TVORBA MULTIMEDIÁLNÍCH VÝUKOVÝCH PROGRAMŮ

Jiří Kofránek, Michal Andrlík, Tomáš Kripner, Zdeněk Wünsch

Anotace

Vývoj efektivních výukových programů, kombinující multimédia se simulačními hrami je náročnou a komplikovanou prací, vyžadující týmovou spolupráci řady profesí – zkušených pedagogů vytvářejících základní scénář, tvůrců simulačních modelů, lékařů, výtvarníků a programátorů. Tuto interdisciplinární kolektivní tvorbu zefektivňuje využívání vhodných vývojových nástrojů, které umožňují komponentovou tvorbu a propojení simulačních programů, a interaktivních multimédií podle daného scénáře do kompaktního celku. Pro kostru aplikace autoři využívají Macromedia RoboHelp, resp. Macromedia Authorware, pro tvorbu simulačních modelů využívají nástroje firmy Mathworks Inc. (Matlab a Simulink). Pro tvorbu interaktivních animací využívají nástroje firmy Macromedia (Flash). Pro tvorbu uživatelského rozhraní výukových programů a simulátorů autoři využívají vývojový nástroj Control Web (z firmy Moravské přístroje) a vývojové prostředí Visual Studio. NET.

Klíčová slova

MULTIMEDIÁLNÍ VÝUKOVÉ PROGRAMY, INTERDISCIPLINÁRNÍ SPOLUPRÁCE, SOFTWAROVÉ VÝVOJOVÉ NÁSTROJE, SIMULACE.

1. Úvod

Navzdory tomu, že se využití počítačů ve výuce stalo tématem řady konferencí, odborných i popularizačních článků, přesto, že hardwarové možnosti i softwarové nástroje dnes umožňují vytvářet náročná interaktivní multimedia, k výraznému rozšíření multimediálních výukových programů ve výuce medicíny zatím nedošlo. Příčin je několik. Za prvé, ukazuje se, že tvorba výukových programů je podstatně náročnější na čas, lidské i materiální zdroje, než je obvykle plánováno (jestliže například v rámci grantů Fondu rozvoje vysokých škol je na vytvoření jednoho multimediálního výukového programu plánováno maximálně 120 tisíc Kč, pak výsledkem nemůže být nic jiného, než po technické o výtvarné stránce nedotažený produkt). Za druhé – tvorba kvalitních výukových programů vyžaduje týmovou multidisciplinární spolupráci zkušených pedagogů, výtvarníků i programátorů. Nároky stoupají, pokud na pozadí výukového programu má běžet simulační program, umožňující interaktivní simulační hry - ve vývojovém týmu pak musí být i odborníci, kteří jsou schopni navrhnout, formalizovat a odladit příslušné modely. Konečně, pro kreativní propojení různých profesí, podílejících se na tvorbě výukové multimediální aplikace, musí být k

dispozici vhodně zvolené vývojové nástroje (jejichž cena není malá a jejichž ovládnutí vyžaduje určité úsilí a čas). O některých těchto nástrojích a našich zkušenostech s nimi jsme referovali na loňském Medsoftu [7]. Nyní bychom se spíše soustředili na popis vývojového cyklu tvorby mutimediální výukové aplikace.

2. Klíč k úspěchu – kvalitní scénář

Klíčem k úspěchu jakéhokoli výukového programu je dobrý scénář. První, na kom závisí úspěch vytvářené aplikace je tedy zkušený pedagog, který musí mít jasno v tom, co a jakými prostředky chce svým studentům pomocí multimediální výukové aplikace vysvětlit. Základem scénáře je obvykle nějaký výukový text – skripta, kapitola v učebnici apod. Při tvorbě scénáře pro multimediální výukovou aplikaci však musíme myslet i na to, jak se bude výukový program jevit na obrazovce, jaká bude posloupnost jednotlivých obrazovek, jaké bude jejich výtvarné ztvárnění, kde budou umístěny interaktivní elementy, kde se bude zapojovat zvuk, jak budou vypadat jednotlivé animace, kde se případně vloží simulační model a jak bude ovládán, kde se vloží test znalostí, jak bude vypadat, jak se bude vyhodnocovat a jak se bude reagovat na jeho výsledek apod.

Zde se nám osvědčilo využívat postupu, který je znám u kresleného filmu – nakreslit (nejlépe ve spolupráci s výtvarníkem) ***obrazový scénář***, tzv. ***"Story Board"*** – hrubou posloupnost jednotlivých obrazovek, a ke každé obrazovce pak napsat komentář (případně odkaz na příslušnou část textu vytvářeném v klasickém textovém editoru).

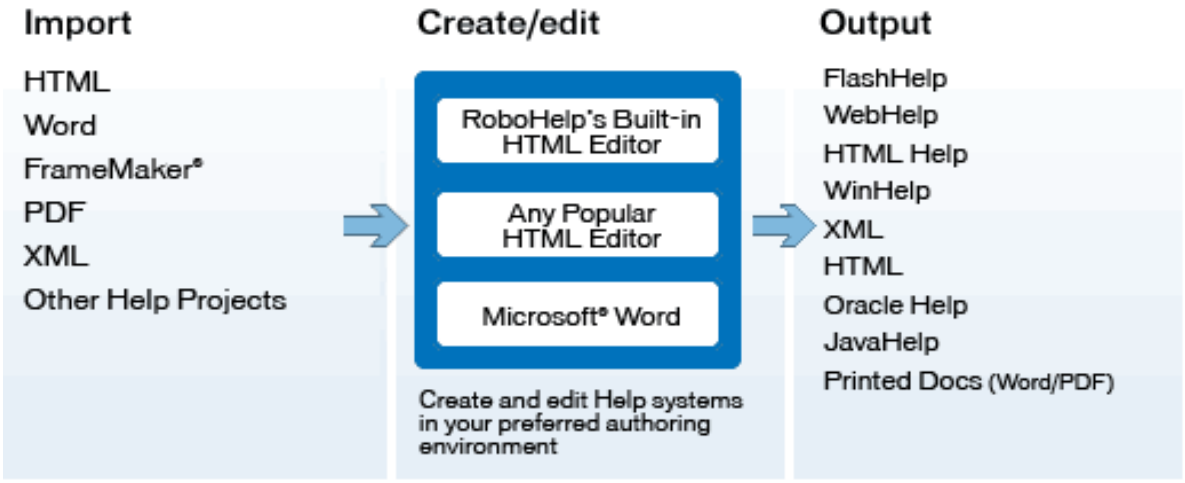
Interaktivní multimediální program však nejsou do počítačové podoby jednoduše přepsaná skripta. Není to ani lineární posloupnost textů, zvuků a pohyblivých obrázků jako kreslený film. Výrazným rysem výukového programu je jeho ***interaktivita*** – a s tím spojená možnost větvení a vzájemného propojení jednotlivých částí. Přetvořit lineární textový a obrazový scénář do scénáře větveného, hypertextovými odkazy provázaného interaktivního programu ovšem není jednoduché.

