JEDNODUCHÁ SIMULAČNÍ HRA KROK ZA KROKEM ANEB OD PŘEDSTAVY K HOTOVÉMU

Petr Stodulka, Pavol Privitzer, Jiří Kofránek

Anotace

Článek popisuje tvorbu interaktivních simulačních her. Ukazujeme postup práce od prvotní myšlenky (scénáře) přes tvorbu fyziologického modelu, návrh uživatelského rozhraní, řídící logiky, propojení do podoby funkční aplikace až k publikaci aplikace na internetu.

Sestavíme jednoduchý simulátor resuscitace pacienta pomocí masáže hrudníku a umělého dýchání (první pomoc). Simulátor je založený na jednoduchém fyziologickém modelu krevního oběhu. Uživateli se zobrazuje průběh hry pomocí animací a grafů příslušných fyziologických proměnných.

Sledujeme nejnovější trendy na poli modelování a vývoje aplikací. V oblasti modelování se přikláníme k modernímu modelovacímu jazyku Modelica, při tvorbě aplikace využíváme možností platformy .NET Framework 3.5 (C# 3.0, WPF, XBAP). Pro návrh používáme vzor Model-View-ViewModel. K řízení logiky simulace potom využíváme možností stavových automatů.

Tato práce navazuje na dlouhodobý výzkum a vývoj v oblasti fyziologického modelování a tvorby výukových simulátorů v naší laboratoři.

Klíčová slova

Výukové simulátory, simulační hry, fyziologické modelování, Modelica, .NET

1. Úvod

Nejprve připomeňme, co míníme simulačními hrami a jaké jsou problémy při jejich tvorbě. O tomto tématu se může čtenář více dočíst např. v [1]. Simulační hra, nebo výukový multimediální simulátor, je aplikace postavená na fyziologickém modelu, která má prezentovat procesy odehrávající se v modelu a umožnit jejich ovlivnění. To vše pokud možno pomocí sofistikovaného, po grafické stránce bohatého, uživatelského rozhraní. Modely, multimedia a programování aplikací jsou

tři víceméně nesourodé okruhy. Pro každý existují speciální nástroje, ale zajistit jejich vzájemnou integraci je náročné.

Náš tradiční přístup bylo vytvořit animace v Adobe Flash, model v Matlab Simulink a jádro simulátoru na platformě .NET v jazyce C#. Flashové animace lze v .NET použít (s omezenými možnostmi) jako ActiveX komponentu. Model jsme konvertovali do použitelné podoby pomocí vlastního převodního nástroje podrobně popsaného v [2].
V tomto článku představujeme technologickou inovaci spočívající v nahrazení Flashových animací technologií WPF a proprietárního Simulinku platformou OpenModelica.

WPF (Windows Presentation Foundation) je novinkou ve verzi platformy .NET Framework 3.0, která jednak při tvorbě uživatelského rozhraní programů nahrazuje Windows Forms a jednak umožňuje vytváření animací plně integrovaných s vlastním programem. Odstraněním technologické bariéry mezi .NET a Flash nám WPF umožňuje těsnou integraci a přesnější řízení animací a vizualizací. Důležitým rysem je také možnost spouštět aplikace přímo v okně webového prohlížeče.

Jazyk Modelica je moderním modelovacím jazykem, který podporuje objektově-orientovanou hierarchii a modularitu a oproti dříve používanému Simulinku umožňuje deklarativní (tzv. akauzální) zápis modelů. Jeho otevřená implementace OpenModelica nám umožňuje detailně prozkoumat použité algoritmy a také se zapojit do vývoje tohoto nástroje.

Tento článek má ukázat uvedené technologie a naše postupy v praxi. Provedeme čtenáře postupem tvorby simulátoru od scénáře přes návrh uživatelského rozhraní, fyziologického modelu a řídícího stavového automatu až po kompletaci samotné aplikace a její zveřejnění na internetu. Zdrojové soubory simulátoru včetně modelu v Modelice je možné získat na stránce http://patf-biokyb.lf1.cuni.cz/wiki/firstaidgame.

