Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

Bakalářská práce

Kapitoly videotutoriálu MVE-2

Plzeň, 2011 Martin Matas

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 12.5.2011 Martin Matas

Abstract

MVE-2 Video Tutorial Chapters

This bachelor thesis describes a process of recording a video tutorial to Modular Visualization Environment – 2 (MVE), which is an environment based on the idea of modular programming.

First, it provides a brief introduction to modular programming in general, then MVE is introduced.

Chapter 3 in this thesis is very important, because the tools available to record videos and the chosen methods and tools are described here. Properties, advantages and disadvantages of the chosen tool are discussed and a guideline to record with the chosen tool is provided.

Content of chapters of the video tutorial is provided in the remaining chapters, they talk about installation of MVE, usage of the environment and even creating custom modules in the end.

Obsah

1 Úvod

Mým úkolem je vybrat vhodné programy, nástroje a postupy k nahrání šesti kapitol video tutoriálu k Modular Visualization Environment – 2 (v textu budu vždy zkracovat na MVE), česky Modulární vizualizační prostředí. To je přívětivé grafické prostředí pro modulární programování. Tyto nástroje a postupy jsou popsány v kapitole 3 v této bakalářské práci.

Na tento popis navazuje druhá část úkolu, a to je samotné nahrání těchto šesti na sebe navzájem navazujících kapitol tvořících základní tutoriál, který diváka provede od teorie přes instalaci a první kroky v prostředí až k vytváření vlastních modulů a tím ho naučí jak používat MVE na úrovni středně pokročilého uživatele. Obsahy těchto kapitol jsou v kapitolách 4 až 9 v této bakalářské práci.

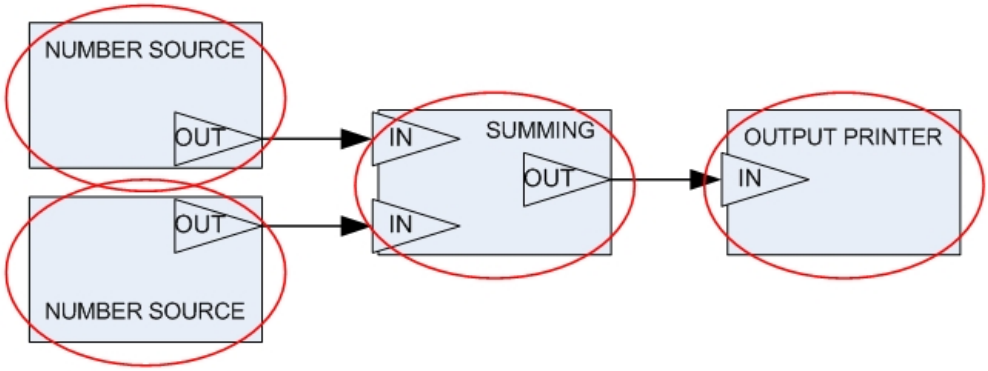
Celý video tutoriál je v angličtině.

2 O modulárním programování

Modulární programování je programování v grafickém prostředí, ve kterém jsou spojovány jednotlivé dostupné moduly do mapy modulů (dále mapa). Každý modul provádí nějakou elementární úlohu. Výslednou mapu lze spustit a tím získáme program, který jsme chtěli od počátku naprogramovat. Moduly spojujeme tak, že spojujeme jejich vstupy a výstupy.

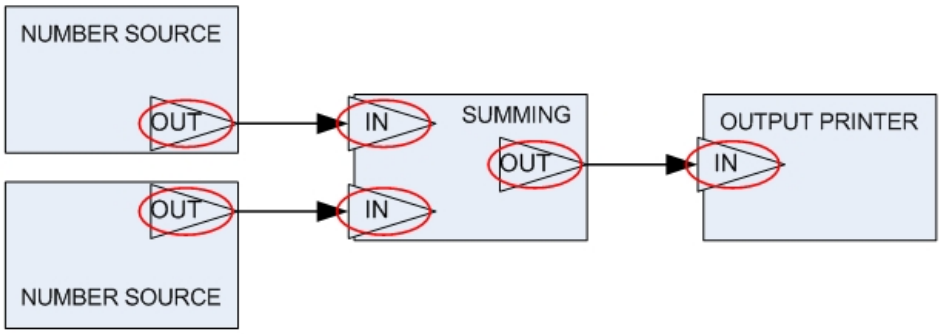
2.1 Modul, port, link

Mapa na obrázcích 1, 2 a 3 je velice jednoduchá mapa, která sčítá dvě čísla.



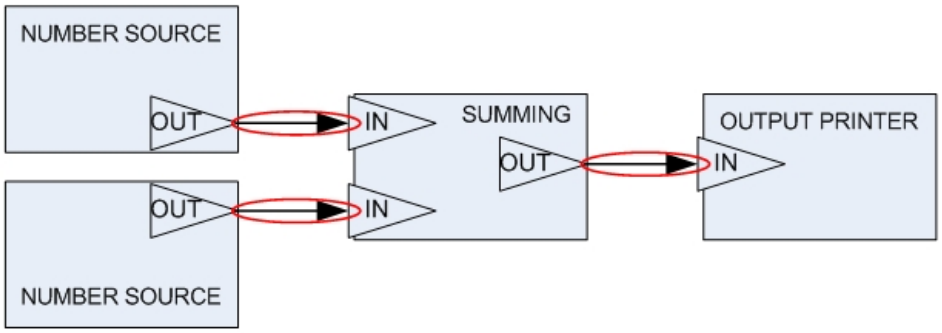
Obrázek 1: moduly

Na obrázku 1 jsou označeny moduly.



Obrázek 2: porty

Na obrázku 2 jsou označeny porty. Porty se dělí na vstupní a výstupní. Díky portu modul dostává/poskytuje svá vstupní/výstupní data.



Obrázek 3: linky

Porty jsou spojeny linky, neboli spojeními. Linky vidíme na obrázku 3.

2.2 Spouštění modulů ve správném pořadí

Prostředí pro modulární programování by mělo zajišťovat proces spouštění modulů ve správném pořadí. Ten probíhá následujícím způsobem:

Prostředí najde všechny terminální moduly – to jsou moduly, které nemají žádný výstup – a pokusí se je spustit. Těm ale ještě nejsou dostupná vstupní data, a tak se prostředí zaměří na moduly, které data poskytnou. Ty se opět pokusí spustit a tím se opět posune zaměření na jiné moduly. To se opakuje tak dlouho dokud nenarazí na moduly, které nepotřebují žádný vstup. Tyto moduly se spustí, a pak se pokračuje ve spouštění v opačném pořadí, než ve kterém se posouvalo zaměření.

Mapa nemůže obsahovat smyčky, například že by byl výstup modulu M propojen na vstup modulu A, výstup A na vstup B, ..., ... na vstup Z a výstup Z na vstup modulu M.

2.3 Výhody a nevýhody

Výhodami modulárního programování je to, že je jednoduché, umožňuje spolupráci více lidí na projektu, protože každý člověk může dělat svůj modul. Výhodnými vlastnostmi modulárního programování jsou typová kontrola dat mezi moduly a znovuvyužitelnost kódu – v tom smyslu, že modul se naprogramuje jen jednou, a pak se může využít v libovolném množství map.

Nevýhodami modulárního programování je to, že můžeme potřebovat modul, který ještě nebyl naprogramován, že přidání nového modulu může být komplikované, a že přináší nějakou práci navíc.

3 O MVE-2

Systém MVE-2 je modulární prostředí postavené na základní myšlence modulárního programování, které je vyvíjeno na Katedře informatiky a výpočetní techniky na Západočeské univerzitě v Plzni. Obsahuje podporu pro vytváření, upravování a spouštění jednotlivých map. Zajišťuje komunikaci mezi jednotlivými moduly, předávání dat mezi nimi, dále pak proces spouštění modulů ve správném pořadí.

Asi nejdůležitější součástí MVE je grafické prostředí MapEditor, ve kterém uživatel vytváří a upravuje mapy a může nastavit, aby se mapa automaticky spustila několikrát za sebou s různými nastaveními. Jedna mapa představuje hotový program, který jsme chtěli naprogramovat. Mapa sestává z modulů, jeden modul vykonává nějakou elementární činnost. Modul může mít libovolný počet různých nastavení, které se v mapě dají přenastavit dle libosti.

MVE je založeno na datovém toku a podporuje vytváření datových struktur které jsou mezi moduly sdíleny. Díky obecnému návrhu jádra není při přidání nové datové struktury potřeba rekompilace celého systému. Prostředí pouze poskytuje standard, který říká, co musí datová množina splňovat, aby ji bylo možné sdílet mezi moduly. Každá knihovna modulů pak může definovat vlastní datové struktury.

K principům modulárního programování patří, že není možné propojit moduly v mapě do cyklu. MVE tento princip respektuje. Pokud jsou v modulech použity běžné porty, tak v případě modulů propojených do cyklu MVE oznámí, že mapu nelze spustit. Zároveň ale přidává ještě jinou možnost, a to díky speciálním druhům portů, které umožní modulu, který je obsahuje, řídit spouštění modulů v části mapy. Díky tomu může být v této části mapy cyklus, se kterým již tento modul počítá.

Dvě vysoce ceněné vlastnosti MVE jsou: možnost cyklického propojení modulů a modulem řízené spouštění části mapy.

Mapy se ukládají v textovém XML souboru. Proto je možné mapu upravovat i bez prostředí MapEditor – je možné upravovat přímo tento XML soubor. Samotné spuštění mapy je pak možné i v rozhraní RunMap pomocí příkazové řádky – mapa se pouze spustí, bez načítání dalšího grafického prostředí apod.

Obecnost a datové množiny, které jsou moduly sdíleny, jsou tak důležité, že při špatném pochopení MVE a při špatném použití by programátorům jenom přidělávalo práci – pokud by si každý programátor udělal své vlastní datové typy a poté by chtěl přimět svoje moduly ke spolupráci s moduly jeho kolegů.

Myšlenka MVE je v tom, aby vznikaly navzájem kompatibilní moduly, a proto musí být nejprve ustanoveno s jakými datovými množinami budou tyto moduly pracovat. Proto je součástí MVE i knihovna modulů Vizualization, která obsahuje množinu datových struktur, která by měla vyhovovat širokému spektru aplikací z počítačové grafiky a vizualizace dat.

Filosofie MVE je velmi podobná systému VTK (Vizualization toolkit) od společnosti Kitware. Byly však provedeny nemalé úpravy pro lepší využitelnost, pochopitelnost a rozšiřitelnost. Celkově je návrh pro MVE „více objektový“. Dále se výrazně projevila snaha o podporu širokého spektra souřadných systémů a různých dimenzí. [1]

4 Nástroje video tutoriálu

4.1 Volba programu

Vycházel jsem z [2], kde je doporučován Adobe Captivate. Je to pohodlné prostředí, ve kterém jsou všechny potřebné funkce rychle k dispozici. Naučil jsem se v něm pracovat velice rychle. Jeho výhody jsou hlavně v oblasti importu. Je snadné importovat jednotlivé slidy z prezentace Microsoft Power Point, nebo importovat zvuk a ten následně upravit - to jsem využíval velice hojně. Viz kapitolu 3.2. Captivate obsahuje podporu pro stříhání importovaného zvuku.

Nevýhodou Adobe Captivate je to, že se program na video dívá spíše jako na prezentaci slidů, než jako na film. Další nevýhodou je nemožnost exportovat hotové video do jiného (rozumného) formátu než do SWF, ale vzhledem k tomu, že SWF je právě to, čeho chci dosáhnout, tak se mě tato nevýhoda netýká. Při přehrávání SWF ale bývá problém se synchronizací zvuku a obrazu, a to zvlášť u videí s velkým rozlišením a velkou kvalitou, což moje video je. Poslední nevýhodou je, že program někdy padá.

Používal jsem licenci Adobe Captivate 5 zapůjčenou z KIVu na ZČU.

4.1.1 Jiné možné programy

Program Wink je také velice vhodným nástrojem k nahrávání obrazovky. Navíc je zdarma. Pouze nemá tolik funkcí jako Captivate. Nejdou do něj importovat slidy z prezentace Microsoft Power Point a importování a stříhání zvuku přímo v programu sice podporováno je, ale mně se nepodařilo je využít tak, aby fungovaly správně. Navíc byl výsledný zvuk velmi špatné kvality.

Program ScreenCamera je komerční produkt, který umí snímat obrazovku a přidat k ní zvuk. Pravděpodobně by byl také velice vhodným kandidátem na vytvoření video tutoriálu. [2]

4.1.2 Camtasia Studio

Jako nejlepší nástroj pro nahrávání video tutoriálu se jeví komerční produkt Camtasia Studio. Tento program před ostatními zmíněnými programy vyniká intuitivitou, možnostmi různých přídavků, rychlostí, stabilitou, exportem do různých formátů.

Verze programu, kterou jsem zkoušel, byla Camtasia Studio 7.1.

Vyzdvihnu několik předností, které nemá žádný jiný ze zmíněných programů:

- Zvýraznění ukazatele myši nebo kliknutí
- Jednoduchý a rychlý export videa do formátu .mp4[1], ke kterému je automaticky generován soubor .html[2], jehož otevřením se video přehraje
- Podpora pro stříhání zvuku včetně efektů jako jsou potlačení šumu nebo přizpůsobení hlasitosti

[1] Standardní soubor videa, vhodný pro přehrání ve většině přehrávačů

[2] Standardní soubor, který se dává na web, lze ho prohlížet v internetovém prohlížeči

4.2 Nahrávání zvuku

Nejprve jsem zkoušel nahrávat zvuk běžným mikrofonem na sluchátkách, která se kupují k domácímu počítači. Výhodou bylo, že se zvuk rovnou nahrával do projektu v Adobe Captivate, ale výsledkem byl velice zašuměný zvuk.

Poté jsem si mohl z KIVu na ZČU zapůjčit kameru Canon Legria HF S21, ke které je přiložen speciální mikrofon. Nahrávání zvuku pomocí tohoto mikrofonu je o hodně kvalitnější. Šumu je v nahrávce minimálně.

Proto videa nahrávám zároveň programem Adobe Captivate a zároveň na kameru. Z videozáznamu z kamery poté programem Avidemux 2.5 oddělím zvuk a ten importuji do projektu Adobe Captivate a podle potřeby sestříhám.

Zkoušel jsem úpravu nahraného zvuku v programu Audacity 1.3 Beta, například jsem zkoušel vystřihávat koktavá místa, pokud jsem se poté opravil. Tento program je určen zejména pro úpravu zvuku a je zdarma. Poskytuje velké množství funkcí a mně se s ním pracovalo výborně. Výsledný upravený zvuk byl perfektní, nebylo poznat vystřižení kusu zvuku. Úprava výsledného zvuku ale zabere hodně času, takže jsem došel k závěru, že je rychlejší nahrát slide, ve kterém jsem koktal, znovu a již ho neupravovat.

4.3 Export videa

Výsledkem mé práce je 7 videí. (kapitol je 6, pátá kapitola je rozdělena do dvou videí)

Výsledná videa jsem exportoval do formátu SWF ve vysoké kvalitě, která se spouští přes soubor s příponou .htm. Všechna videa mají rozlišení 1280 x 1024.

Toto jsou parametry videa, které byly požadovány.