Prvním problémem, který je nutno vyřešit je způsob způsob ***jak ve scénáři zobrazit vlastní strukturu výukového programu***, zahrnující výklad, interakce s uživatelem, větvení programu apod. Nejjednodušší je v textovém či obrazovém editoru pomocí klasických flowchartů či struktogramů popsat příslušná větvení, rozhodovací bloky apod. s příslušnými odkazy na stránky textu a příslušné obrázky uložené v dalších souborech.

Osvědčilo se vytvářet při psaní scénáře využít schopností ***Microsoft Wordu*** vytvářet příslušné hypertextové odkazy – tím již vlastní scénář dostává jisté rysy budoucí interaktivity. Pro detailnější hypertextové propojení textové části scénáře je vhodné využít i vývojový

nástroj původně určený na tvorbu nápověd ("helpů") – podle našich zkušeností nejvhodnějším vývojovým prostředím, který je v současné době na trhu, je program ***RoboHelp Office*** (nyní ve verzi ***X5***), vyvinutý firmou E-help, kterou nedávno koupila společnost ***Macromedia***.

RoboHelp je velmi výkonný nástroj, v němž můžeme vytvořit textovou část scénáře včetně příslušných hypertextových propojení a větvení. Dobře spolupracuje s Microsoft Office, takže do prostředí RoboHelpu je snadné inkorporovat texty napsané ve Wordu. V prostředí RoboHelp je snadné vytvořit klasickou strukturu "elektronické knihy" včetně rozbalovacího stromu kapitol a rejstříku. V poslední verzi RoboHelpu již byly odstraněny problémy s diakritickými znaménky českého jazyka, které v předchozích verzích při určité neopatrnosti byly příčinou mnoha svízelných okamžiků.



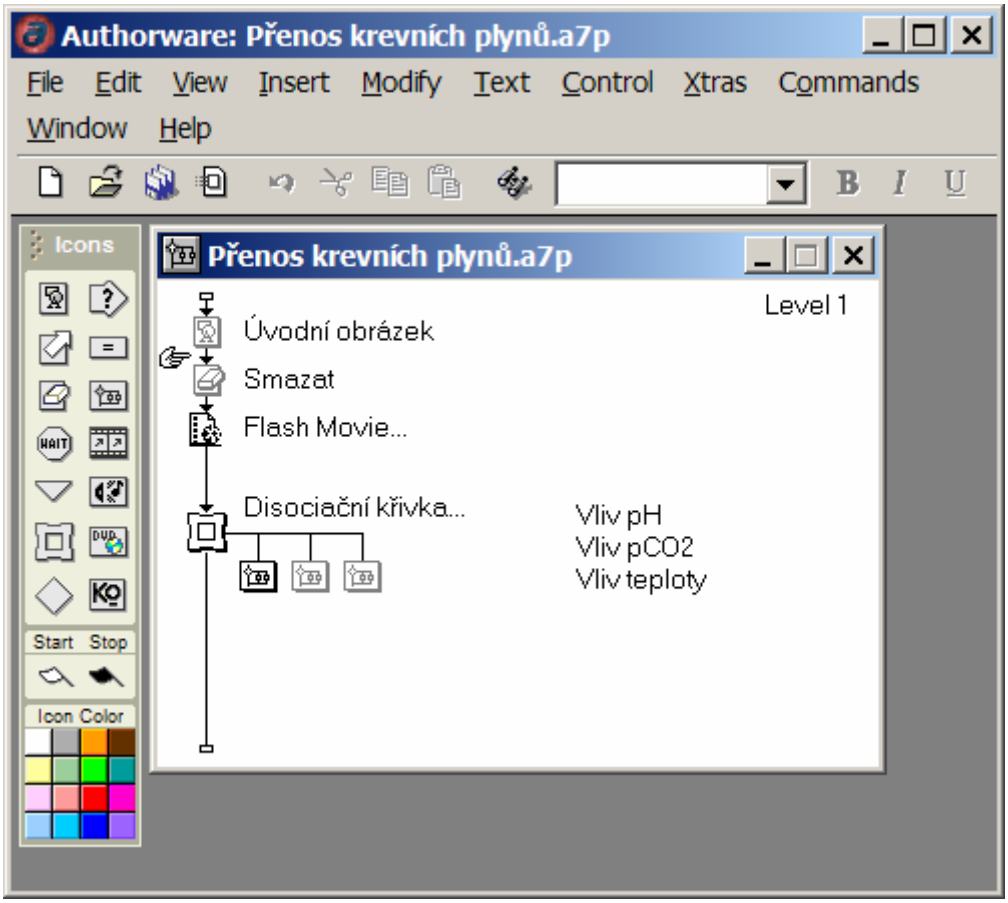
Obr. 1 *Vstupy a výstupy programu Robo Help*

3. Kostra aplikace – RoboHelp a Authorware

RoboHelp můžeme využít nejen pro tvorbu scénáře, ale i jako ***kostru pro vytváření výkladové části vlastní výukové aplikace*** – vstupem je textový soubor s vloženými obrázky ve Wordu, či ve formátu PDF, vstupem může však být i HTML nebo XML soubor, po případě i jiný soubor formátu RoboHelp. RoboHelp sám je možno snadno propojit s Microsoft Wordem pro tvorbu a editaci textů a zároveň i s jakýmkoliv HTML editorem. RoboHelp dobře spolupracuje se všemi produkty firmy Macromedia – můžeme do něj například snadno vkládat interaktivní animace vytvořené v prostředí Macromedia Flash, nebo naopak celý hypertextovými odkazy provázaný výstup z RoboHelpu uložit jako komponentu do Flashe. RoboHelp může také vygenerovat klasický HTML soubor či XML soubor, který můžeme dále využít (obr.1). Vynikající vlastností poslední verze RoboHelpu je také jeho schopnost spolupracovat s prostředím Microsoft .NET – v němž je např. možné

implementovat simulační program, který běží na pozadí výukové aplikace.

Dalším vhodným vývojovým nástrojem pro vytváření kostry aplikace je vývojové prostředí ***Authorware*** opět od firmy Macromedia (nyní ve vyzrálé verzi 7). Výhodou tohoto nástroje je jednoduchý grafický jazyk, kterým pomocí myši snadno zobrazíme větvení a vazby uvnitř jednotlivých hierarchicky uspořádaných prvkůl výukového programu (viz obr. 2). Do jednotlivých prvků je možno vkládat různé komponenty – od obrázků, videa, zvuků, přes html soubory a interaktivní animace vytvořené ve Flashi až po volání externích programů (simulačních modelů).