2. Architektura

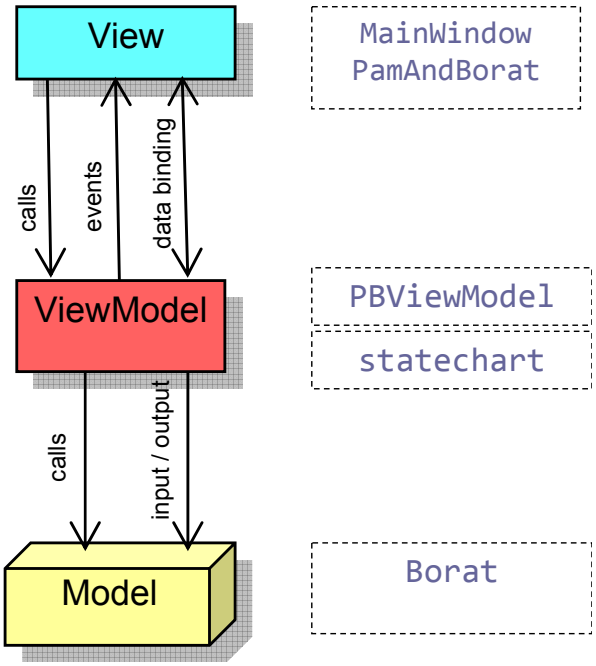
Vycházíme z varianty návrhového vzoru Model-View-Controller, která se ve světě WPF aplikací používá častěji a nazývá se Model-View-ViewModel. Podrobný popis této architektury lze najít na webu [3-5].

Model je obdobou datové vrstvy. V našem případě představuje fyziologický model konvertovaný z jazyka Modelica. Třída modelu (Borat) nabízí vstupy a výstupy modelu a metody pro jeho řízení (Reset(), Step(), …).

ViewModel představuje vrstvu, která řídí chod aplikace i modelu a zprostředkovává data vizualizační vrstvě ve vhodné podobě. Na návrh a řízení chodu aplikace používáme stavové automaty.

View je prezentační vrstva, která komunikuje s uživatelem. V optimálním případě jeimplementovaná jako čistý XAML kód a je přímo napojena na ViewModel pomocí událostí, commandů a databindingu.

Důsledné rozvrstvení umožňuje souběžnou práci programátora a grafika, který vytváří vizuální stránku aplikace včetně animací.



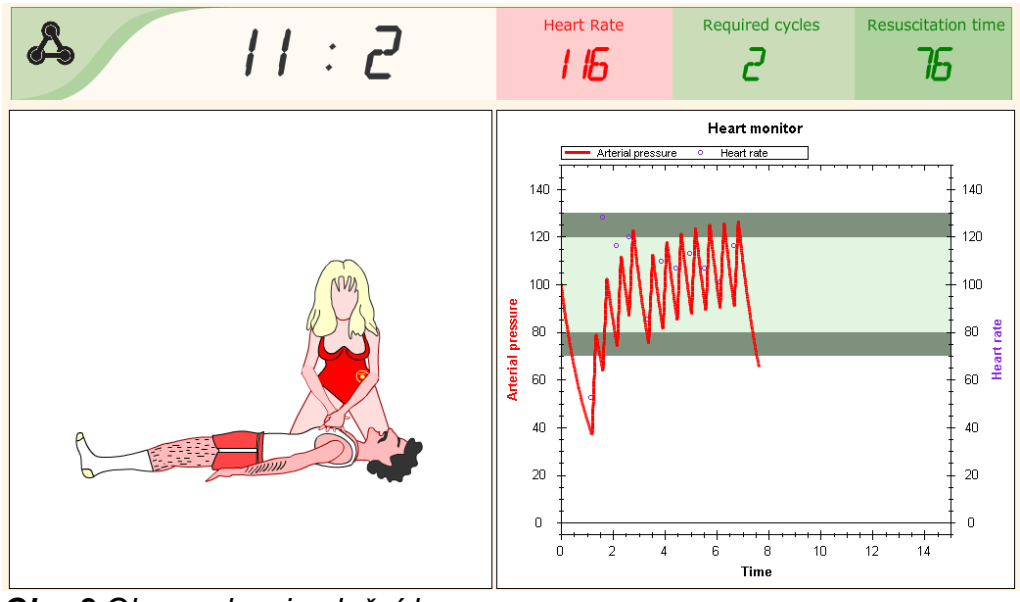
Obr. 1 *Využití architektury Model-View-Controller v simulační hře*

3. Postup práce

3.1. Scénář

Cílem hry bude prezentovat nový doporučený poměr třiceti stlačení hrudníku ku dvěma vdechům z úst do úst při resuscitaci pacienta. Tento poměr bude prezentován jednoduchou hrou, kdy klikáním myší na hrudník nebo na hlavu pacienta dává záchranář masáž hrudníku nebo umělé dýchání. Správný postup pro oživení pacienta:

1. kontrola dechu poslechem
2. třicetkrát masáž hrudníku
3. zaklonění hlavy
4. dvakrát stisk nosu a vdech z úst do úst
5. opakování bodů 2. až 4. dokud je potřeba



Obr. 2 *Obrazovka simulační hry*

Simulátor kontroluje správný poměr akcí a fyziologický model počítá arteriální tlak pacienta. Podle frekvence stisku hrudníku (obdoba frekvence srdečního tepu) a podle počítaného tlaku je pacient po určité době prohlášen za úspěšně resuscitovaného nebo za mrtvého.

V případě, že tlak pacienta je po určitou dobu mimo vyhrazený interval (tepová frekvence je příliš vysoká nebo příliš nízká), je zobrazeno varování a uživateli je připsán trestný bod. Naopak za dokončený resuscitační cyklus je mu trestný bod upsán. Po odepsání všech trestných bodů je hra úspěšně dokončena. Dosažením 10 trestných bodů pacient umírá.

Kromě animace záchranáře a pacienta je uživateli zobrazen graf arteriálního krevního tlaku pacienta a číselné údaje o průběhu hry – celkový čas resuscitace, tepová frekvence pacienta apod.

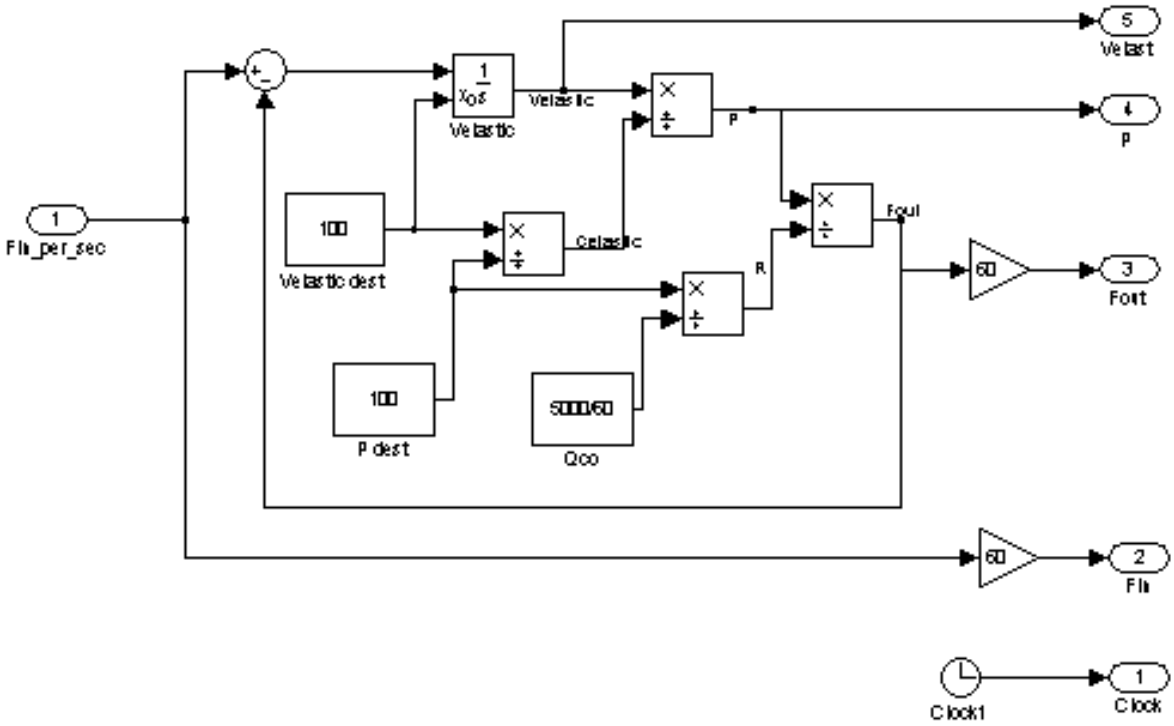
3.2. Model

Pro účely tohoto příkladu stačí jednoduchý model, který nahrazuje krevní řečiště elastickou trubicí a krevní tlak počítá pomocí elasticity, rezistence a objemu tohoto kontejneru. Vstupem modelu je příznak tlaku na hruď pacienta **c** (0 nebo 1), výstupem je tlak **p** (v mmHg) tekutiny v popsané elastické trubici. **p** zastupuje v tomto případě arteriální krevní tlak. Druhou proměnnou vystupující z modelu je **q**, tok tekutiny v ml/s. Tok tekutiny však v simulátoru nepoužíváme.