Názvy, délky (v minutách a sekundách) a velikosti výsledných videí:

-	Chapter 1 - Theoretical introduction	5:29	5,0MB
-	Chapter 2 - Introduction to MVE	12:43	96,9MB
-	Chapter 3 - Custom map creation	12:50	61,5MB
-	Chapter 4 - Advanced map execution	7:12	31,9MB
-	Chapter 5a - Custom basic module, part 1	12:53	95,0MB
-	Chapter 5b - Custom basic module, part 2	15:47	53,1MB
-	Chapter 6 - Custom special module	14:31	56,7MB

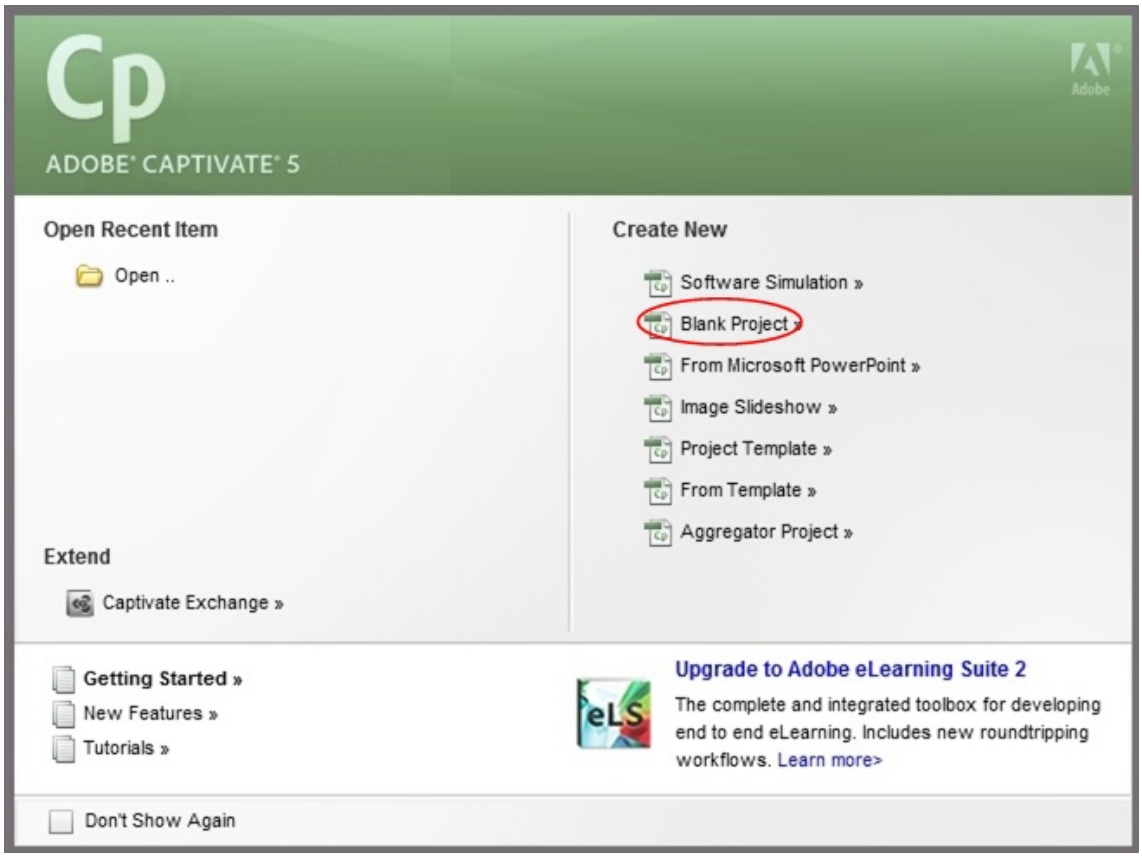
4.4 Návod na nahrávání s Adobe Captivate

Tento návod se týká Adobe Captivate verze 5.

Bude zde vysvětlen kompletní postup k vytvoření celého projektu, jeho úpravu a export hotového videa. Instalace Adobe Captivate 5 zde nebude vysvětlena.

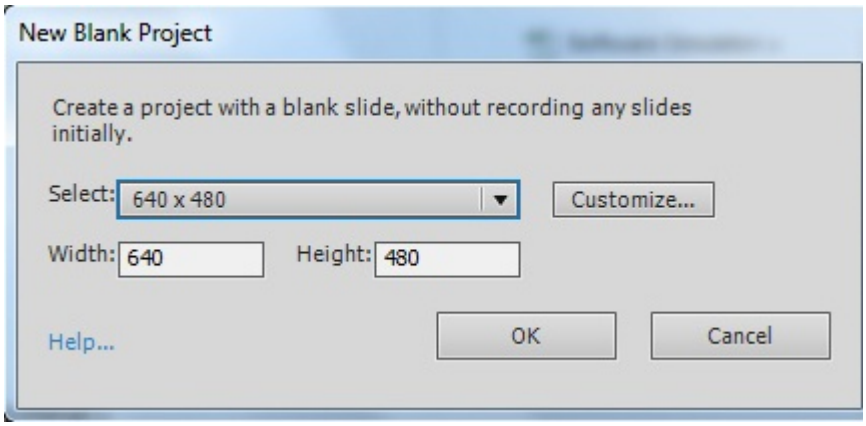
4.4.1 Tvorba nového projektu

Při spuštění Adobe Captivate se ukáže úvodní obrazovka. Klikneme na ‚Blank Project' pro vytvoření nového prázdného projektu.



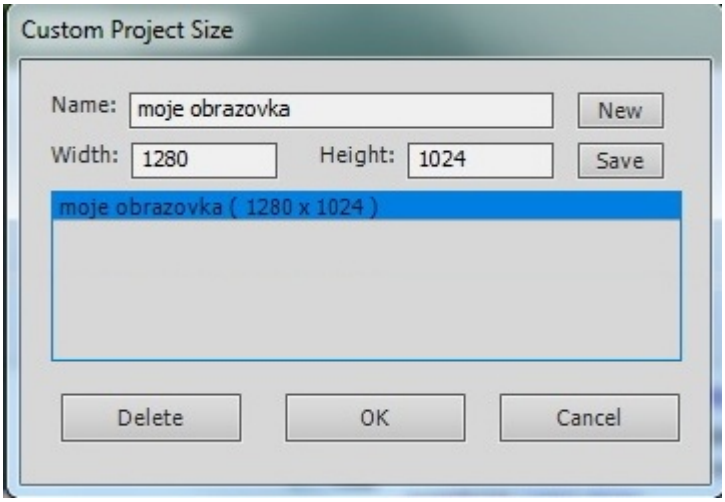
Obrázek 4: úvodní obrazovka Adobe Captivate

Ukáže se okno, kde vybíráme velikost nahrávané oblasti. Pokud chceme nahrávat celou obrazovku, klikněte na ‚Customize‘.



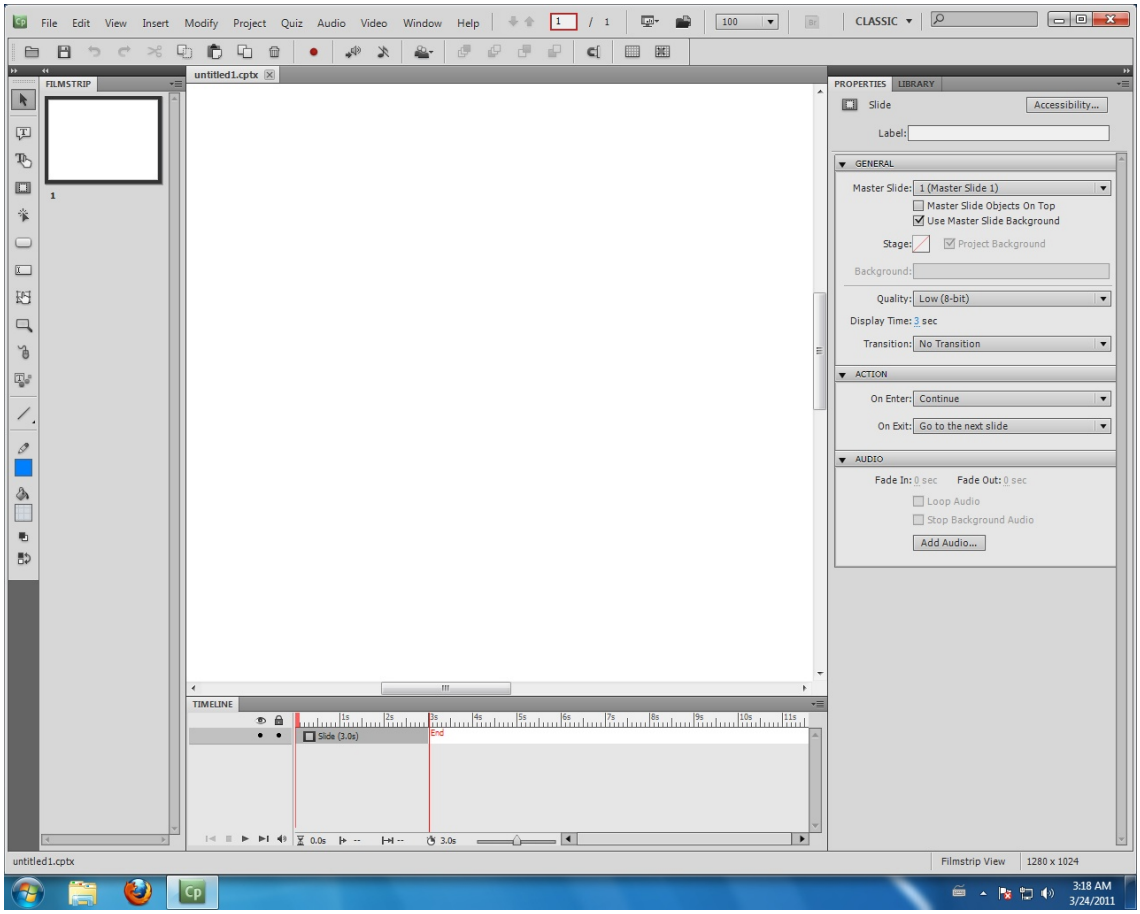
Obrázek 5: nový projekt

Do pole ‚Name‘ vyplníme název nastavení rozlišení, do pole ‚Width‘ dáme šířku rozlišení monitoru a do pole ‚Height‘ dáme výšku rozlišení monitoru. Poté klikneme na ‚Save‘ a ‚OK‘.



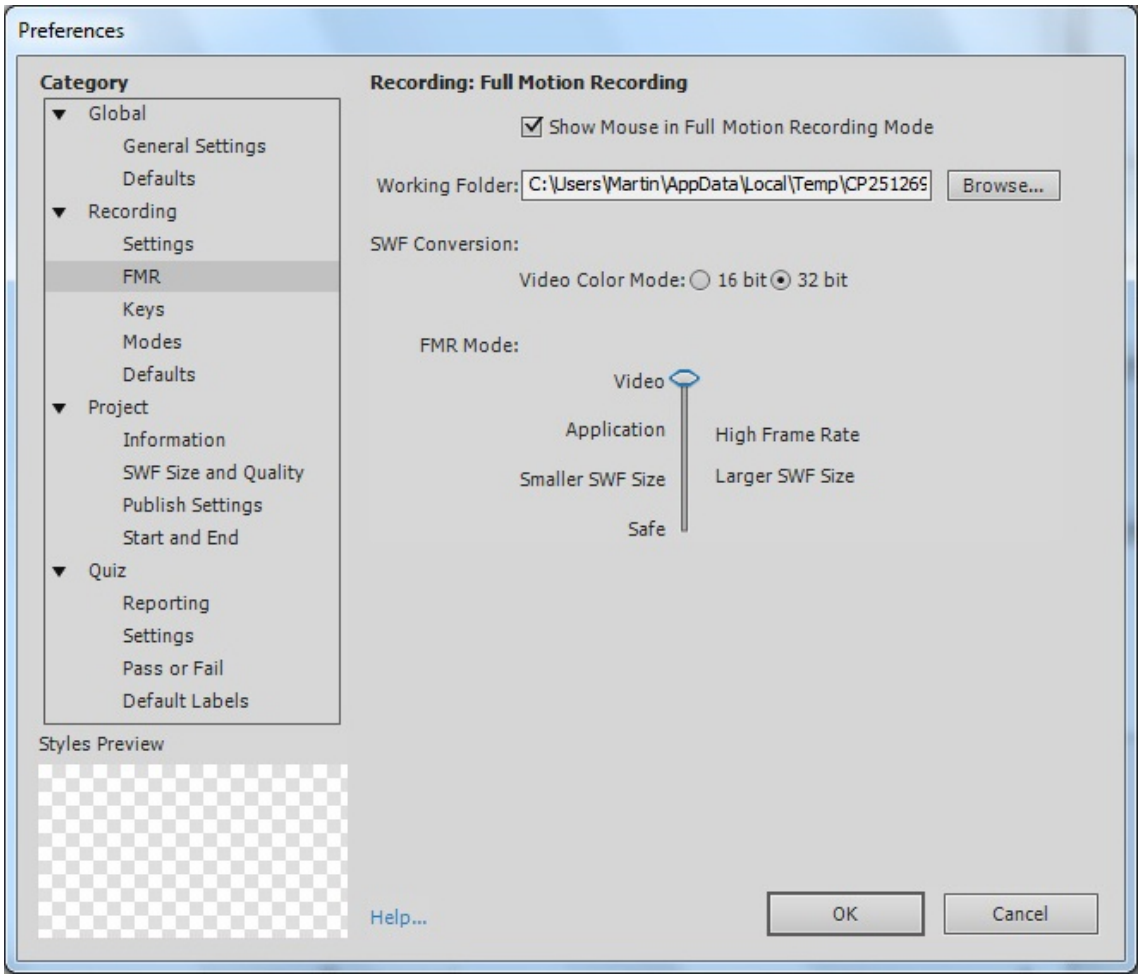
Obrázek 6: nastavení rozměrů nahrávání

Ocitáme se v prostředí Adobe Captivate.



Obrázek 7: prostředí Adobe Captivate

Kliknutím na menu ‚File' a vybráním ‚Project Info...' otevřeme menu ‚Preferences' (nastavení). Zde můžeme přizpůsobit nastavení programu.



Obrázek 8: menu 'Preferences' (nastavení)

Vlevo v nabídce ‚Category' (na obrázku 8) vybíráme kategorie nastavení. V kategoriích ‚FMR' a ‚SWF Size and Quality' vybíráme kvalitu nahrávaného videa. Mým úkolem je mít dobrou kvalitu, proto jsem nastavení v obou kategoriích nastavil na maximum.

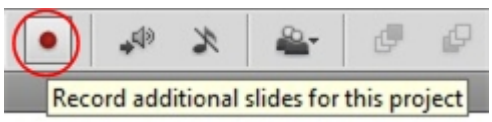
V kategorii ‚Settings' můžeme nastavit, aby se při nahrávání schovávala ikona Captivatu. Pouze v možnosti ‚Hide' zatrhneme ‚Recording Windows' (nahrávající okno), ‚Task Icon' (ikona programu) a ‚System Tray Icon' (ikona v system trayi).

Nyní jsme připraveni nahrávat.