Obr. 2. *Grafické prvky programu Authorware umožňují vizuálně naprogramovat větvenou strukturu a interaktivitu aplikace.*

4. Svaly aplikace – interaktivní multimediální komponenty

Pro vytváření interaktivních multimediálních komponent je velmi vhodným vývojovým prostředím ***Maromedia Flash***, které umožňuje vytvářet animované interaktivní komponenty, jejichž chování se dá programovat (a v našich aplikacích propojit se simulačním modelem. Vytvořené "flashové" komponenty se dají přehrávat (pomocí

vestavěného interpretu, volně stažitelného z Internetu) přímo ve webových stránkách, nebo se dají využít jako ActiveX komponenty v jiných programech.

Flash prošel poměrně dlouhým vývojem. Původně to byl především program pro pouhé vytváření animovaných obrázků. Postupně se rozšiřovala možnost vytvářené animace řídit pomocí skriptů, jejichž syntaxe se postupně vyvíjela a obohacovala. Poslední verze (prodávaná pod názvem Macromedia Flash MX 2004) má již pořadové číslo 7 a má zabudován objektový řídící jazyk (ActionScript, nyní ve verzi 2.0), syntaxí velmi podobný jazyku Java, který umožňuje poměrně pohodlné a sofistikované řízení chování vizuálních interaktivních elementů.

5. Flash – objektový vizualizační nástroj pro výtvarníky a programátory

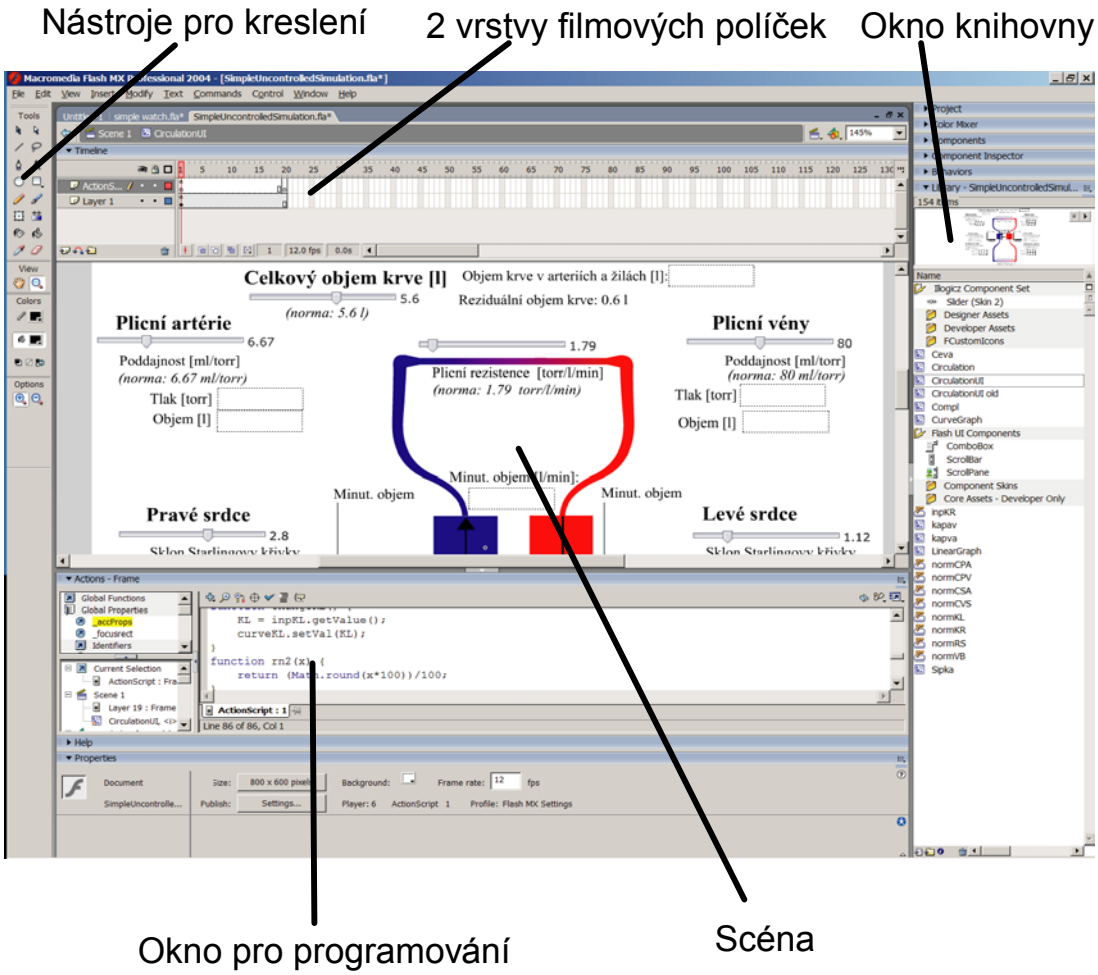
Velký úspěch vývojového prostředí Macromedia Flash je mimo jiné založen na tom, že tvůrcům se poměrně úspěšně podařilo definovat rozhraní pro výtvarníky (vytvářející základní animační prvky) a programátory, kteří těmto komponentám mohou pomocí výše zmíněného objektového jazyka vdechnout interaktivnost.

Náš vývojový tým vytvořil pro vývoj interaktivní *grafiky* ***speciální laboratoř na Výtvarné škole Václava Hollara*** a věnoval velké úsilí tomu, aby profesionální výtvarníky v prostředí Macromedia Flash naučil pracovat. Naše úsilí mělo úspěch. Výtvarníci ztratili ostych před počítačem a záhy pochopili, že "digitálním štětec" představuje jen další nástroj pro kreativní výtvarné vyjadřování, jehož ovládnutí navíc přináší další velké možnosti jejich pracovního uplatnění

Základní komponentou Flashových aplikací je ***film (movie).*** Film je možné dělit na jednotlivé ***scény****,* které je možno postupně či programově (na přeskáčku) přehrávat. Scény jsou tvořené sekvencí jednotlivých ***snímků (frames).*** Filmy (movies) je možno libovolně řetězit - z každého filmu (movie) je možné programově zavolat zavedení dalšího filmu a pak spustit jeho přehrávání. To má výhodu zejména v internetových aplikacích, kdy po zavedení a spuštění první části animace se na pozadí z příslušného serveru stahuje její další část.

Při vytváření počítačových animací se vychází z klasického způsobu ***tvorby kresleného filmu***, kde se pro každé políčko kreslí jednotlivé součásti na průhledných průsvitkách vrstvených na sebe. Některé průsvitky se nemusí pro další políčko překreslovat, nebo se jen o málo posunou (např. pozadí), zatímco jiné (např. postavička) se na každém políčku překreslují, buď celé, nebo jen zčásti, aby postupně vznikl animovaný pohyb.