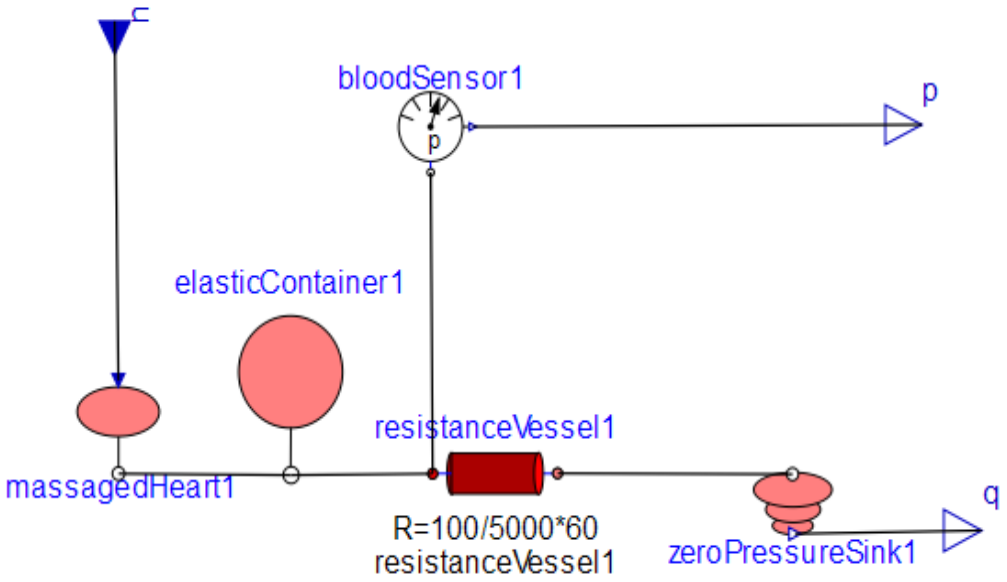
Pro projekt OpenModelica jsme vytvořili rozšíření pro konverzi modelicového modelu do jazyka C#, který je dostupný na adrese: http://patf-biokyb.lf1.cuni.cz/wiki/modelica.net. Popsaný model pomocí tohoto nástroje převedeme do podoby třídy napsané v jazyce C# (třída Borat). Tato třída obsahuje soustavu diferenciálních rovnic reprezentujících model převedených do explicitního tvaru vhodného pro řešení pomoci ODE solveru. Borat dědí z obecné třídy ModelicaModel, která poskytuje základní funkcionalitu modelu jako je řízení běhu, vstupů, výstupů apod.

3.3. Animace

WPF je prezentační technologie, která kromě standardních ovládacích prvků uživatelského rozhraní, jako jsou tlačítka apod., umí nativně definovat a zobrazovat animace. Ty jsou zapsány v podobě XAML zdrojových souborů, které lze upravovat ručně nebo pomocí speciálních designerských nástrojů (např. MS Expression Blend). Animované sekvence jsou vyjádřeny pomocí časové osy (třída Storyboard). Ovládáním časové osy potom řídíme zobrazované animace.