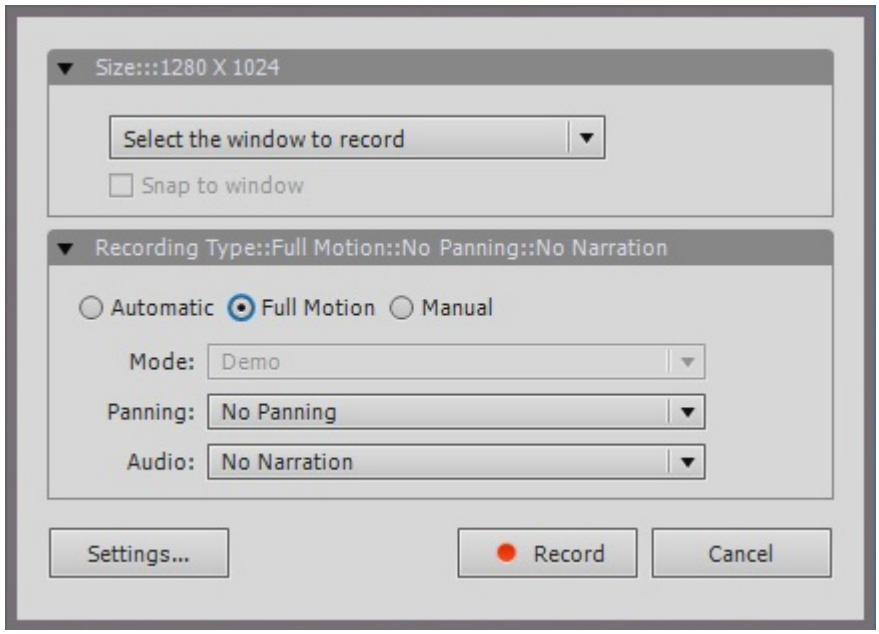
4.4.2 Nahrávání videa

V horní liště je tlačítko s červeným kruhem uprostřed. To je tlačítko nahrávání. Po stisknutí tohoto tlačítka se objeví okno (obrázek 10), ve kterém se ujistíme, že máme zatrhnutou možnost ‚Full Motion'.

Stisknutím tlačítka ‚Record' spustíme odpočítávání tří sekund, po kterých začne nahrávání.



Obrázek 9: tlačítko nahrávání



Obrázek 10: okno před nahráváním

Na ukončení nahrávání je implicitně nastavena klávesa End.

Tím jsme nahráli první slide. Těchto slidů může být ve videu libovolný počet. Pořadí jednotlivých slidů ve videu se dá měnit pouhým přetáhnutím myši v liště ‚FILMSTRIP' na levé straně obrazovky (viz obrázek 7).

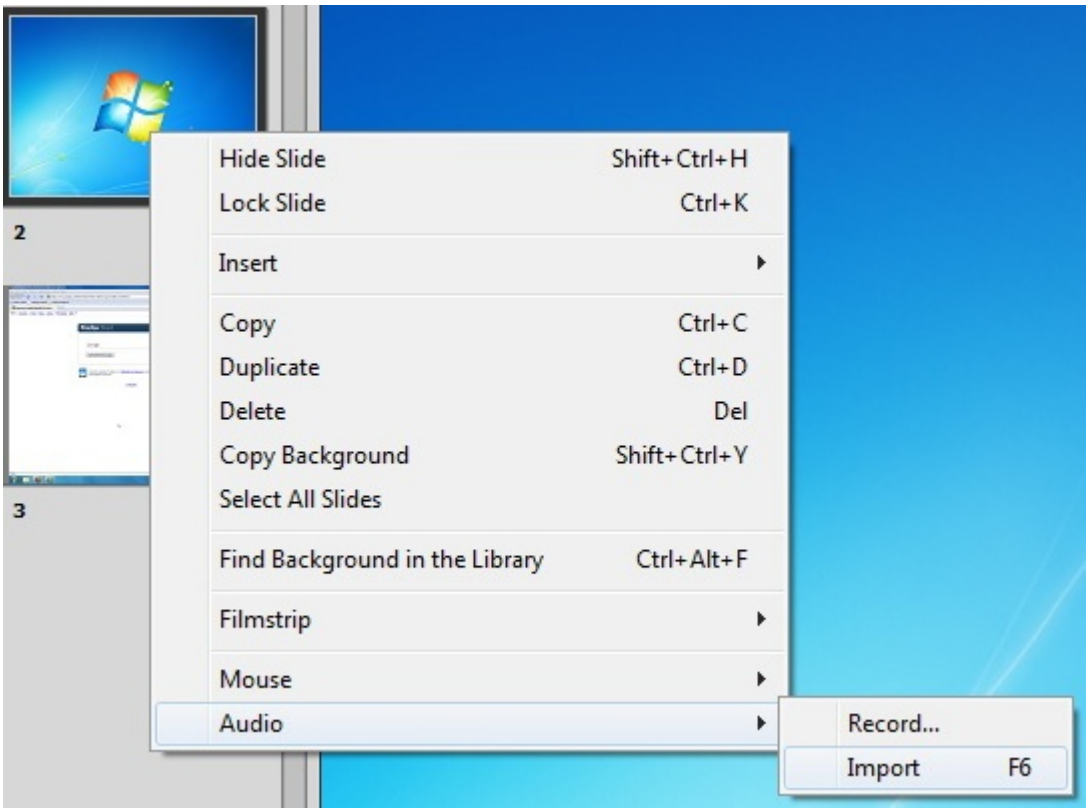
4.4.3 Nahrávání a editace zvuku

Pro importování zvuku klikneme na slide v liště ‚FILMSTRIP' pravým tlačítkem, vybereme možnost ‚Audio' a ‚Import'. Viz obrázek 11.

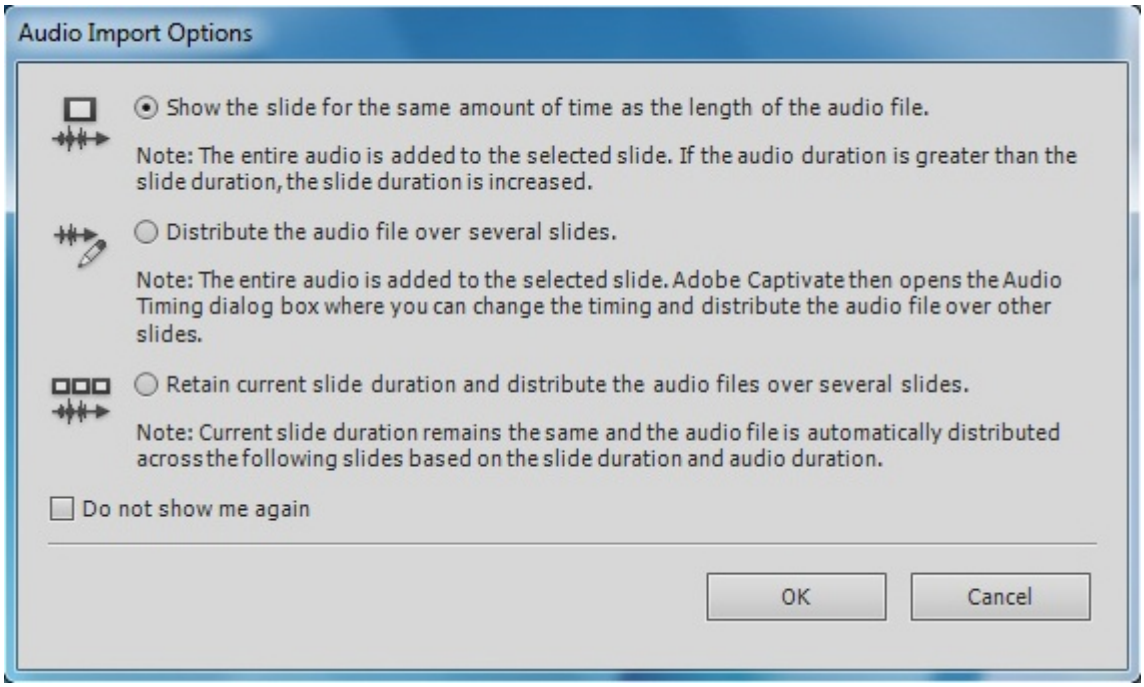
Pokud je importovaný zvuk delší, než je délka slidu, zobrazí se okno ‚Audio Import Options'. V něm máme na výběr ze tří možností, které popíši odshora dolů (viz obrázek 12). První možností je, že se délka slidu prodlouží na délku zvuku.

Druhou možností je, že je zvuk přidán a my ručně upravíme časování slidů, takže zvuk může zasahovat do více slidů.

Třetí možností je, že zvuk bude opět celý přidán, takže bude zasahovat do více slidů, ale vše udělá Captivate automaticky a nezmění ani časování slidů ani délku zvuku.



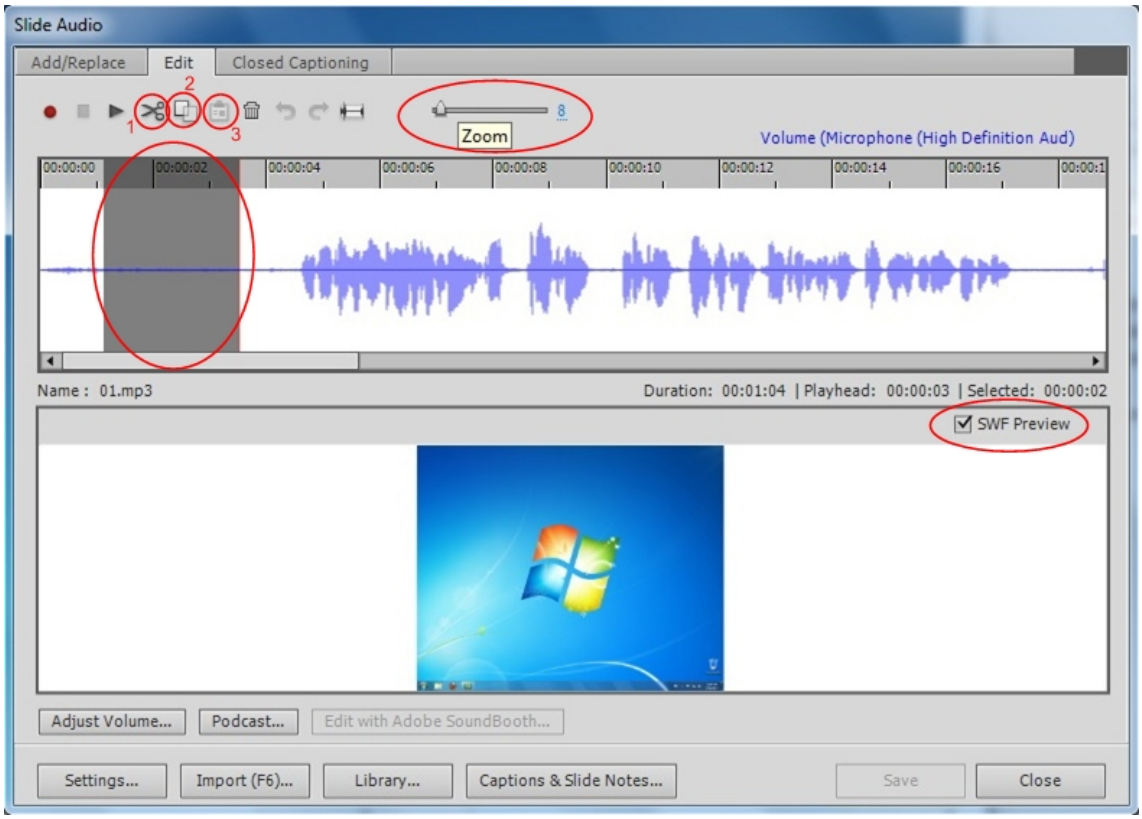
Obrázek 11: import zvuku



Obrázek 12: okno 'Audio Import Options'

Captivate nabízí podporu pro stříhání zvuku. Stejným způsobem kterým jsme ke slidu importovali zvuk, můžeme zvuk editovat, jen místo ‚Import' klikneme na ‚Edit'.[3]

[3] Editovat zvuk lze jen u slidu, který již nějaký zvuk má. Jestli již slide zvuk má, uvidíte pod slidem ikonku reproduktoru.



Obrázek 13: stříhání zvuku

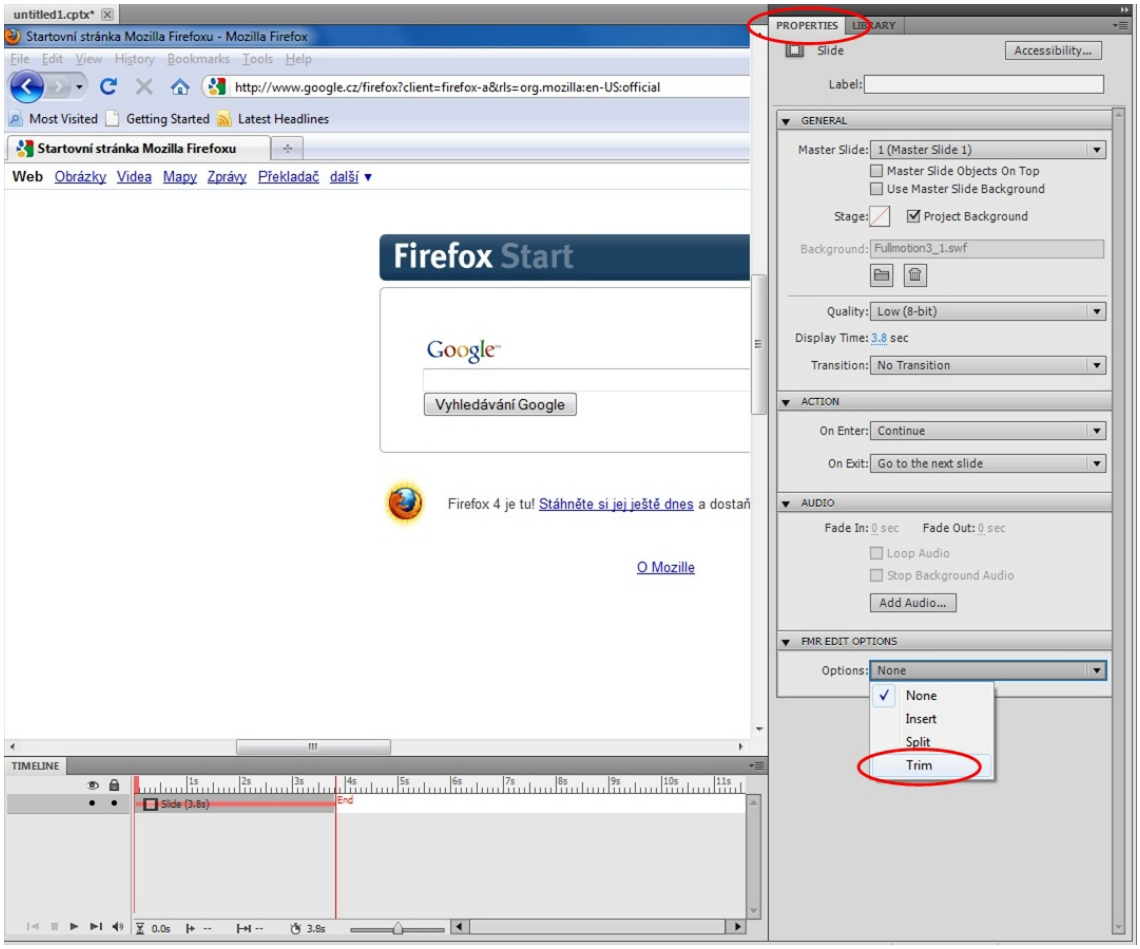
Zobrazí se nám okno stříhání zvuku jako na obrázku 13. Velice důležitá věc je nastavit si správně přiblížení, neboli ‚Zoom'. Když pojedeme posuvníkem doleva, oddálíme pohled na zvuk, doprava ho přiblížíme.