Ve Flashi (viz obr. 3) je každá scéna tvořena několika ***vrstvami***, které fungují obdobně jako průsvitky při ruční tvorbě kresleného filmu.

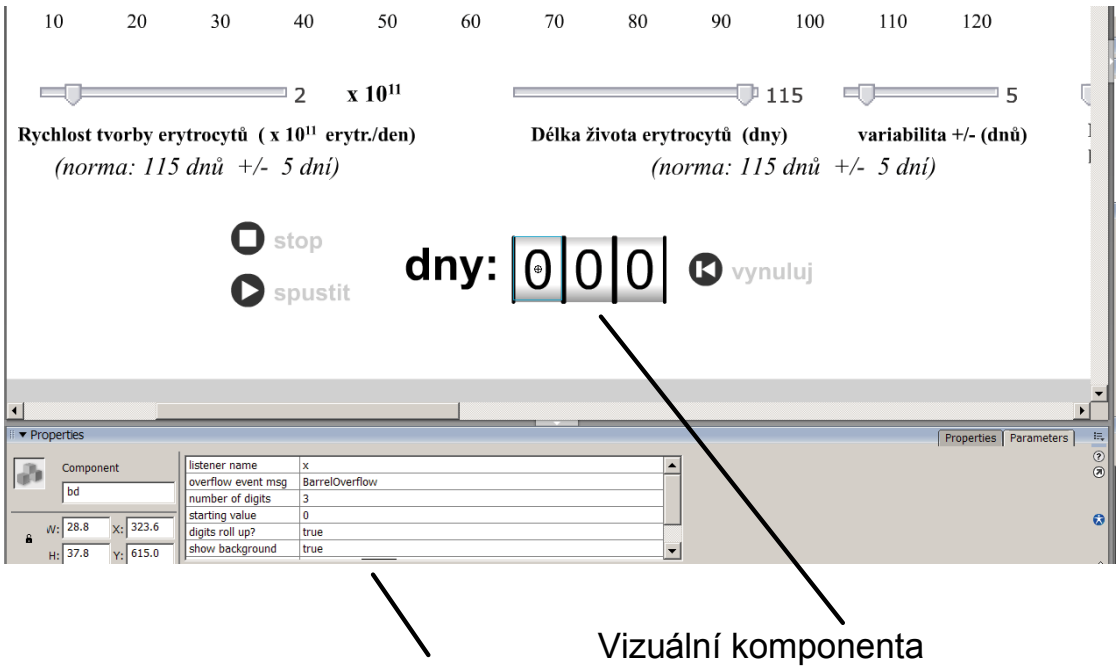


Obr. 3*. Prostředí Macromedia Flash MX poskytuje výtvarníkům nástroje pro kreslení vektorových obrázků. Do jednotlivých vrstev filmových políček je však (jako v příkladě na obrázku) také možno vložit instanci filmového klipu (MovieClipu) z knihovny. Chování jednotlivých vizuálních (i nevizuálních) komponent je možno programovat (v okně pro programování).*

Každé políčko filmu, resp. scény (frame) má tedy několik vrstev do nichž se ukládají jednotlivé obrazové prvky. Tyto vizuální prvky se mohou v každé vrstvě samostatně kreslit – Flash obsahuje poměrně výkonný nástroj pro vytváření ***vektorových obrázků*** (vektorové obrázky je samozřejmě možné i importovat z mnoha jiných externích kreslících programů). Jinou možností je vybrat si z knihovny příslušný obrázek a vytvořit jeho instanci. Instancí však nemusí být jenom statický obrázek. Instancí může být i tzv. ***filmový klip (MovieClip)***, který je vlastně instancovanou třídou již dříve vytvořeného filmu. Tak například při vytváření obrázku letadla můžeme jako jeden jeho element do jedné z vrstev vložit z knihovny instanci filmového klipu s točící se vrtulí. Speciálním druhem filmových klipů jsou tlačítka (Buttons), u nichž je

možno definovat jak jejich vizuální vzhled (při přejetí kurzoru nad tlačítkem a při jeho stisku), tak i chování obsloužením příslušné vyvolané události. Vlastní MovieClip může mít i poměrně složitou hierarchickou strukturu – film, který ho vytváří může obsahovat instance dalších filmových klipů. Tak např. MovieClip auta může v sobě obsahovat MovieClipy jeho točících se kol. Každá instance MovieClipu má své vlastnosti (koordináty umístění na scéně, velikost, přebarvení, průhlednost apod.), které je možné dynamicky měnit z programu. Krom toho, třída MovieClip má celou řadu metod, které můžeme využívat (např. metodu pro detekci kolize z jinou instancí MovieClipu na scéně apod.).

Při tvorbě MovieClipu můžeme také naprogramovat specifické metody, které pak můžeme volat u všech jeho instancí. Tak je možné naprogramovat poměrně složité chování vizuálních komponent. Konečně, poslední dvě verze Macromedia Flash umožňují poměrně jednoduše vytvářet speciální MovieClipy jako **skutečné komponenty**, a jejich vlastnosti pak nastavovat ve speciálním editoru komponent (viz obr. 4) a za běhu volat jejich metody. To otevřelo možnost vytváření (a následné distribuci či prodeji) nejrůznějších vizuálních (ale i nevizuálních) komponent od řady tvůrců (viz www.macromedia,.com).



***Obr. 4**. Flash umožňuje naprogramovat vizuální (ale i nevizuální) komponenty a využívat je obdobně jako v jiných objektových jazycích. Komponenty je možné exportovat a nabízet dalším vývojářům. Postupně se tak rozvíjí trh s komponentami pro Flash*.

Ve vývojovém prostředí je možné jednotlivé filmy vytvořit (jak po grafické, tak i po programátorské stránce), otestovat a přeložit do mezijazyka (ve formě .swf souboru), který je možno interpretovat pomocí volně stažitelného interpretu (tzv. Flash Playeru) a přehrávat buď jako samostatně spustitelnou animaci nebo ji prohlížet přímo v internetovém prohlížeči. Krom toho je možné vytvořený .swf soubor interpretovat pomocí speciální ActiveX komponenty, kterou je možno zabudovat do jiného programu – v našem případě do aplikace vytvářené ve Control Webu nebo v Microsoft Visual Studiu (viz dále). Důležité je, že tato komponenta si s aplikací může vyměňovat zprávy a tak můžeme jinou aplikací pohodlně řídit chování interaktivní animace. Aplikace zároveň může od interaktivní animace přijímat zprávy, referující o zásazích uživatele.