Obr. 3 *Simulační model v Simulinku*



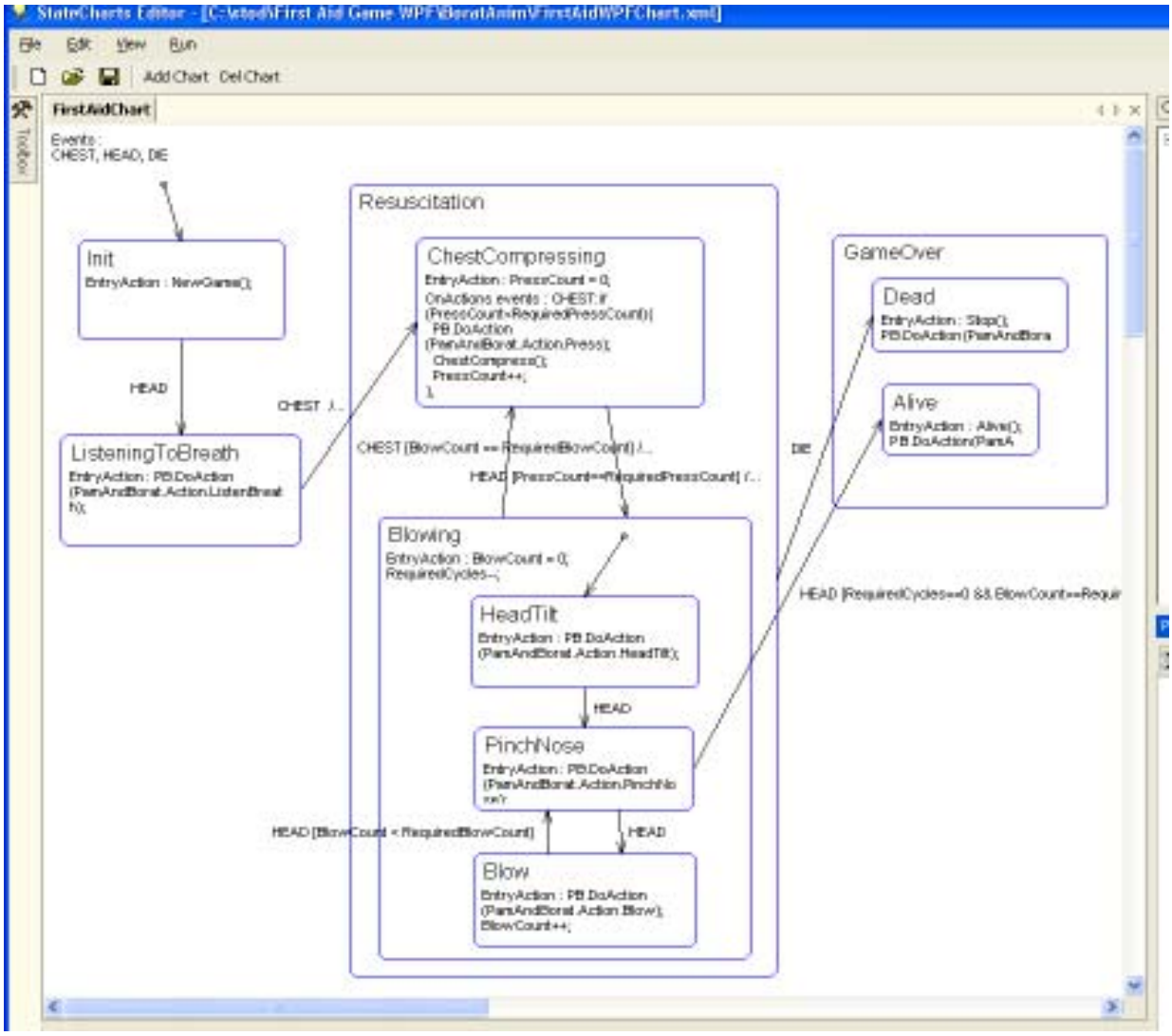
Obr. 4 *Simulační model v Modelice*

Animace v jazyce XAML je možné získat také konverzí Flashových animací pomocí nástroje The Converted [6]. Tímto postupem jsme získali soubor s animací PamAndBorat.xaml. Identifikovali jsme úseky časové osy vyjadřující jednotlivé fáze animace a vložili je podpůrného kódu (viz konstruktor PamAndBorat v souboru PamAndBorat.xaml.cs). Dosáhli jsme tak oddělení implementace animace od vyvolávání jednotlivých sekvencí.

3.4. Řízení aplikace

Logikou simulátoru myslíme hlídání počtu cyklů, správné pořadí prováděných resuscitačních akcí a následné řízení zobrazovaných animací, nebo reakce na vstupy od uživatele. Tato pravidla lze lehce vyjádřit v grafické podobě pomocí *stavových automatů* detailně popisovaných v článku [7]. Použitím nástroje Statecharts můžeme navrhovat řídící automaty a generovat kód, který se integruje do *ViewModel* vrstvy aplikace a zajišťuje její řízení. Domovská stránka projektu Statecharts je http://patf-biokyb.lf1.cuni.cz/wiki/statecharts.

Definice řídícího automatu našeho simulátoru je v souboru FirstAidWPFChart.xml. Výstupem po jeho kompilaci je zdrojový soubor FirstAidWPFChart.xml.cs a knihovna FirstAidWPFChart.xml.dll. Tyto dva soubory musí být spolu s knihovnou Statecharts/Runtime.dll připojeny k projektu vytvářené aplikace.