Pro správné načasování a sesynchronizování zvuku a obrazu dohromady si zapneme volbu ‚SWF Preview'. Poté až budeme zvuk přehrávat (trojúhelník vlevo nahoře), spolu s ním uvidíme i malé video našeho slidu.

Na obrázku 13 vidíme také tmavý obdélník. To je oblast zvuku, kterou jsme označili. Označení se dělá pouhým přetažením myši zleva doprava, nebo zprava doleva. Poté, co máme oblast označenou, můžeme ji smazat stiskem klávesy delete, vyjmout (kroužek 1 na obrázku 13), kopírovat (kroužek 2 na obrázku 13), nebo vložit (kroužek 3 na obrázku 13).

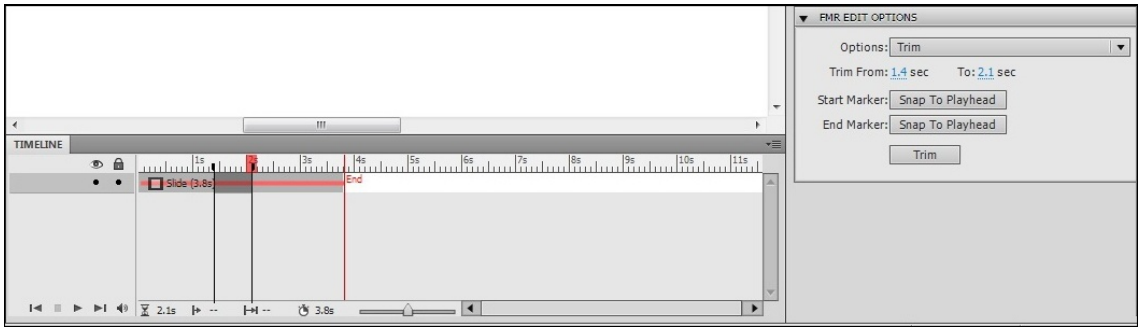
4.4.4 Editace videa

Lze upravovat i video slidu. K tomu slouží možnost ‚Trim'. Na pravém panelu vybereme kartu ‚PROPERTIES' a v ní dole v nabídce ‚Options' vybereme ‚Trim'.



Obrázek 14: vybrání možnosti 'Trim'

Nyní můžeme označit libovolnou část videa. Označenou část měníme buď oběma tlačítky ‚Snap To Playhead‘, nebo tím, že označenou část přesuneme přetažením myši. Označenou část nakonec odstraníme stisknutím tlačítka ‚Trim‘.



Obrázek 15: označení oblasti ve funkci 'Trim'

4.4.5 Export videa

Celé video můžeme nyní shlédnout stisknutím klávesy F4.

Pro export videa vybereme nabídku ‚File‘ a klikneme na ‚Publish...‘. Vybereme možnost ‚Flash(SWF)‘. Do políčka ‚Project Title‘ napíšeme jméno našeho videa. Políčko ‚Export To HTML‘ doporučuji nechat zaškrtnuté, protože tímto způsobem video přehraje každý. Znamená to, že výsledkem bude video ve formátu .swf, ale spouštět se bude přes automaticky vytvořený soubor .htm. Nyní již tlačítkem ‚Publish‘ vytvoříme požadované video.

5 Kapitola 1 – Teoretický úvod

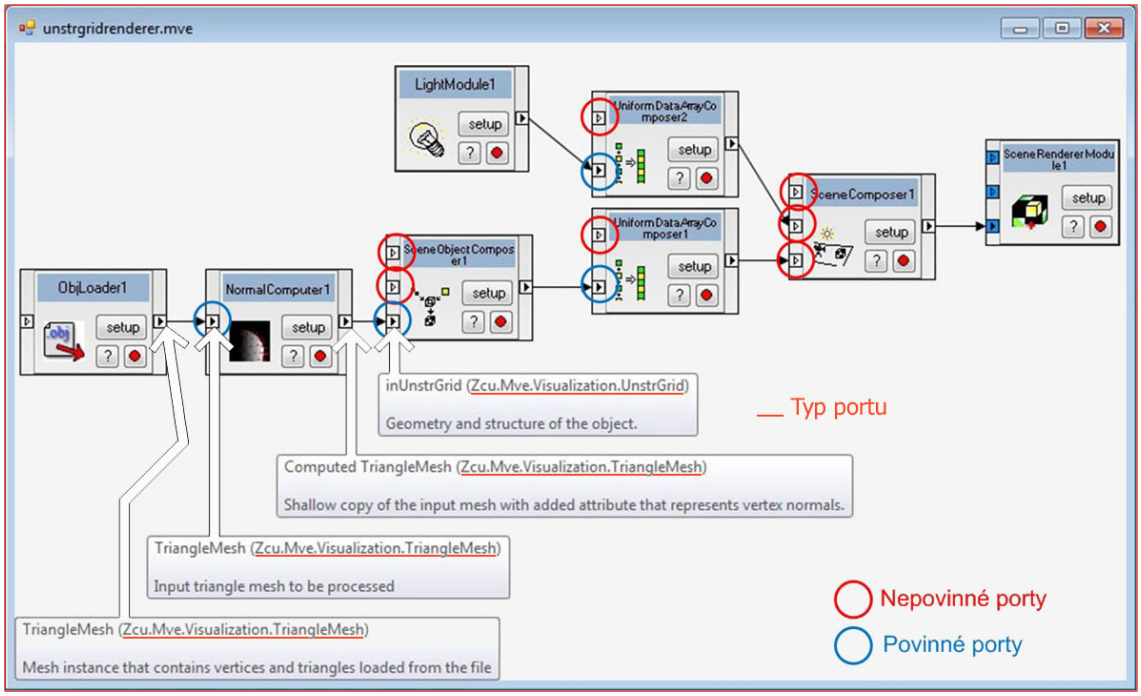
Tato kapitola sestává ze dvou částí. První část mluví obecně o modulárním programování a druhá část seznamuje se základními vlastnostmi Modulárního vizualizačního prostředí – 2 (MVE). První část kapitoly je zároveň kapitolou 2 v této práci (viz kapitolu 2: O modulárním programování).

5.1 Seznámení s MVE

MVE je uživatelsky přívětivé prostředí, které umožňuje modulární programování. Je vyvíjeno na Západočeské univerzitě v Plzni.

Obsahuje grafické uživatelské prostředí, ve kterém z modulů konstruujete mapy, dále pak množství již hotových modulů a podporu pro spouštění map z příkazové řádky.

MVE je založeno na technologii .NET.



Obrázek 16: příklad mapy z MVE

Vstupní porty se dělí na povinné a nepovinné. Všechny povinné vstupní porty musí být zapojeny, aby bylo možné mapu spustit. Na obrázku 16 je vidět, že nepovinné porty se zobrazují jako prázdné trojúhelníky, povinné jako vyplněné trojúhelníky.

Každý vstupní i výstupní port má určený datový typ. Datové typy dvou spojených portů musí být vždy stejné. Datové typy se mohou navzájem dědit, proto existuje výjimka tohoto pravidla, a to tehdy, pokud jsou spojeny vstupní a výstupní port, kde datový typ výstupního portu je potomkem datového typu vstupního portu.

MVE obsahuje množství již hotových modulů. Jsou to zejména základní moduly pro numerické operace, moduly pro práci s řetězci a moduly pro vizualizaci dat. Vlastní moduly se také dají velice snadno přidat.

MVE můžete získat na internetové adrese `http://mve.kiv.zcu.cz`

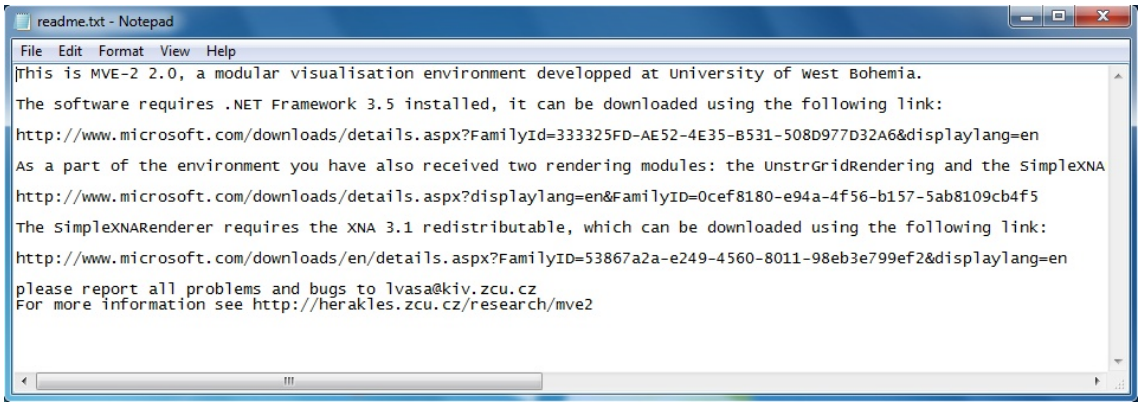
6 Kapitola 2 – Úvod do MVE

Tato kapitola se zabývá návodem na stažení, instalaci a spuštění MVE, prvními kroky v prostředí MapEditor, jsou ukázány některé hotové mapy a na nich jsou předvedeny základní funkce prostředí MapEditor. Na závěr je ukázáno jak mapy pomocí příkazové řádky spouštět v programu RunMap (který je součástí MVE).

6.1 Stažení a instalace

MVE stáhneme z adresy mve.kiv.zcu.cz, kde klikneme nejprve na odkaz ‚Main download', poté u nadpisu ‚MVE 2.0' klikneme na odkaz ‚zip'. Tím stáhneme zip soubor[4], který rozbalíme a tím jsme získali kompletní MVE.

Některé volně stažitelné programy jsou k běhu MVE buď potřeba, nebo se pro něj doporučují. Proto otevřeme soubor readme.txt, který jsme získali rozbalením zip souboru. V něm najdeme odkazy pro stažení těchto programů.



Obrázek 17: soubor readme.txt

První odkaz stáhne .NET Framework 3.5, který je nutně potřeba k běhu MVE. Ve Windows 7 je již .NET Framework 3.5 obsažen, takže není potřeba ho instalovat.

Druhý odkaz stáhne DirectX a třetí odkaz stáhne XNA 3.1. Tyto programy nejsou nutně potřeba k běhu MVE, ale potřebují je některé moduly pracující s grafikou, které již jsou obsaženy v MVE.

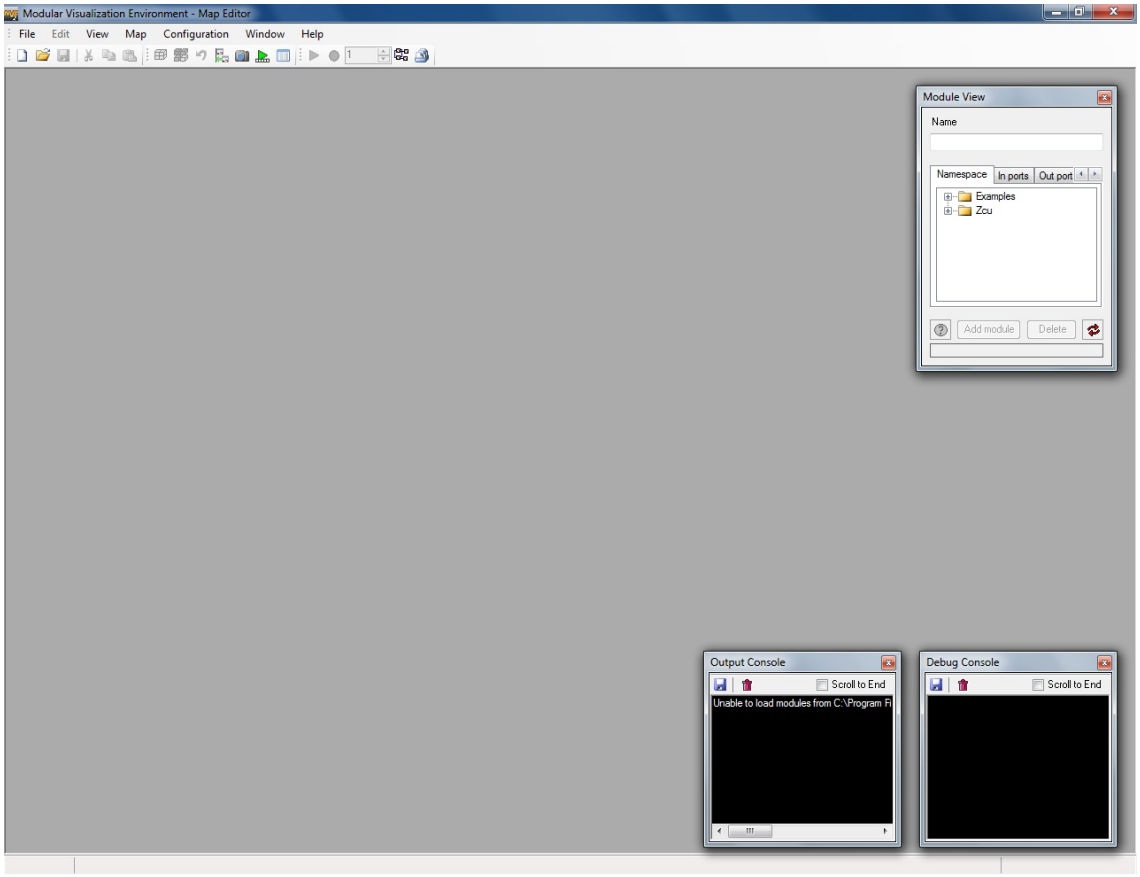
Nainstalujeme stáhnuté programy.

Nyní je vše připraveno pro spuštění MVE, respektive MapEditoru.

6.2 Příklady hotových map

Ve složce Mve2-bin spustíme soubor MapEditor.exe. Objeví se prostředí, ve kterém budeme editovat a spouštět naše mapy.