6. Mozek výukové aplikace – simulační modely

Moderní výukové programy nejsou jen multimediální náhradou klasických učebnic, jsou zcela novou výukovou pomůckou. Obsahují totiž ***simulační komponenty***, které umožňují pomocí simulačních her si názorně "osahat" vykládaný problém ve virtuální realitě a přinášejí tak zcela nové možnosti pro vysvětlování složitých problémů. ***Simulační hrou*** je možné bez rizika zkoumat chování simulovaného objektu – přistávat virtuálním letadlem, léčit virtuálního pacienta apod. Ale nejenom to. Modelovaný objekt můžeme rozdělit na jednotlivé subsystémy a testovat jejich chování odděleně i jako součást vyššího celku. Tak např. při studiu složitých fyziologických regulací můžeme např. dočasně odpojit vybrané regulační smyčky a umožnit studentům sledovat reakce těchto subsystémů na změny vstupních veličin (které jsou v reálném organismu ovšem samy regulovány). Tím dovolíme sledovat dynamiku chování jednotlivých subsystémů při postupných změnách pouze jediného vstupu, zatímco jiné vstupy jsou nastaveny na zvolenou konstantní hodnotu (tzv. princip "ceteris paribus"). Postupně pak můžeme jednotlivé dočasně rozpojené regulační vazby opět zapojovat a studovat jejich vliv na chování organismu při nejrůznějších patologických poruchách a reakcích na příslušnou terapii. Podle našich zkušeností právě tento přístup vede ***k lepšímu pochopení*** složitých dynamických jevů v patogenezi nejrůznějších onemocnění a porozumění patofyziologických principů příslušných léčebných zásahů.

Implementace simulačních her do výukového programu není triviální problém. Pro tvorbu simulačních modelů využíváme speciální vývojový nástroj, určený pro tvorbu. ladění a verifikaci simulačních modelů – ***Simulink*** a ***Matlab*** od firmy MathWorks. Tvorba simulačních modelů v biomedicínských vědách je spíše ***výzkumná***, než vývojová práce, která má často multidisciplinární charakter – na jedné straně stojí

systémový analytik - expert na formalizaci a tvorbu simulačních modelů (teoretický fyziolog, vytvářející formalizovaný popis fyziologického systému a testující jeho chování pomocí simulačního modelu). Na druhé straně stojí klasický ***experimentální fyziolog*** či ***klinik***, pro kterého může být popis fyziologického systému pomocí diferenciálních rovnic španělskou vesnicí, ale který dokáže snadno rozpoznat, nakolik odpovídá chování počítačového simulačního modelu biologické realitě. Podle našich zkušeností, problém dorozumění mezi těmito dvěma skupinami specialistů může zásadně usnadnit důsledné využívání tzv. ***simulačních čipů*** [6] při výstavbě simulačního modelu[1]. Verifikované simulační modely (jejichž chování v dané míře přesnosti odpovídá biologické realitě) je pak možno využít ve výukových programech. Ne však přímo v prostředí Matlab/Simulink.

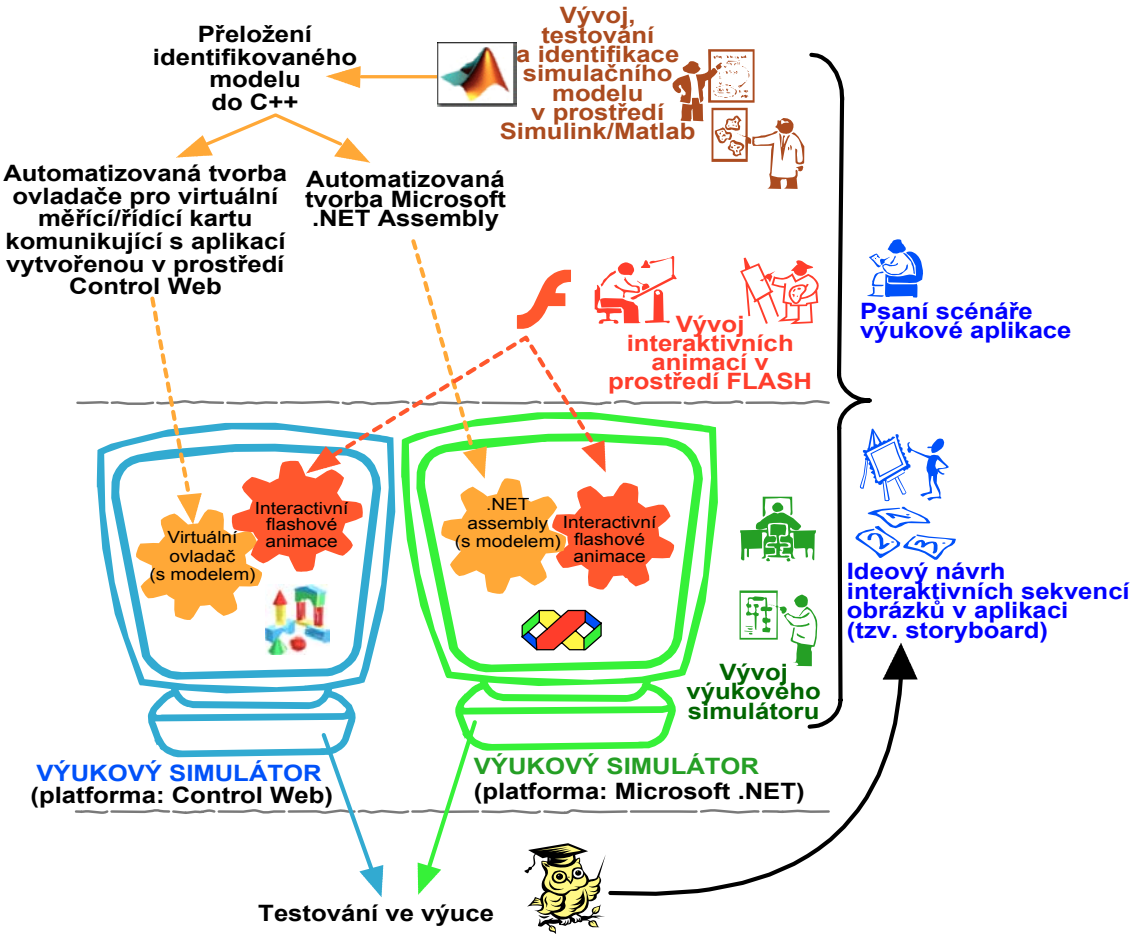
7. Kontejnery pro výukové aplikace

Pro integraci hypertextu, interaktivních animací a dalších multimediálních komponent se simulačními modely klademe na jedné straně velké požadavky na možnosti grafického uživatelského rozhraní, které ale nesmí být příliš náročné na výpočetní zdroje, protože chceme na druhé straně zachovat dostatečnou kapacitu pro numerické výpočty pro běh simulačního modelu. Obdobné požadavky má průmysl – při tvorbě velínů chceme na obrazovce počítače dostatečně flexibilně zobrazovat nejrůznější měřící přístroje a na druhé straně chceme zachovat dostatečnou numerickou kapacitu pro vlastní měření a řízení průmyslové aplikace. Proto jsme se při hledání vývojového nástroje poohlédli po nástrojích původně určených pro tvorbu průmyslových aplikací. Vybrali jsme nakonec nástroj Control Web [4] z dílny Zlínské firmy Moravské přístroje.