Obr. 5 *Hierarchický stavový automat využívaný pro řízení aplikace*

3.5. Zveřejnění hotové aplikace

Pro publikaci aplikace využijeme schopnost .NET aplikací běžet přímo v okně webového prohlížeče. Aplikace se transparentně nahraje do paměti počítače a spustí se v bezpečném prostředí, tzv. security sandboxu. Spuštění aplikace je z pohledu uživatele rovno návštěvě běžné webové stránky. Před samotným spuštěním aplikace je provedena kontrola přítomnosti frameworku .NET 3.5, případně je uživateli nabídnuta jeho instalace.

Technologie běhu aplikace v okně prohlížeče se nazývá XBAP – XAML Browser Application. Pro vytvoření XBAP aplikace je potřeba ve Visual Studiu při výběru typu projektu zvolit šablonu WPF Browser Application.

4. Závěr

Ve stručnosti jsme ukázali architekturu simulátorů vyvíjených v naší laboratoři. Na konkrétním jednoduchém příkladu simulační hry pro nácvik základů moderní laické první pomoci jsme prošli jednotlivé kroky potřebné k realizaci výsledné aplikace. Výraznou inovací je použití moderních softwarových nástrojů jak v oblasti modelování, tj. jazyka Modelica, tak i na aplikační úrovni, kde se technologie WPF ukazuje jako robustní platforma pro silně graficky orientované aplikace.

Nemalou výhodu použitého postupu vidíme v bezešvém skloubení různých technologií pro návrh simulačních matematických modelů, aplikační logiky, grafické stránky a vlastní implementace výukového simulátoru. Na jednom projektu takto mohou současně spolupracovat modeláři, grafici i programátoři, a to každý ve své vlastní doméně bez ohrožení udržitelnosti integrity zdrojových souborů. To je, jak nám praxe ukazuje, zásadní báze pro realizaci mnohem komplexnějších simulátorů, než je uvedený demonstrační příklad simulační hry první pomoci.

Z uživatelského hlediska zatím zůstává omezením dostupnost .NET Frameworku použité verze jenom na platformě Windows XP a vyšší. Mnohé si v tomto směru slibujeme od nové nastupující platformy MS Silverlight, zvláště od očekávané verze 2.0. Je to ve své podstatě plug-in pro internetové prohlížeče, který je založen na technologii WPF, avšak bude dostupný i pro jiné operační systémy jako OS Mac či Linux.

Kompletní zdrojové kódy simulační hry první pomoci je možno stáhnout ze stránek http://patf-biokyb.lf1.cuni.cz/wiki/firstaidgame.

5. Literatura

[1] Stodulka, P., Privitzer, P., Kofránek, J., Tribula, M., Vacek, O. (2007): *Development of web accessible medical educational simulators*. In. Proceedings of the 6th EUROSIM, Vol. 2. Full Papers (CD), MO-3-P4-2, str. 1-6, ISBN 978-3-901608-32-2, University of Ljubljana, 2007.

[2] Stodulka, P., Privitzer, P., Kofránek, J., Mašek, J. (2006): *Nové postupy v tvorbě simulátorů – inteligentní propojení Matlab Simulinku s platfor*mou .NET a tvorba stavových automatů řídících výslednou aplikaci.In MEDSOFT 2006, str. 171-176, Agentura Action-M, ISBN 80-86742-12-1, Praha, 2006

[3] http://blogs.msdn.com/johngossman/archive/2005/10/08/478683.aspx

[4] http://www.orbifold.net/default/?p=550

[5] http://blog.quantumbitdesigns.com/2008/01/20/wpf-application-design-and-architecture/

[6] *The Converted – Swf to Xaml Converter* – http://theconverted.ca/

[7] Kofránek, Stodulka (2004): *Hierarchické komunikující stavové automaty.* Objekty 2004, str. 116-133, VŠB Ostrava, ISBN 80-248-0672-X, Ostrava 2004.

Poděkování

Práce na vývoji lékařských simulátorů je podporována projektem Národního programu výzkumu č. 2C0603, rozvojovým projektem MŠMT C34/2008 a společností BAJT servis s.r.o..

Petr Stodulka
Laboratoř biokybernetiky a počítačové potobry výuky, Ústav patologické fyziologie 1. LF UK, Praha
U nemocnice 5
128 53 Praha 2
tel: 224965912
fax:224912834
e-mail: petr.stodulka@gmail.com
http://www.physiome.cz