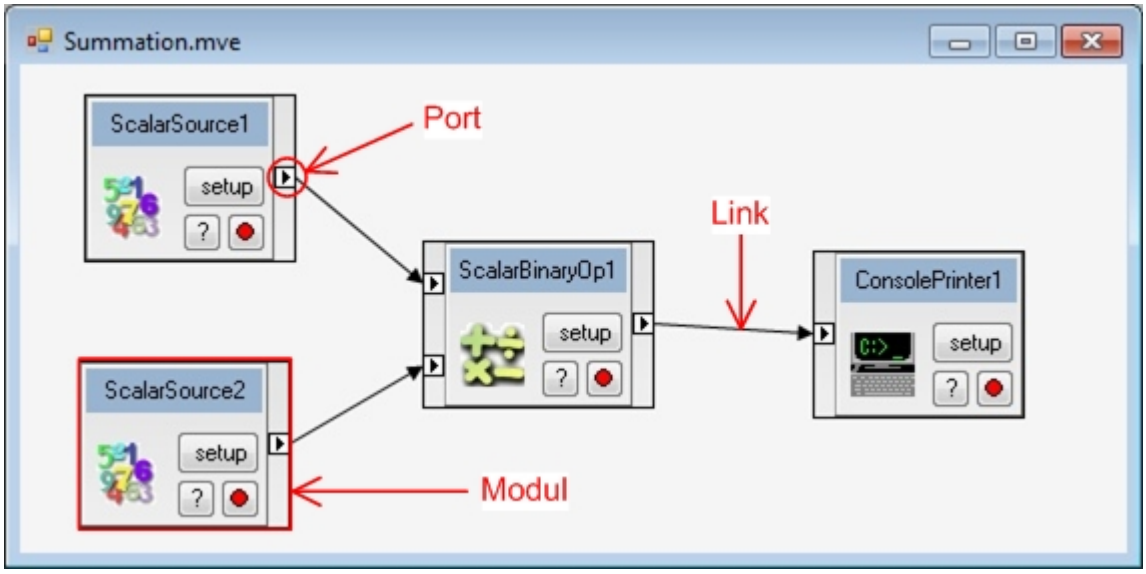
[4] Soubor s příponou .zip, může obsahovat libovolné soubory a složky, které jsou v něm komprimované (zabalené), po dekomprimaci (rozbalení) získáme původní soubory a složky



Obrázek 18: prostředí MapEditor

6.2.1 Mapa Summation

Klikneme na ‚otevřít' (open) a najedeme do adresáře s MVE, poté Mve2-bin, poté adresář maps a otevřeme soubor Summation.mve. Otevře se hotová mapa, kterou již nemusíme nijak upravovat. Tato mapa dokáže sčítat desetinná čísla.



Obrázek 19: mapa summation

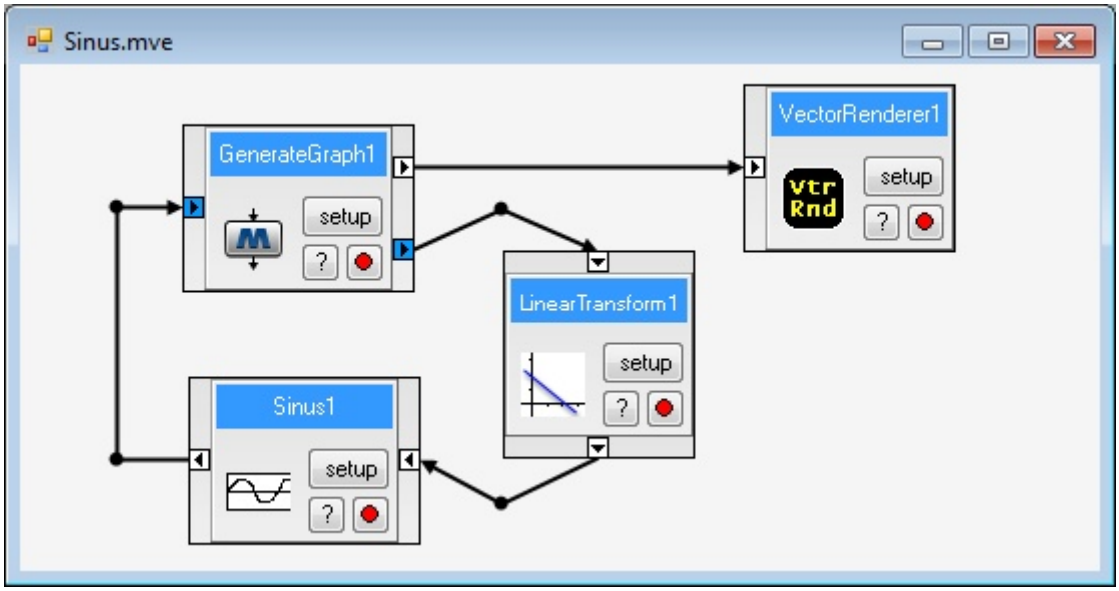
V ní vidíme jednotlivé moduly (velké ‚krabičky'), které obsahují porty (malé čtverce s trojúhelníky), a linky, neboli spojení, která mají na starost propojení jednotlivých

portů. Mapu můžeme spustit kliknutím na ikonu zeleného trojúhelníku v nabídce MapEditoru. Výsledek se vypíše do konzole Output Console.

Můžeme změnit nastavení jednotlivých modulů tím, že klikneme na tlačítko setup na modulu. V tomto případě například změníme nastavení dvou modulů vlevo na obrázku 19 tak, aby jeden generoval číslo 4 a druhý číslo 5. Pokud nyní znovu spustíme mapu, do Output Console se vypíše číslo 9.

6.2.2 Mapa Sinus

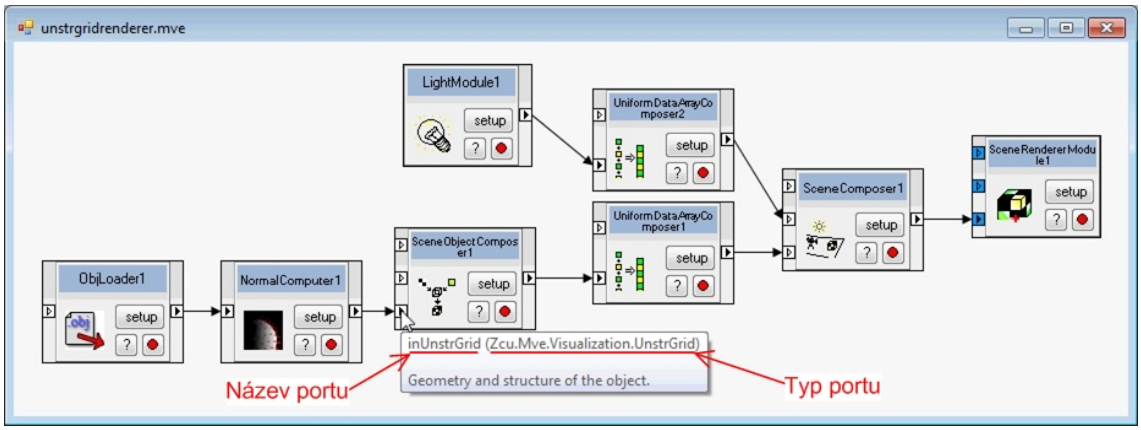
Stejným způsobem jako v předchozím případě otevřeme mapu Sinus.mve. Tato mapa vypočítá a ukáže graf funkce sinus.



Obrázek 20: mapa sinus

Můžeme využívat spoustu funkcí abychom mapu udrželi přehlednou. Můžeme moduly přesouvat, otáčet je, můžeme přesouvat spojení, přesouvat jejich kontrolní body – to jsou body, ve kterých se čára spojení láme. Můžeme také vytvářet nové kontrolní body. Vše je jednoduché a intuitivní.

Porty se dělí na vstupní a výstupní. Poznáme je tak, že trojúhelník vstupního portu směřuje dovnitř modulu a trojúhelník výstupního portu směřuje ven z modulu.



Obrázek 21: mapa unstrgridrenderer

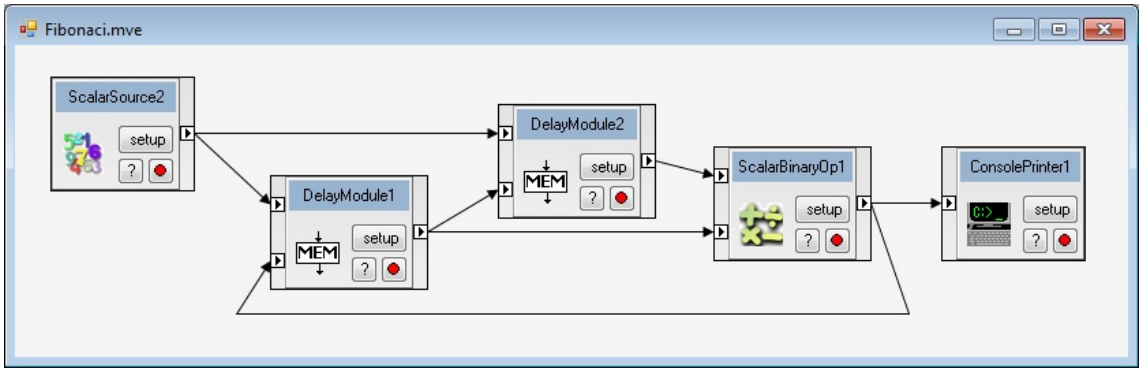
6.2.3 Mapa unstrgridrenderer

Na této mapě si ukážeme, že mezi moduly mohou být přesouvány i složité datové struktury. Tato mapa vykreslí a ukáže 3D objekt, kterým lze otáčet.

Otevřeme mapu unstrgridrenderer.mve. Z obrázku 21 je vidět, že když ukážeme na port, ukáže se bublina, která nám oznamuje název portu, typ portu a jeho popis.

6.2.4 Mapa Fibonaci

Na následující mapě se seznámíme s funkcí vícenásobného spuštění mapy. Otevřeme mapu Fibonaci.mve. Tato mapa počítá Fibonacciho čísla.



Obrázek 22: mapa fibonaci

Tato mapa je navržena tak, aby při každém svém spuštění vypsala pouze jedno Fibonacciho číslo. Proto teď použijeme funkci vícenásobného spuštění mapy – číslo v nabídce MapEditoru, které je již vyplněno na hodnotu 10. Proto když nyní jednou klikneme na zelený trojúhelník (spuštění mapy), vypíše se do Output Console prvních 10 Fibonacciho čísel.

6.3 Rozhraní RunMap

Hotová mapa se dá spustit i jinak, než v prostředí MapEditor. Dá se spustit v příkazové řádce pomocí rozhraní RunMap.

Spustíme příkazovou řádku – ve Windows 7 to uděláme kliknutím na Start, napsáním cmd a stisknutím enter. V příkazové řádce jdeme do adresáře, kde máme MVE, poté do adresáře Mve2-bin. K přesouvání mezi adresáři slouží v příkazové řádce příkaz cd. Pokud chceme o jeden adresář nahoru, napíšeme cd .. (mezera a dvě tečky)

Napište RunMap.exe (mezera) maps (zpětné lomítko) (název mapy)

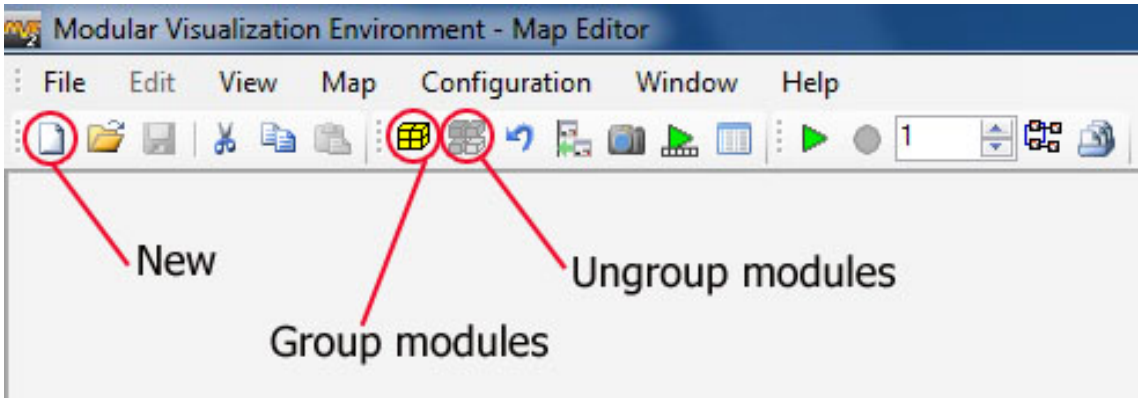
Např.: `RunMap.exe maps\unstrgridrenderer.mve`

Tím se spustí mapa stejně, jako bychom jí spustili v MapEditoru, ale bez nutnosti ho spouštět. Nelze ale tímto způsobem mapu měnit.

7 Kapitola 3 – Konstrukce vlastní mapy

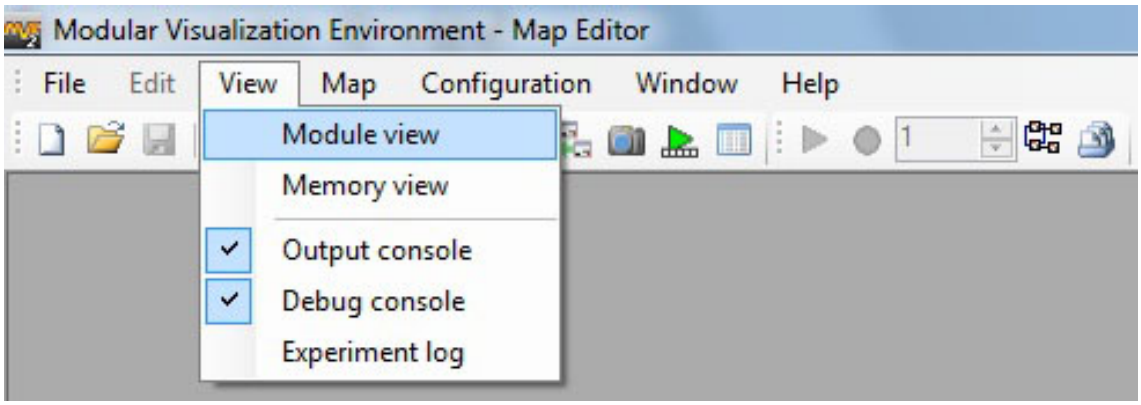
Obsahem této kapitoly je návod na vytvoření nové mapy, seznámení s některými speciálními moduly a se slučováním modulů.