Z vývojového prostředí Simulinku vygenerujeme zdrojový program v jazyce C++ ze kterého vytvoříme (pomocí námi vyvinutého nástroje) ***ovladač pro virtuální měřící/řídící kartu*** pro prostředí ***Control Web*** (viz obr. 5). Řadič virtuální karty, obsahující simulační model toto prostředí "ošáli" – výstupy modelu jsou v prostředí Control Web interpretovány

[1] Simulační čipy jsou zvláštním způsobem upravené grafické komponenty vývojového prostředí Simulink, které svým chováním připomínají elektronické čipy: analogicky jako se v elektronických obvodech ve vodičích připojených k jednotlivým pinům elektronických čipů rozvádí elektrický proud, tak se i v Simulinkových schématech k jednotlivým vstupním a výstupním "pinům" simulačních čipů rozvádí informace. Obdobně, jako se v elektrických obvodech dá průběžně měřit napětí a proud pomocí měřících přístrojů, tak i v obvodech tvořených simulačními čipy v grafickém prostředí Simulinku se dají zobrazovat informace pomocí virtuálních displejů a osciloskopů pomocí myši připojených k jednotlivým propojkám. Obdobně jako v elektronických obvodech uživatele čipu zajímá chování čipu a nikoli jeho vnitřek, tak i v simulačních čipech při zkoumání chování je možné testovat pouze chování čipu a nikoli jeho vnitřek, který graficky reprezentuje síť použitých vztahů (graficky vyjádřené příslušné rovnice).

jako měřené signály z technologie a vstupy modelu jsou interpretovány jako řídící signály směřující do technologického zařízení.



Obr 5. *Vývojový cyklus tvorby výukových simulátorů. Na počátku je sestavení scénáře výukového programu včetně návrhu sekvencí obrázků (tzv. storyboard). Pak následuje tvorba simulačních modelů, které budeme využívat v simulátoru, resp. výukovém programu využívajícím simulační hry. Pro tvorbu simulačních modelů využíváme vývojové prostředí Matlab/Simulink od firmy Mathworks. Zároveň v naší laboratoři interaktivní grafiky vytváříme pohyblivé animační obrázky v prostředí Macromedia Flash. Odladěný simulační model je pak implementován ve formě řadiče virtuální měřící/řídící karty do prostředí Control Web od firmy Moravské přístroje, v němž je vytvořeno uživatelské rozhraní. Řadič virtuální karty, obsahující simulační model, toto prostředí "ošálí" – výstupy modelu jsou v prostředí Control Web interpretovány jako měřené signály z technologie a vstupy modelu jsou interpretovány jako řídící signály směřující do technologického zařízení. Flashové animace jsou do vytvářené výukové aplikace umístěny jako Active X komponenty a propojeny se vstupy/výstupy simulačního modelu. Animace pak mohou být řízeny simulačním modelem a do simulačního modelu mohou zároveň přicházet hodnoty vstupů generované interakcí uživatele s flashovou grafikou. Další platformou, kterou jsme v poslední době začali využívat při tvorbě výukových simulátorů je prostředí Microsoft .NET. Do něj umisťujeme jak simulační model ve formě automaticky generovaného .NET assembly ze Simulinku, tak i flashovou interaktivní animaci (případně do něj můžeme umístit i celou aplikaci vytvořenou v nové verzi prostředí Control Web). Důležité je otestování výukových simulátorů ve výuce, které přináší nové požadavky pro revizi, rozšíření či vytvoření dalších výukových simulátorů.*

Flashové animace jsou do vytvářené výukové aplikace umístěny jako Active X komponenty a propojeny se vstupy/výstupy simulačního modelu. Animace pak mohou být řízeny simulačním modelem a do simulačního modelu mohou zároveň přicházet hodnoty vstupů generované interakcí uživatele s flashovou grafikou. Do prostředí Control Web jsme např. implementovali náš simulátor fyziologických funkcí Golem [5,6]

Další platformou, kterou jsme v poslední době začali využívat jako kontejner při tvorbě výukových programů je prostředí ***Microsoft .NET***, keré mimo jiné také poskytuje dostatečně silnou podporu pro tvorbu webových aplikací. Do prostředí Microsoft .NET umisťujeme jak ***simulační model ve formě automaticky generovaného .NET assembly ze Simulinku***, tak i ***flashovou interaktivní animaci*** (případně do něj můžeme umístit i celou aplikaci vytvořenou v nové verzi prostředí Control Web).

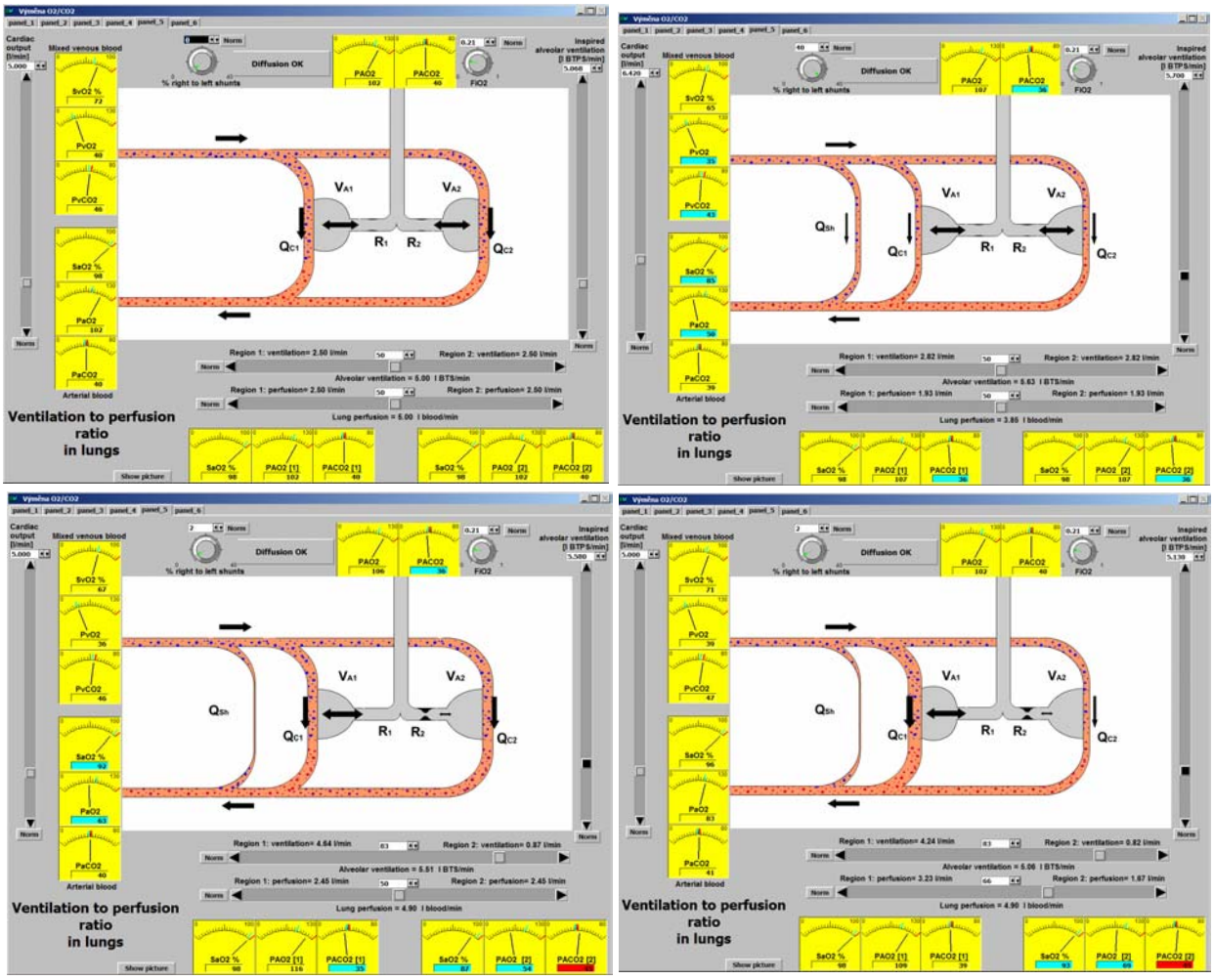
9. Propojení multimediálních prvků a simulačních modelů

