru pro simulaci obecných blokových schémat.

Program umožňuje simulaci blokových schémat se zvoleným krokem a vizualizaci virtuálního modelu vybranou rychlostí. Program je vytvořen ve vývojovém prostředí *Microsoft Visual Studio*. Celý software je postaven modulárně. K propojení jednotlivých modulů slouží jednotka *scene*. Jednotlivé moduly realizují výpočetní a vykreslovací jádro programu. Část modulů obsahuje definici uživatelského rozhraní. V práci je zahrnuta uživatelská dokumentace k implementovanému programu.

K návrhu simulačních schémat jsou využity bloky z prostředí *MATLAB-Simulink*, které zajišťují standardizaci a čitelnost těchto schémat. Pro zvýšení možností programu je definována skupina rozšiřujících bloků. Rozšiřující bloky slouží například pro realizaci vizualizace, nebo připojení *PLC* TECOMAT prostřednictvím protokolu EPSNET do simulovaného systému. Schémata jsou do programu načtena jako zdroje informací a definují simulovaný systém. Po jejich výpočtu lze zobrazit průběhy jednotlivých sledovaných veličin. Pro vizualizaci virtuálních modelů jsou načteny třírozměrné modely ve formátu *ASE*. Větší věrohodnosti modelu je dosaženo využitím textur formátu *TGA*. Mezi výhody formátu *TGA* patří například možnost použití *alpha kanálu*. Vykreslení je realizováno pomocí rozhraní *OpenGL*. Při návrhu uživatelského rozhraní je kladen důraz na jednoduchost a intuitivní ovládání.

Pro demonstraci funkčnosti softwaru jsou vytvořeny dva ukázkové příklady. Obsáhlý virtuální model klade zvýšený nárok na hardware počítače. Při vykreslení vizualizace modelu jsou kladeny nároky na celý grafický systém. Program je využitelný pro široké spektrum virtuálních modelů, obzvláště pro simulaci procesů řízených *PLC*. Aplikace je vhodná i pro výuku předmětů zabývajících se prací s *PLC,* nebo předmětů, studujících vlastnosti dynamických systémů.

Tvorba programu byla náročná jak z hlediska časového, tak nutností pochopení problematiky mnoha oblastí. Realizace zadáné diplomové práce byla velmi zajímavá a zábavná. Přínosem není jen vytvoření obsáhlého software, ale i získání nových informací a další programátorský růst.

Použité informační zdroje

[1] WOO, M. *et al.; OpenGL Programming Guide*, Third Edition, Massachusetts: Addison-Wesley, Reading, 1999.

[2] KRUGLINSKI, D. J. et al.: *Programujeme v Microsoft Visual C++*, Computer Press, Praha, ISBN 8072263625

[3] ŠALOUN, Petr, *Programovací jazyk C pro zelenáče*, Neokortex, 1999, ISBN 80-86330-02-X

[4] TEIXEIRA, Steve, PACHECO, Xavier, *Mistrovství v Delphi 6*, Computer Press Vydání první, 2002, ISBN 80-7226-627-6

[5] CANTÚ, Marco, *Mistrovství v Delphi 2 – pro Windows 95/NT*, Computer Press, Vydání první, 1996, ISBN 80-85896-75-3 CANTÚ, Marco, *Mistrovství v Delphi 2 – pro Windows 95/NT*, Computer Press, Vydání první, 1996, ISBN 80-85896-75-3

[6] STROUSTRUP, Bjarne, *C++ Programovací jazyk*, BEN - technická literatura, 1. české vydání, 1997, ISBN 80-901507-2-1 a ISBN 80-86056-20-1

[7] Uživatelská příručka firmy TECO a.s., *Sériová komunikace programovatelných automatů TECOMAT a regulátorů TECOREG – model 16 bitů*, 11. vydání - březen 2004,TECO a.s

[8] Pascal.webz.cz [online], *Internetové stránky programu Pascal,* Aktualizováno: 31. 1. 2005, Dostupné z: <http://pascal.webz.cz/kurs/lekce/lekce0.htm>

[9] Grafika Publishing, s.r.o., *Internetové stránky programátorů builder,* Dostupné z: <http://www.builder.cz/art/delphi/dserial_trocha_historie.html>

[10] Uživatelská příručka firmy TECO a.s., *Sériová komunikace programovatelných automatů TECOMAT - model 32 bitů,* 15. vydání - březen 2009, Dostupné z: <http://www.tecomat.cz/docs/cze/General/txv00403.pdf>

[11] Uživatelská příručka firmy TECO a.s., *Začínáme v prostředí MOSAIC,* 8. vydání - duben 2010,
Dostupné z: <http://www.tecomat.cz/docs/cze/Software/Mosaic/TXV00320.pdf>

[12] KRČMÁŘ,. Petr, *Root.cz (www.root.cz), informace nejen ze světa Linuxu*, Dostupné z: <http://www.root.cz/clanky/graficky-format-tga-jednoduchy-oblibeny-pouzivany/>

[13] MATLAB - The Language Of Technical Computing, *Internetové stránky programu MATLAB,* Dostupné z: <http://www.mathworks.com/products/matlab/>

[14] SOBOLÍK, Martin, Ing., Průmyslová automatizace, *Programování PLC*, Dostupné z: <http://www.plc-control.net/plc_control.htm>

[15] VAVROUŠEK, Miroslav, Bakalářská práce, *Tvorba nástroje pro návrh obecných virtuálních modelů*, Technická univerzita v Liberci, květen 2008

[16] Internetové stránky firmy TECO a.s.,
Dostupné z: <http://www.tecomat.cz>