7.1 Konstrukce základních map



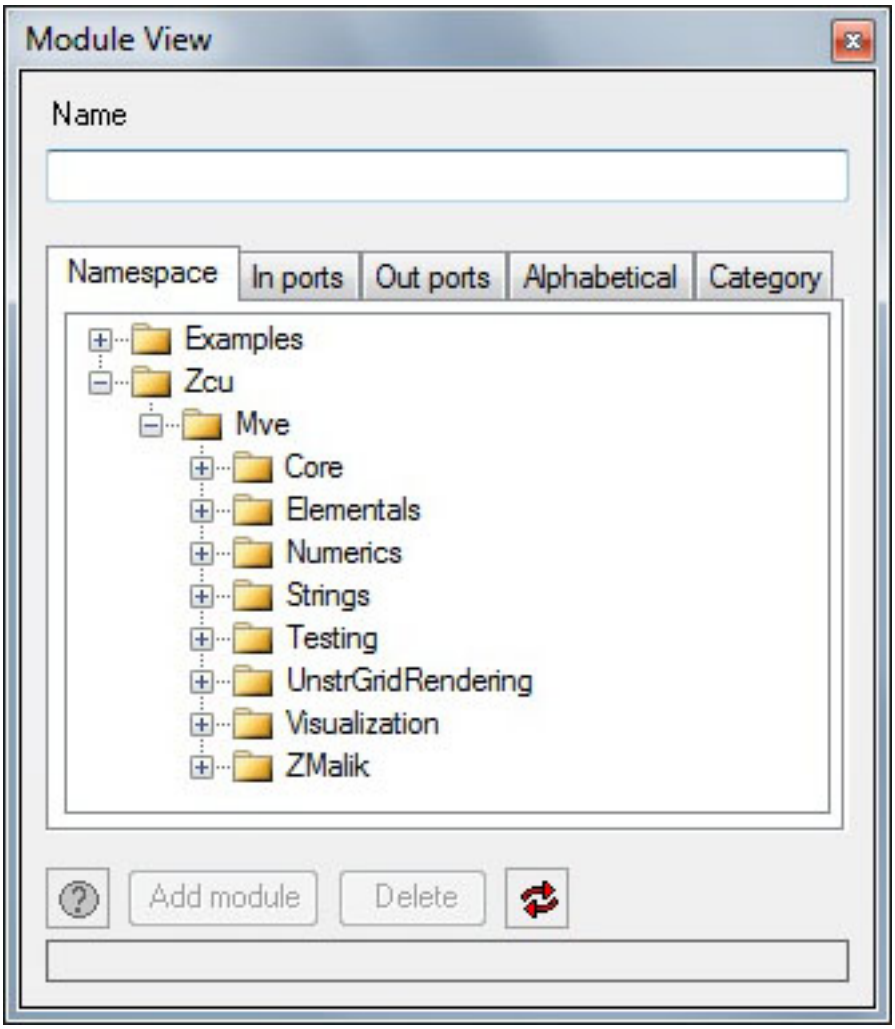
Obrázek 23: nástrojová lišta MapEditoru

Kliknutím na tlačítko ‚New' (viz obrázek 23) se vytvoří nová prázdná mapa, do které lze pak přidávat moduly. Všechny moduly lze najít v okně Module View, kde jsou řazeny podle jmenného prostoru (karta ‚Namespace'), vstupních portů (karta ‚In ports'), výstupních portů (karta ‚Out ports'), abecedy (karta ‚Alphabetical') nebo kategorie (karta ‚Category') – viz obrázek 25. Okno Module View lze zobrazit nebo schovat volbou ‚View' a poté ‚Module View' (viz obrázek 24).

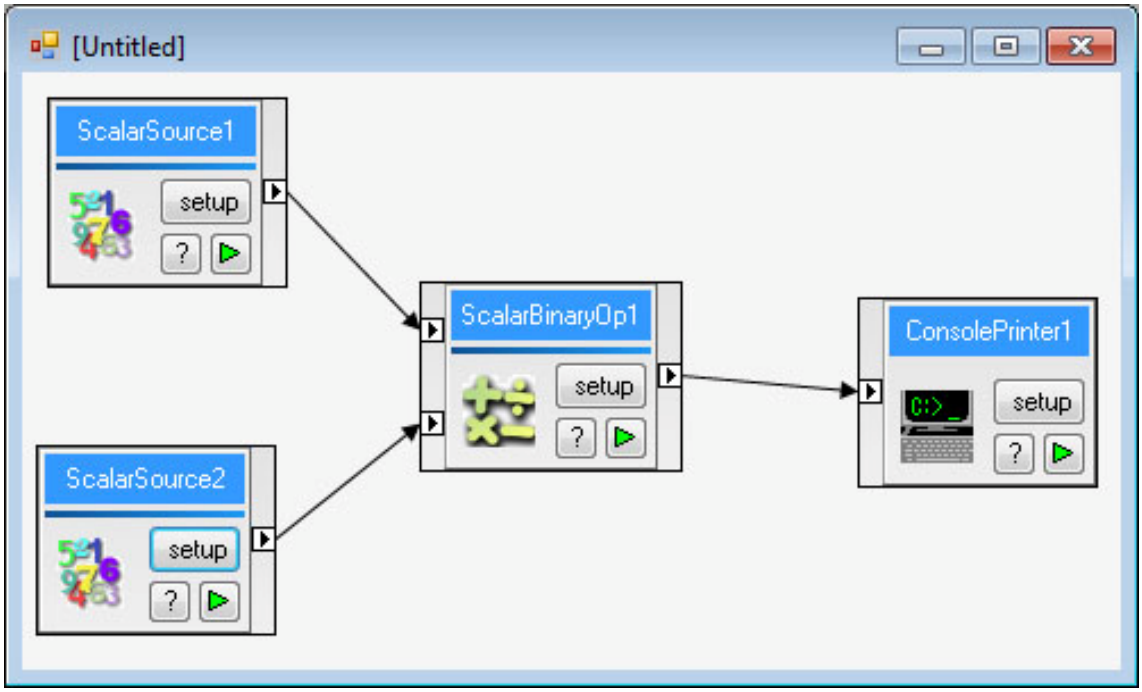


Obrázek 24: zobrazení okna Module View

Mapu vytvoříme tak, že přidáme do mapy námi zvolené moduly, a ty pospojujeme spojeními (linky). Pak můžeme mapu spustit.



Obrázek 25: Module View



Obrázek 26: mapa sčítající 2 čísla

První mapa, kterou budeme vytvářet, je velice jednoduchá. Umožňuje sečtení dvou desetinných čísel. Abychom ji mohli vytvořit, otevřeme si jmenný prostor Zcu.Mve.Numerics, ve kterém se nacházejí moduly, které pracují s čísly.

Do naší zatím prázdné mapy přidáme 2 moduly ScalarSource, které slouží ke generování čísla, 1 modul ScalarBinaryOp, který provádí matematickou operaci se dvěma desetinnými čísly a 1 modul ConsolePrinter, který vypisuje do výstupní konzole. Viz obrázek 26.

Můžeme vytvořit tu samou mapu, jen s tou úpravou, že bude pracovat s celými čísly. Moduly ScalarSource tedy nahradíme moduly IntegerSource a modul ScalarBinaryOp nahradíme IntegerBinaryOp.

Mapu můžeme uložit v menu ‚File' v nástrojové liště MapEditoru vybráním volby ‚Save As'.

7.2 Moduly ve jmenném prostoru ‚Strings'

Moduly v tomto jmenném prostoru pracují s řetězci. Nás budou zajímat jen moduly ‚StringSource' a ‚DirectoryLister'.

Modul ‚StringSource' je velice jednoduchý modul, který generuje řetězec, který má zadán ve svém nastavení (setup).

Modul ‚DirectoryLister' generuje řetězec, který odpovídá plnému názvu souboru, který se nalézá v určité složce. Funguje tak, že v jeho nastavení nastavíme složku, kterou bude procházet, a na jeho vstupní port připojíme celé číslo, které udává, kolikátý soubor má být ten, jehož jméno je vygenerováno jako řetězec.

7.3 Speciální moduly

Obrázek 27: speciální moduly

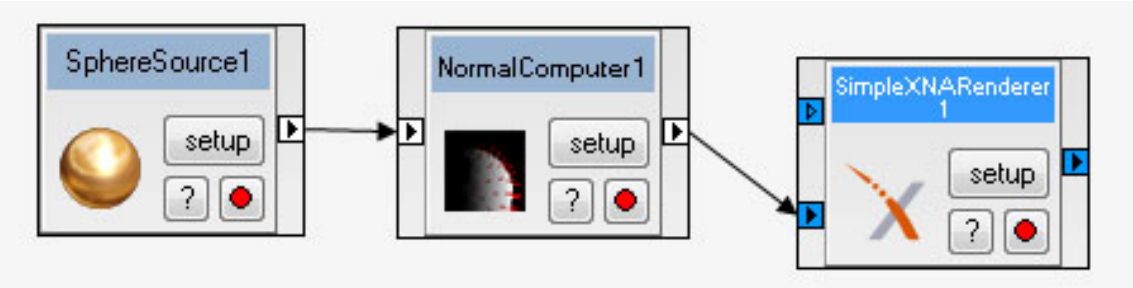
Modul XmlSaver je zvláštní tím, že se jeho výstupní port objeví až poté, co je k němu připojen vstupní port. Nově objevený výstupní port bude stejného typu, jako vstupní port a bude obsahovat stejné informace. Jediné, co tento modul udělá je, že data, která skrz něj projdou, uloží do .xml souboru. Proto se tento modul dá vložit kamkoliv do mapy, a to z něj činí dobrý nástroj na debugování mapy.

Modulu XmlLoader se zase změní typ výstupního portu až poté, co v jeho nastavení nastavíme soubor, ze kterého xml data načítá. Typ portu je poté shodný s typem dat v souboru.

Každý datový typ v MVE má svojí metodu, která kontroluje soudržnost, či správnost dat. DataChecker spouští právě tuto metodu a výsledek vypíše do Output Console.

7.4 Jmenný prostor Visualization

V této podkapitole budou probrány pouze základní vlastnosti a moduly z tohoto jmenného prostoru. Více informací o něm budou probrány v kapitole 9 video tutoriálu MVE, která není součástí této práce.



Obrázek 28: mapa zobrazující kouli

Ve jmenném prostoru Zcu.Mve.Visualization se nachází další důležité jmenné prostory:

- Sources – zde jsou moduly generující geometrické objekty, moduly nic nenačítají ze souborů
- Loaders – zde jsou moduly generující složité objekty, které načítají ze souborů
- ZMalik – zde se nachází pouze jeden modul – SimpleXNARenderer, což je modul, který vykreslí objekt, který dostane na vstup
- Processing – zde se nachází modul NormalComputer spolu s dalšími moduly, které nějak zpracovávají proud grafických dat

Příklad na obrázku 28 je mapa, která zobrazí kouli. Koule je nejprve vygenerována v modulu SphereSource, poté jsou k ní spočteny normály modulem NormalComputer. Zde nebudeme vysvětlovat podrobnosti o normálách, pouze řekneme, že je důležité, aby je každý grafický objekt měl. A poté je objekt vykreslen modulem SimpleXNARenderer.

7.5 Groupy, neboli slučování modulů

Když označíme více modulů, můžeme je graficky sloučit do jedné tzv. groupy stisknutím tlačítka ‚Group modules' (viz obrázek 23). Groupa je vlastně jen modul s ikonou groupy. Funkčnost zůstane stejná, jen se tyto moduly zobrazí jako groupa, která má všechny výstupní porty a nezapojené vstupní porty jako měly všechny moduly, které jsou v ní zahrnuty.

V groupě můžeme i nastavovat nastavení všech modulů, které jsou součástí groupy.

Pokud chceme zrušit sloučení modulů, označíme groupu a klikneme na ‚Ungroup modules' (viz obrázek 23).

Groupu můžeme uložit když na ni klikneme pravým tlačítkem a vybereme ‚Save Group...'. Tím se objeví v okně Module View. Z něj ji můžeme smazat, pokud ji vybereme a klikneme na tlačítko ‚Delete'.

8 Kapitola 4 – Pokročilé spouštění mapy

V této kapitole se dozvíme jak dokáží některé moduly ovlivňovat svoje nastavení, pokud jsou spuštěny víckrát, dále pak jak pracovat s funkcí program a poslední část se zabývá funkcí Experiment Log.

8.1 Běžné vícenásobné spuštění modulu

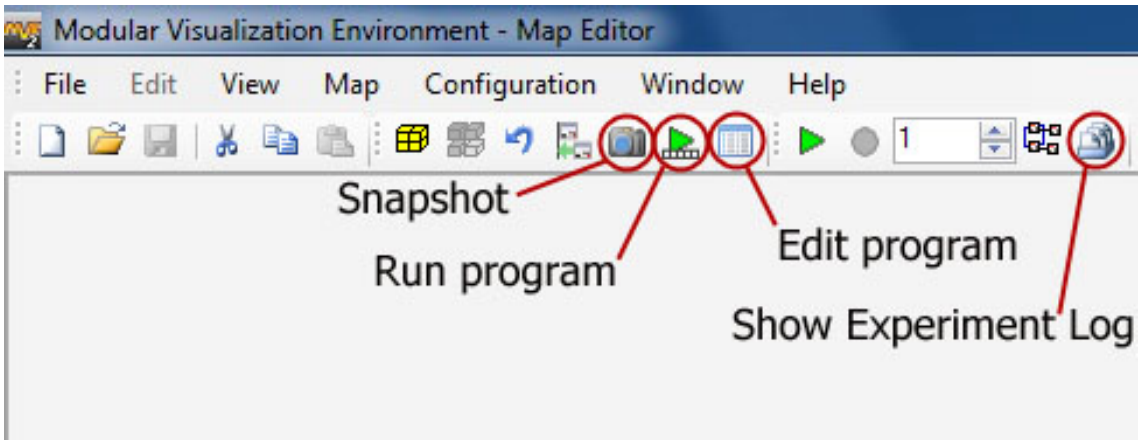
Některé moduly, jako například Counter, si pamatují svoje nastavení z předešlého spuštění a dokáží ho měnit. Například pokud dáme Counter do mapy, kterou spustíme víckrát pomocí funkce vícenásobného spuštění, tak dá Counter v každém spuštění na svůj výstup číslo o jedna větší.

Většina modulů ale svoje nastavení nemění, a proto zůstává nastavení v každém spuštění mapy stejné.

Naproti tomu ve funkci program toto nastavení měnit můžeme.

8.2 Funkce program

V této funkci si můžeme připravit dopředu libovolný počet spuštění mapy, které spustíme stiskem jediného tlačítka. Navíc můžeme měnit nastavení každého modulu v každém spuštění mapy zvlášť.



Obrázek 29: nástrojová lišta MapEditoru

Klikneme na tlačítko ‚Edit program'. Viz obrázek 29.

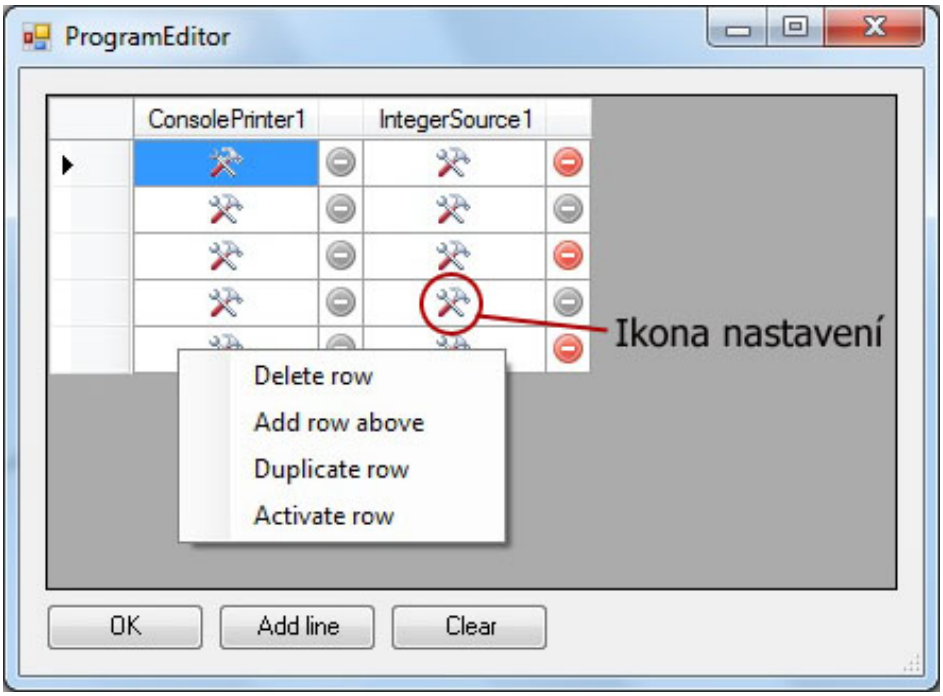
Ukáže se vám okno Program editoru jako na obrázku 30, jen prázdné. Nové řádky programu můžeme přidat stiskem tlačítka ‚Add line'.