Simulační model je tedy propojen s flashovu animací. ***Animace pak mohou být řízeny na základě výstupů implementovaného simulačního modelu*** - např. schematický obrázek cévy se může roztahovat nebo komprimovat, plicní sklípek může hlouběji či mělčeji "dýchat", ručička měřícího přístroje se může pohybovat a průběžně zobrazovat hodnotu nějaké výstupní proměnné modelu atd.

Na druhé straně můžeme ***přes vizuální prvky*** vytvořené ve Flashi (nejrůznější tlačítka, knoflíky, táhla apod.) ***do simulačního modelu zadávat nejrůznější vstupy***.

Příklad využití flashové animace ve výukovém programu využívajícím simulační hru je zobrazen na obr. 6. Interaktivní animace, vytvořená v prostředí Macromedia Flash byla umístěna do kontejneru pro ActiveX komponentu v programu vytvořeném v prostředí Control Web. Ručičky virtuálních přístrojů zobrazují příslušné fyziologické hodnoty čtené z běžícího simulačního modelu na pozadí a zároveň se příslušně mění chování flashová animace.

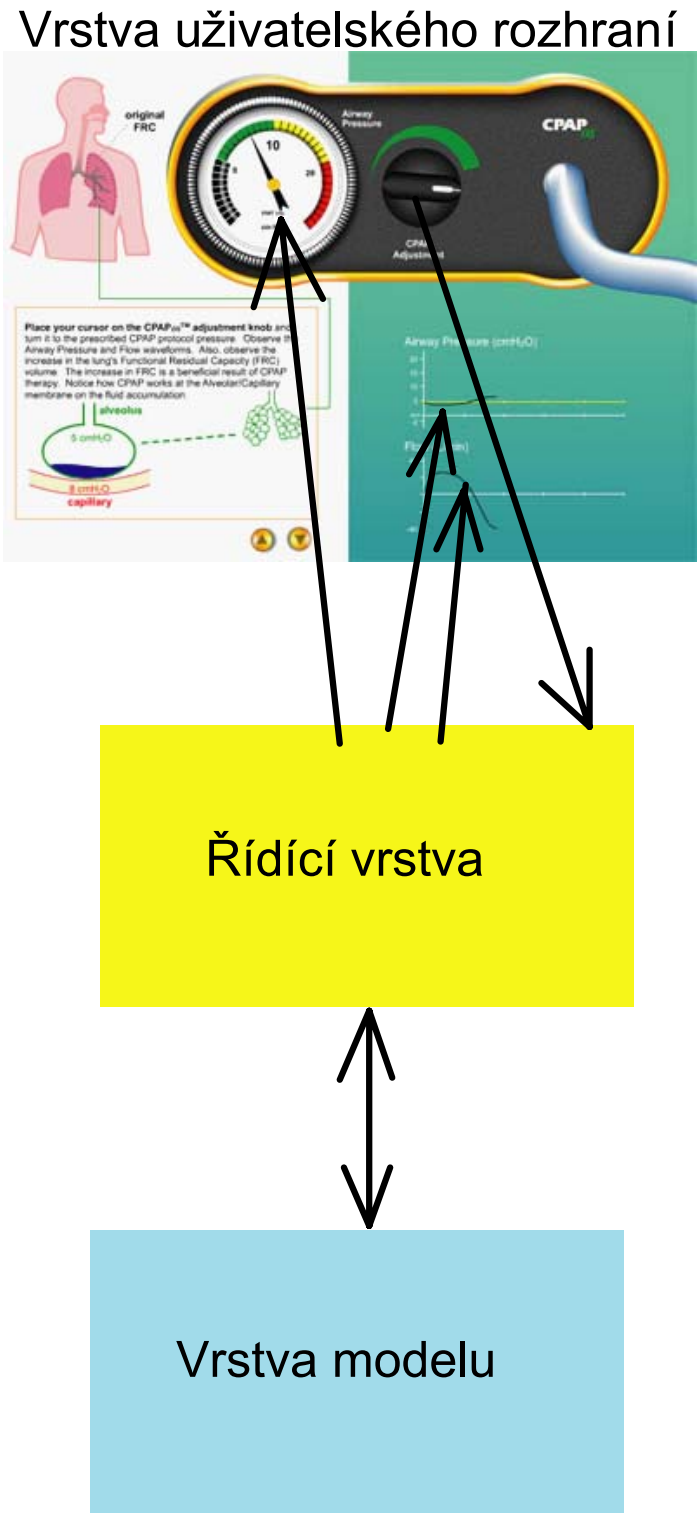
Model běžící na pozadí (realizovaný ve formě řadiče virtuální měřící/řídící karty v případě aplikace v Control Webu, nebo jako .NET assembly v případě aplikace implementované v Microsoft .NET) komunikuje s vizuálními elementy na popředí. V případě flashových animací je změna zobrazovaných hodnot či požadavek na změnu vstupních dat modelu realizován pomocí událostí, které jsou příslušně obslouženy a změní nastavení vizuálních elementů či vstupní hodnoty modelu.



Obr. 6. *Ukázka různých stavů vizuálního rozhraní v simulační hře ve výukovém programu poruch respirace. Uživatelské rozhraní, vytvořené v prostředí Control Web s využitím flashové animace, schematicky zobrazuje různé stupně ventilace a perfúze dvou částí plic. Student si "hraje" s pohyblivým obrázkem a model na pozadí propočítává příslušné fyziologické parametry, zobrazované jako hodnoty virtuálních měřících přístrojů (v Control Webu). Zároveň se mění "hloubka dýchání" a "velikost prokrvení" plic na schematickém obrázku tvořeném flashovou animací.*

10. UCM architektura

V případě složitější architektury může být logika změn poměrně složitá, proto je vhodnější mezi vrstvu vizuálních elementů a vrstvu simulačního modelu vložit **řídící vrstvu**, kde se **na jednom místě řeší veškerá logika komunikace uživatelského rozhraní s modelem a kde je ukládán i příslušný kontext** (obr. 7). Toto uspořádání je nezbytné při složitějších modelech a simulátorech, jejichž uživatelské zobrazení je reprezentováno mnoha virtuálními přístroji na více propojených obrazovkách. Výhody tohoto uspořádání zvláště vyniknou při modifikacích jak modelu, tak i uživatelského rozhraní. V literatuře [2] se hovoří o tzv. **UCM architektuře** výstavby simulátorůl (User interface – Control layer – Model layer).



***Obr.** 7 Tzv. UCM architektura při tvorbě simulátorů. Mezi vrstvu modelu a vrstvu uživatelského rozhraní je vhodné vložit řídící vrstvu kam jsou směrovány veškeré zprávy a události vznikající ve virtuálních přístrojích uživatelského rozhraní a kam je zároveň směrována veškerá komunikace s modelem. V této vrstvě se řeší veškerý kontext zobrazovaných dat a příslušné požadavky na komunikaci s modelem. Veškerá logika zobrazování a komunikace je pak soustředěna do jednoho místa což podstatně ušetří čas při modifikacích uživatelského rozhraní nebo změnách modelu.*

Velmi se nám osvědčilo využít v tomto jádře **stavový automat** (která nám nejlépe zapamatuje příslušný kontext). Prostředí Matlab/Simulink nabízí realizaci stavových automatů prostřednictvím speciálního toolboxu **Stateflow** bezprostředně propojitelným se simulačním modelem v Simulinku.

Stavový automat je však možno realizovat i na straně uživatelského rozhraní – přímo ve Flashi. Tento automat pak bezprostředně komunikuje s vizuálními objekty ve Flashi. Obdobným způsobem konstruují simulátory Kayne a Castillo [2]. Od těchto autorů jsme také převzali komponentu stavového automatu, naprogramovanou přímo v ActionScriptu v prostředí Macromedia Flash MX.

11. Závěr

Zdá se, že pomalu končí doba, kdy vytváření výukových programů bylo otázkou entuziasmu a píle skupin nadšenců. Tvorba moderních výukových aplikací je náročný a komplikovaný projekt, vyžadující ***týmovou spolupráci*** řady profesí – profesí – od zkušených učitelů, jejichž scénář je základem kvalitní výukové aplikace, přes systémové analytiky, kteří ve spolupráci s profesionály daného oboru jsou odpovědni za vytvoření simulačních modelů pro výukové simulační hry, výtvarníky, kteří vytvářejí vnější vizuální podobu, až po programátory, kteří celou aplikaci "sešijí" do výsledné podoby. Aby tato interdisciplinární kolektivní tvorba byla efektivní, je nutno pro každou etapu tvorby využívat ***specifické vývojové nástroje***, s dostatečnou technickou podporou, které umožňují komponentovou tvorbu simulačních modelů, vytváření interaktivních multimédií a jejich závěrečné propojení podle daného scénáře do kompaktního celku [7,8].

10. Literatura

[1] Hall, B., Wann, S. (2003): *Object-oriented programming with ActionScript.* New Riders, Boston, Indianopolis, London, Munich, New York, san Francisco, 2003, ISBN 0-7357-1183-6.

[2] Kaye J., Castillo D. (2003): *Flash MX for interactive simulation*. Thomson Delmar Learning Inc. New York, 2003, ISBN 1-4018-1291-0.

[3] Moock C. (2002):: *ActionScript for Flash MX. The Definitive Guide.* O' Reilly. Bejing,Cambridge,Farnham,Köln, Paris,Taipei, Tokyo, 2002, ISBN 0-596-00396-X.

[4] Kofránek, J., (2000): *Control Web 2000 – objektové vývojové prostředí pro distribuované aplikace reálného času*. In: Objekty' 2000 (editor: Vojtěch Merunka, Vladimír Sklenář). Česká zemědělská univerzita v Praze, 2000. ISBN 80-213-0682-3, str. 151-163.

[5] Kofránek, J., L. D. Anh Vu, H. Snášelová, R. Kerekeš, T. Velan, (2001): *GOLEM – Multimedia simulator for medical education*. In: *Studies in Health Technology and Informatics.*, vol. 84. MEDINFO 2001, Proceedings of the 10th World Congress on Medical Informatics. (editors: V.L. Patel, R. Rogers, R. Haux), IOS Press, Amsterdam, Berlin, Oxford, Washington DC, 2001, pp. 1042-1046.

[6] Kofránek, J., M.Andrlík, T. Kripner, J. Mašek, T. Velan, (2002): *Simulation Chips for GOLEM – Multimedia Simulator of Physiological Functions*. In: *Simulation in the Health and Medical Sciences 2002*. (editors: James G. Anderson, M. Katzper). Society for Computer Simulation International, Simulation Councils, San Diego, 2002, str. 159-163.

[7] Kofránek J., Andrlík M. Kripner T., Mašek J., Stodůlka P. (2003): *„Od umění k průmyslu“ – propojení technologií při tvorbě lékařských výukových programů.* Medsoft 2003, s.43-56. ISBN 80-86742-00-8.

[8] Kofránek, J.,T. Kripner, M. Andrlík, J. Mašek (2003): *Creative connection between multimedia, simulation and software development tools in the design and development of biomedical educational simulators*, In*: Simulation* Interoperability Workshop, Position papers, Volume II, Orlando, FALL 2003, paper 03F-SIW-102, pp. 677 - 687, ISBN 1-930638-32-9

Poděkování

Práce na vývoji lékařských simulátorů a tvorba výukových programů se simulačními hrami je podporována Výzkumným záměrem MSM-111100008 (physiome.cz) a Rozvojovým programem MŠMT č. 394.

Jiří Kofránek
Biokybernetické oddělení Ústavu patologické fyziologie 1. LF UK
U nemocnice 5
128 53 Praha 2
tel: 22496 2793
fax:22496 0503
e-mail: kofranek@cesnet.cz