Program má tolik sloupců, kolik modulů má mapa. Každý modul má svůj sloupec, a tím i svoje nastavování. Každá řádka odpovídá jednomu spuštění mapy.

Pokud v Program editoru změníme nastavení některého modulu v některém spuštění mapy (to uděláme kliknutím na příslušnou ikonu nastavení – viz obrázek 30), pak je tato změna nastavení indikována zčervenáním ikony mínusu vpravo od ikony nastavení. Tuto změnu můžeme zrušit a tím vrátit nastavení na výchozí hodnoty kliknutím na tuto červenou ikonu mínusu, což je indikováno tím, že ikona mínusu zšedne.

Když klikneme na některou z řádek pravým tlačítkem, objeví se kontextová nabídka nabízející možnosti:

- ‚Delete row' – odstranit řádku
- ‚Add row above' – přidat řádku s původním nastavením nad tuto řádku
- ‚Duplicate row' – přidat řádku s nastavením stejným, jako má tato řádka, pod tuto řádku
- ‚Activate row' – změnit aktuální nastavení mapy do podoby nastavení této řádky



Obrázek 30: okno Program editoru

Na obrázku 30 vidíme, že bylo změněno nastavení modulu IntegerSource1 v prvním, třetím a pátém spuštění mapy. Ve druhém a čtvrtém spuštění zůstalo původní nastavení. Výsledkem bude to, že se mapa poprvé a podruhé spustí s nastavením prvního řádku, potřetí a počtvrté s nastavením třetího řádku a popáté s nastavením pátého řádku.

Tlačítko ‚Clear' vymaže všechny řádky z Program editoru.

Stiskem tlačítka ‚OK' uložíme program a poté ho tlačítkem ‚Run program' v liště MapEditoru (viz obrázek 29) spustíme.

Tlačítko ‚Snapshot' v liště MapEditoru (viz obrázek 29) slouží k vytvoření nové řádky v programu, která je přidána na konec programu. Tato řádka má nastavení stejné jako měla mapa v době zmáčknutí tlačítka.

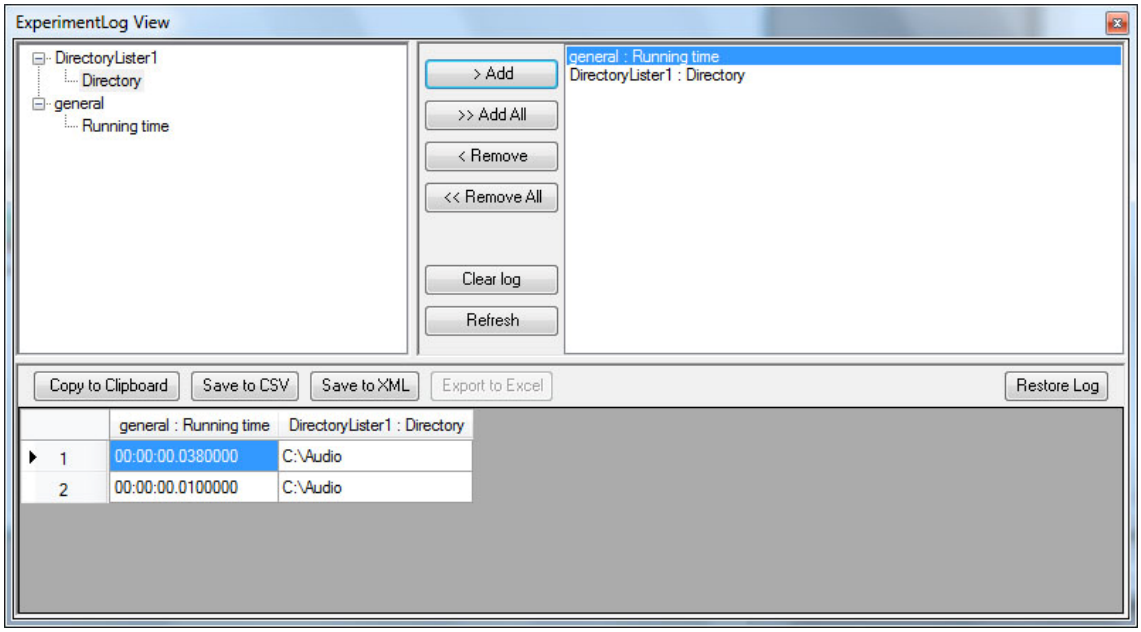
8.3 Funkce Experiment Log

Nyní si ukážeme jak funguje a k čemu slouží funkce Experiment Log. Do Experiment Logu může kterýkoliv modul uložit jakoukoliv položku nebo více položek.

Zmáčkneme tlačítko ‚Show Experiment Log' na liště MapEditoru (viz obrázek 29) a následně se zobrazí okno Experiment Logu (viz obrázek 31).

Na levé straně obrazovky je seznam všech modulů, které ukládají alespoň jednu položku a u každého modulu je seznam položek, které do Experiment Logu ukládá. Kliknutím na ‚Add' přidáme položku na pravou stranu obrazovky, čímž říkáte, že chcete tuto položku sledovat. Následně se tato položka objeví jako sloupec v dolní části obrazovky, kde můžete sledovat její hodnotu v jednotlivých spuštěních mapy. Každému spuštění mapy odpovídá jedna řádka.

Jedna položka je vždy navíc, je to položka Running time, která je poskytována MVE a ta ukazuje celkový čas jednoho spuštění mapy.



Obrázek 31: Experiment Log

Celá tabulka, která se ukazuje v dolní části obrazovky, se dá exportovat, a to stiskem tlačítka ‚Copy to Clipboard', čímž se tabulka zkopíruje do schránky, nebo stiskem tlačítek ‚Save to CSV' nebo ‚Save to XML', které tabulku uloží do souboru.

Obsah Experiment Logu se automaticky ukládá do souborů s příponou .autosave a může být z těchto souborů obnoven. To se udělá tlačítkem ‚Restore Log'.

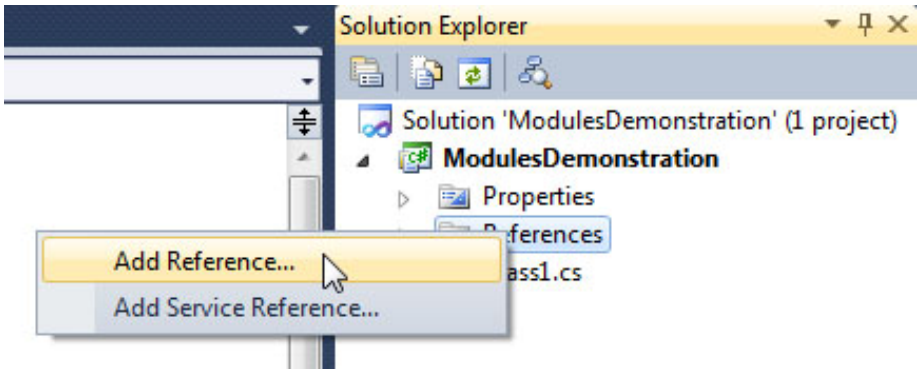
9 Kapitola 5 – Vlastní základní modul

V této kapitole se budeme zabývat vytvářením vlastního modulu se základními funkcemi. Většinu této kapitoly zaplňuje psaní zdrojových kódů, proto se v celé této kapitole odkazuji na přílohy do sekce zdrojové kódy, kde naleznete celé kódy jednotlivých modulů a tříd. Zde bude pouze výpis nejdůležitějších metod a konstrukcí.

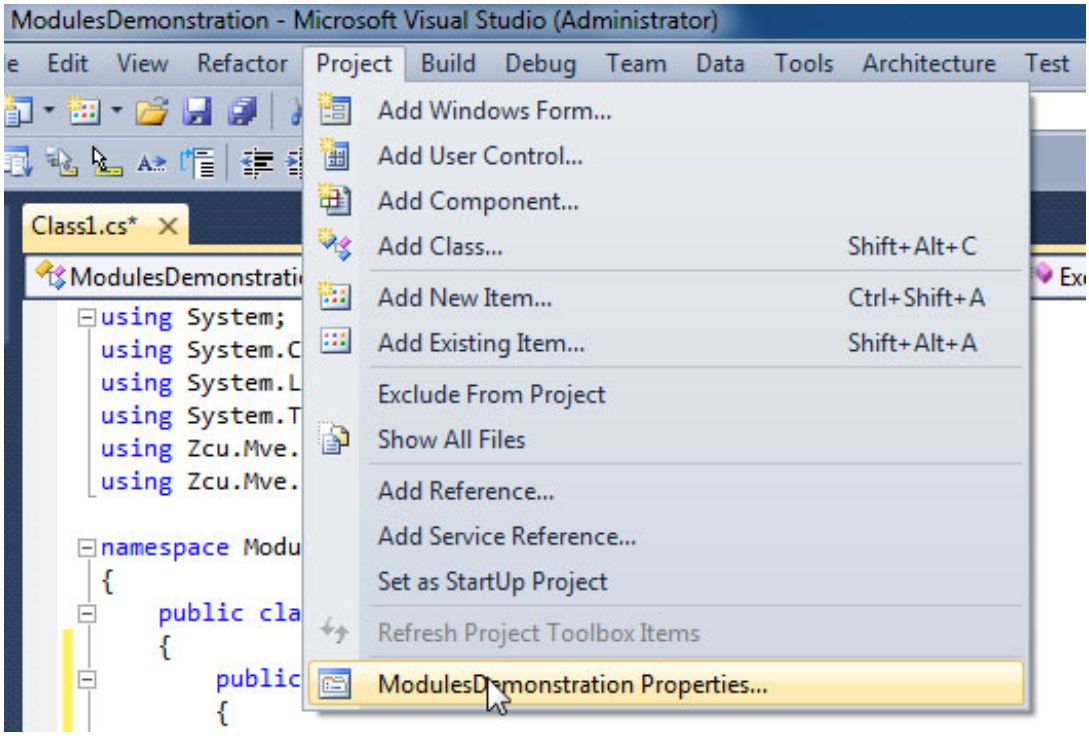
9.1 Příprava projektu

Nejefektivnější metodou na vytvoření vlastního modulu je programování v programu Visual Studio, přičemž nezáleží na verzi, kterou máte.

Vytvoříme nový projekt typu Class Library. Výsledkem tohoto typu projektu je .dll knihovna.



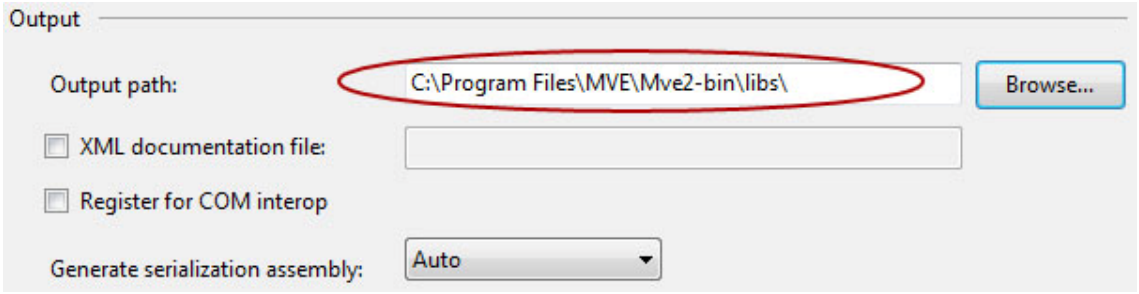
Obrázek 32: nabídka Přidat odkaz



Obrázek 33: volba Project - Properties

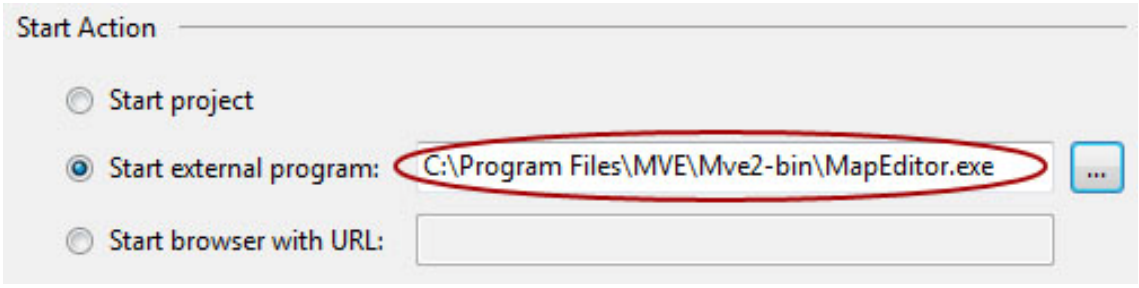
Potřebujeme oddědit modul, který budeme programovat, od třídy Module, takže budeme potřebovat knihovnu MveCore. Dále budeme potřebovat pracovat se základními numerickými datovými typy, takže budeme potřebovat knihovnu Numerics. V modulu Randomizer budeme potřebovat pracovat i s datovými typy spojenými s vizualizací, protože budeme potřebovat i knihovnu Visualization. Proto přidáme v projektu odkaz (reference) na MveCore.dll, Numerics.dll a Visualization.dll, které můžeme najít ve složce libs v MVE. Viz obrázek 32.

Zvolíme ‚Project' – ‚Properties' (viz obrázek 33), poté kartu ‚Build'. Změníme ‚Output path' tak, aby směřovala do MVE do složky libs (viz obrázek 34).



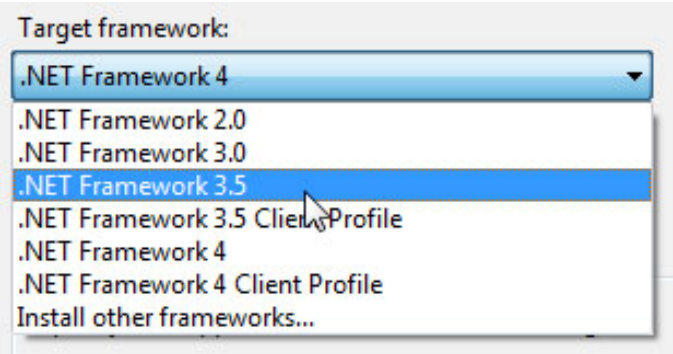
Obrázek 34: nastavení v kartě Build

Dále zvolíme kartu ‚Debug', kde zvolíme ‚Start external program' a najdeme MapEditor.exe (viz obrázek 35).



Obrázek 35: nastavení v kartě Debug

A nakonec zvolíme kartu ‚Application', kde změníme vlastnost ‚Target framework' na ‚.NET Framework 3.5' (viz obrázek 36).



Obrázek 36: nastavení v kartě Application

Dále je potřeba zbavit se okna Loader Lock Exception, které vyskakuje při debugování modulu. Proto jdeme do menu ‚Debug', zvolíme ‚Exceptions...', poté rozklikneme volbu ‚Managed Debugging Assistants', v ní najdeme ‚LoaderLock' a odškrtneme políčko ve sloupci ‚Thrown'.

9.2 Programování modulu MyModule

Při programování tohoto Modulu se naučíme základní požadavky na vlastní modul a základní práci s porty.

Aby mohl modul vůbec fungovat, musí splňovat tři věci:

- Musí být oděděn od třídy Module:

```
public class MyModule : Module
```

- Musí mít konstruktor (v něm obvykle přidáváme všechny porty, ale může být i prázdný):

```
public MyModule()
```

- Musí mít přetíženou metodu ‚Execute', ve které je vykonávána činnost modulu. Když MVE spouští modul, spouští právě tuto metodu:

```
public override void Execute()
```

Takový modul už může být přeložen, vytvořen a vložen do mapy a už může vykonávat nějakou základní funkčnost.

To nejdůležitější u modulu je komunikace s ostatními moduly, a ta se děje skrz porty. Ty přidáváme následujícími metodami, obvykle v konstruktoru.

- Metoda pro přidání vstupního portu, v závorce je nejprve název portu a poté datový typ portu:

```
AddInPort("Number1", typeof(Scalar));
```

- Metoda pro přidání výstupního portu, v závorce je nejprve název portu a poté datový typ portu:

```
AddOutPort("Result", typeof(Scalar));
```

- Nakonec ze vstupního portu získáme data, která uložíme do proměnné. Identifikátor ‚(Scalar)' je explicitní přetypování a v závorce metody je název portu, ze kterého data získáváme:

```
Scalar n1 = (Scalar)GetInput("Number1");
```

9.3 Programování modulu Randomizer

Při programování tohoto modulu se naučíme pracovat se složitějšími datovými typy z knihovny Visualization, budeme pracovat s grafickými daty.

- Nejdůležitějším datovým typem je tento:

```
TriangleMesh
```

- To je datový typ pro trojúhelníkovou síť. Ta sestává z geometrie (údaje o vrcholech), topologie (údaje o trojúhelnících) a atributů.

- Pro práci s geometrií trojúhelníkové sítě potřebujeme datový typ MVE odpovídající poli, bude to pole vrcholů (‚input.points.count' je počet vrcholů). Právě v něm je geometrie trojúhelníkové sítě uložena:

```
UniformDataArray newPoints = new UniformDataArray(typeof(Point3D),
input.Points.Count);
```

- Pro práci s topologií trojúhelníkové sítě potřebujeme metodu, která vrací ‚UniformDataArray' trojúhelníků, jinak řečeno vrací pole, které obsahuje celou topologii. Každý trojúhelník je buňka typu ‚Triangle'. Na konci musí být ‚[0]',

protože polí buněk typu trojúhelník by v trojúhelníkové síti mohlo teoreticky být víc (v našem případě není):

```
GetCells(typeof(Triangle))[0]
```

Trojúhelníková síť může mít libovolné množství atributů. Atribut se dá přidat dvojím způsobem. Buď přiřadíme nějakou hodnotu každému vrcholu, nebo přiřadíme nějakou hodnotu každé buňce z jednoho pole buněk (v našem případě to znamená každému trojúhelníku).

- Následující metodu potřebujeme k přidání atributů k vrcholům trojúhelníkové sítě ‚result', identifikátor atributů bude ‚cosines', přidáváme pole hodnot ‚attr':

```
result.AddPointAttrs("cosines", attr);
```

- Následující metodu potřebujeme k přidání atributů k buňkám trojúhelníkové sítě ‚result', ‚triangles' je identifikátor buněk, ke kterým atribut přidáváme (tím identifikujeme jedno pole buněk), identifikátor atributů bude ‚index', přidáváme pole hodnot ‚triangleIndex':

```
result.AddCellAttrs("triangles", "index", triangleIndex);
```

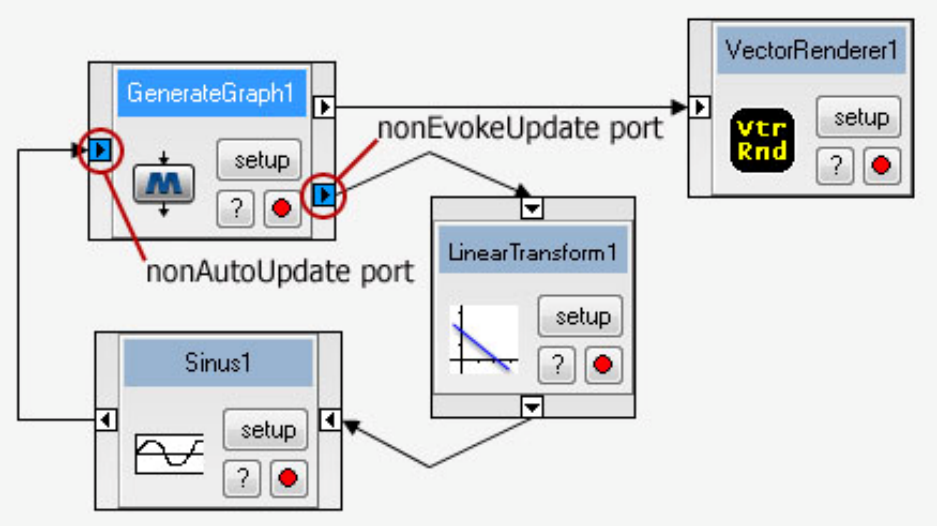
10 Kapitola 6 – Vlastní speciální modul

Obsahem této kapitoly je přidání tří speciálních vlastností modulu, který jsme vytvořili v kapitole 5. Jsou to vlastnosti: přidání speciálních portů, možnost nepočítání známého výsledku a vlastní okno nastavení.

Většinu této kapitoly zaplňuje psaní zdrojových kódů, proto se v celé této kapitole odkazuji na přílohy do sekce zdrojové kódy, kde naleznete celé kódy jednotlivých modulů a tříd. Zde bude pouze výpis nejdůležitějších metod a konstrukcí.

10.1 Přidání speciálních portů

Existuje speciální vstupní port, kterému se o přísun dat nestará MVE automaticky, ale říká si o ně sám. Tento port se nazývá nonAutoUpdate port. Také existuje speciální výstupní port, který při požadavku, aby poskytl data, nespustí modul, který ho obsahuje (protože tento modul už běží), ale pouze poskytne data. Tento port se nazývá nonEvokeUpdate port. Kombinace těchto dvou modulů umožňuje, aby byl v mapě cyklus. Tento cyklus je řízen modulem, který obsahuje tyto dva porty. Viz obrázek 37.



Obrázek 37: speciální porty

- Přidání nonAutoUpdate portu:

```
AddInPort("Number1", typeof(Scalar), true, false);
```

- Přidání nonEvokeUpdate portu:

```
AddOutPort("Result", typeof(Scalar), false);
```

- Metoda, kterou se vyvolá potřeba dat na vstupním portu, a tím se spustí jeden běh cyklu:

```
UpdateInput("Number1");
```

10.2 Možnost nepočítání známého výsledku

Pokud víme, že vstoupila do modulu stejná data jako v předchozím běhu a víme, že se nezměnilo nastavení modulu, pak můžeme (pokud jsme si uložili výsledek předchozího běhu) vystavit na výstup výsledek předchozího běhu, aniž bychom ho znovu počítali.

- Metoda, která říká, jestli jsou data na všech vstupních portech stejná jako v předchozím běhu:

```
IsAllDataSame()
```

- Metoda, která vystaví data na výstup. Vytváříme nový datový typ ‚Scalar', který obaluje staré číslo ‚result', které je známé již z předchozího běhu, ‚true' jako třetí parametr metody říká ostatním modulům v mapě, že se data nezměnila, a tak mohou například také přeskočit jejich výpočet a vystavit již známá data

```
SetOutput("Result", new Scalar(result), true);
```

10.3 Vlastní okno nastavení

- Pokud vám nestačí okno nastavení, které vytváří MVE automaticky, můžete vytvořit své vlastní okno. Jediné, čím jste omezeni, je, že toto okno musí být potomkem třídy ‚ModuleSetup':

```
public partial class MySetupDialog : ModuleSetup
```

- Ve třídě, která představuje váš modul, vytvoříte instanci vlastního okna nastavení (v mém příkladě instanci třídy ‚MySetupDialog')

- Přetížíte metodu ‚InvokeSetup', aby vracela instanci vašeho okna nastavení:

```
public override ModuleSetup InvokeSetup()
{
    return mySetup;
}
```

11 Závěr

Pro vytváření video tutoriálu jsem vybral program Adobe Captivate 5 a kameru Canon Legria HF S21, kterou jsem nahrával zvuk. Poté jsem zvuk vložil do projektu programu Captivate a výsledek exportoval do videa SWF, které se spouští přes soubor s příponou htm.

V angličtině jsem nahrál šest kapitol video tutoriálu pro MVE. Kapitola 5 je kvůli její značné délce rozdělena do dvou videí, proto je výsledkem 7 videí o délkách mezi 5 a 15 minutami. Dohromady tedy mají 82 minut a zabírají 400MB.

S výslednou prací jsem spokojen. Videa jsou srozumitelná a obsahově souhlasí s celým rozsahem základního video tutoriálu pro Modulární vizualizační prostředí. Videa divákovi umožní používat všechny funkce, které bude jako středně pokročilý uživatel chtít používat.

Program Adobe Captivate jako nástroj pro nahrávání videa hodnotím jako vhodný a dostačující, avšak má své nevýhody a někdy padá. Pokud bych dělal další video tutoriály, rozhodně bych si vybral jiný program, a to Camtasia Studio.

Přehled zkratek

KIV	Katedra informatiky a výpočetní techniky
MVE-2	*Modular Visualization Environment – 2*, Modulární vizualizační prostředí
MVE	*Modular Visualization Environment – 2*, Modulární vizualizační prostředí
SWF	*ShockWaveFlash*, formát video souboru
ZČU	Západočeská univerzita
xml	Extensible Markup Language – standardní formát pro výměnu informací

Literatura

Elektronické zdroje

[1] *MVE-2 Handbook in Czech (7.2.2005)*
URL: http://herakles.zcu.cz/research/projects/11/down/doc-mve-cz.pdf
[citováno 3. března 2011]

[2] Vendula Ferschmannová – *Multimediální a hypermediální systémy: Výuková videa k ekonomickému systému FlexiBee*
Semestrální práce předmětu MHS, Západočeská univerzita v Plzni. 2010

Přílohy

Zdrojové kódy

Class1.cs – Kapitola 5

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Zcu.Mve.Core;
using Zcu.Mve.Numerics;

namespace ModulesDemonstration
{
    public class MyModule : Module
    {
        double multiplier;

        public double Multiplier
        {
            get { return multiplier; }
            set { multiplier = value; }
        }

        public MyModule()
        {
            AddInPort("Number1", typeof(Scalar));
            AddInPort("Number2", typeof(Scalar));
            AddInPort("Number3", typeof(Scalar));
            AddOutPort("Result", typeof(Scalar));
        }

        public override void Execute()
        {
            Scalar n1 = (Scalar)GetInput("Number1");
            Scalar n2 = (Scalar)GetInput("Number2");
            Scalar n3 = (Scalar)GetInput("Number3");

            double result = (n1.Value + n2.Value + n3.Value) *
multiplier;

            Scalar resultSc = new Scalar(result);
```

```
            SetOutput("Result", resultSc);
        }
    }
}
```

Randomizer.cs – Kapitola 5

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Zcu.Mve.Core;
using Zcu.Mve.Numerics;
using Zcu.Mve.Visualization;

namespace ModulesDemonstration
{
    class Randomizer : Module
    {
        public Randomizer()
        {
            AddInPort("Input", typeof(TriangleMesh));
            AddOutPort("Output", typeof(TriangleMesh));
        }

        public override void Execute()
        {
            TriangleMesh input = (TriangleMesh)GetInput("Input");
            TriangleMesh result = (TriangleMesh)input.ShallowCopy();

            Random r = new Random(0);

            UniformDataArray newPoints = new
UniformDataArray(typeof(Point3D), input.Points.Count);

            for (int i = 0; i < input.Points.Count; i++)
            {
                Point3D origPoint = (Point3D)input.Points[i];

                origPoint.X += r.NextDouble();
                origPoint.Y += r.NextDouble();
                origPoint.Z += r.NextDouble();

                newPoints[i] = origPoint;
```

```
            }

            result.Points = newPoints;

            UniformDataArray triangles =
input.GetCells(typeof(Triangle))[0];

            Console.WriteLine("There are " + triangles.Count + "
triangles.");

            foreach (string s in input.GetCellIds())
            {
                WriteConsole(s);
            }

            Triangle t = (Triangle)triangles[0];
            WriteConsole(t.V1.ToString() + " " + t.V2.ToString() + " "
+ t.V3.ToString());



            UniformDataArray attr = new
UniformDataArray(typeof(Scalar), input.Points.Count);
            for (int i = 0; i < input.Points.Count; i++)
            {
                attr[i] = new
Scalar(Math.Cos(((Point3D)input.Points[i]).X));
            }
            result.AddPointAttrs("cosines", attr);

            UniformDataArray triangleIndex = new
UniformDataArray(typeof(Integer), triangles.Count);
            for (int i = 0; i < triangles.Count; i++)
            {
                triangleIndex[i] = new Integer(i);
            }
            result.AddCellAttrs("triangles", "index", triangleIndex);

            WriteLog("Number of points", input.Points.Count);

            SetOutput("Output", result);
        }
    }
}
```

MySetupDialog.cs – Kapitola 6

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Zcu.Mve.Core;

namespace ModulesDemonstration
{
    public partial class MySetupDialog : ModuleSetup
    {
        bool isChecked;

        public bool IsChecked
        {
            get { return isChecked; }
        }

        public MySetupDialog()
        {
            InitializeComponent();
        }

        private void checkBox1_CheckedChanged(object sender, EventArgs
e)
        {
            this.isChecked = checkBox1.Checked;
        }
    }
}
```

Class1.cs – Kapitola 6

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Zcu.Mve.Core;
using Zcu.Mve.Numerics;

namespace ModulesDemonstration
```

```
{
    public class MyModule : Module
    {
        double multiplier;

        public double Multiplier
        {
            get { return multiplier; }
            set { multiplier = value;
            paramsChanged = true;
            }
        }

        bool paramsChanged = false;

        double result = double.NaN;

        MySetupDialog mySetup;

        public MyModule()
        {
            AddInPort("Number1", typeof(Scalar), true, false);
            AddInPort("Number2", typeof(Scalar));
            AddInPort("Number3", typeof(Scalar));
            AddOutPort("Result", typeof(Scalar), false);

            mySetup = new MySetupDialog();
        }

        public override void Execute()
        {
            if     (paramsChanged     ||     !IsAllDataSame()     ||
double.IsNaN(result))
            {
                UpdateInput("Number1");

                Scalar n1 = (Scalar)GetInput("Number1");
                Scalar n2 = (Scalar)GetInput("Number2");
                Scalar n3 = (Scalar)GetInput("Number3");

                result  =  (n1.Value  +  n2.Value  +  n3.Value)  *
multiplier;

                Scalar resultSc = new Scalar(result);
```

```
                SetOutput("Result", resultSc);

                paramsChanged = false;
            }
            else
            {
                SetOutput("Result", new Scalar(result), true);
            }
        }

        public override ModuleSetup InvokeSetup()
        {
            return mySetup;
        }
    }
}
```