TECHNOLOGIE TVORBY WEBOVÝCH SIMULÁTORŮ

David Polák, Filip Ježek, Jan Šilar, Jiří Kofránek

Abstrakt

Vytvořili jsme novou technologii tvorby webových simulátorů BodyLight.js, která kombinuje moderní internetové technologie (JavaScript, ECMA6, HTML5, WebAssembly), moderní na rovnicích založený modelovací jazyk (Modelica), nové standardizované rozhraní simulačních modelů (Functional Mock-up Interface, verze 2), simulační runtime běžící v prohlížeči, využívají technologii WebAssembly a grafické vizualizace, vytvářené pomocí Adobe Animate. Na tvorbu finální aplikace jsme napsali nástroj nazvaný Composer, který umožňuje vizuální tvorbu webových stránek, propojení interaktivních animací a ovládacích prvků (posuvníků, tlačítek a přepínačů) se simulačním modelem do výsledné výukové aplikace. Simulátor je realizován jako interaktivní graf či obrázek propojený se simulačním modelem v pozadí. Výsledkem je webová aplikace s interaktivními simulátory spustitelnými přímo v internetovém prohlížeči.

Klíčová slova

Internet, simulace, výuka, webové simulátory

1 Úvod

Kdo si dnes vzpomene na kdysi běžnou součást kancelářského vybavení – psací stroj. Místo nich jsou dnes v kancelářích počítače, které se staly běžnou součástí kancelářského vybavení. Obdobně, jako počítače vytlačily psací stroje z kanceláří, lze očekávat, že počítače, tablety, chytré telefony propojené na vysokorychlostní internet se ve velmi blízké budoucnosti stanou běžnou a široce využívanou výukovou pomůckou.

Pro použití počítačů ve výuce je klíčovým limitujícím faktorem dostatek vhodných výukových programů. Jejich tvorba není jednoduchá. Zdaleka nestačí jen převést skripta (případně doplněné multimediálními komponenty) do počítačem prezentovatelné podoby. Zásadní inovací je začlenění simulačních komponent a simulátorů do výukových aplikací.

Před téměř dvěma a půl tisíci lety Konfucius napsal: „Co slyším, to zapomenu, co spatřím, to si pamatuji, co dělám, tomu rozumím." Tuto starou čínskou moudrost potvrzují i moderní metody učení nazývané někdy jako „learning-by-doing," (učení praxí) kde mají velké uplatnění simulační hry. Simulační hrou je možné bez rizika otestovat chování simulovaného objektu – např. zkusit přistávat virtuálním letadlem nebo, v případě lékařských simulátorů, léčit virtuálního pacienta či testovat chování jednotlivých fyziologických subsystémů. Spojení internetu a interaktivního multimediálního prostředí se simulačními modely přináší zcela nové pedagogické možnosti zejména pro vysvětlování složitě provázaných vztahů. Simulační hry dosažitelné přes internet pomáhají studentům pochopit, jak pracuje organismus v normě a v patologii. V zapojení multimediálních výukových her do výuky nachází své moderní uplatnění staré krédo Jana Amose Komenského „Schola Ludus" – tj. „škola hrou" [1], které tento evropský pedagog razil již v 17. století.

2 Tvorba webových simulátorů – propojení profesí i nástrojů

Tvorbě webově dostupných výukových aplikací se věnujeme řadu let. Internet a interaktivní grafika řízená modelem na pozadí – to je skutečná škola hrou pro 21. století, která umožňuje vytvořit simulační hry dosažitelné přes internet, které jako norimberským trychtýřem pomohou studentům pochopit dynamické souvislosti. Proto jsme vytvořili projekt internetového atlasu fyziologie a patofyziologie, soustřeďujícího internetem dostupné simulační hry propojené s výkladovými kapitolami [2]. Atlas je dostupný na adrese *http://physiome.cz/atlas*.

Tvorba kvalitního výukového softwaru, který by dokázal využít potenciál, který rozvoj informačních a komunikačních technologií přinesl, dnes nestojí na píli a nadšení jednotlivců. Je to náročný a komplikovaný projekt, vyžadující týmovou spolupráci řady profesí – od zkušených učitelů, jejichž scénář je základem kvalitní výukové aplikace, přes systémové analytiky, kteří jsou ve spolupráci s profesionály daného oboru 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 mezioborová spolupráce byla účinná, je zapotřebí pro každou etapu vývoje mít k dispozici řadu specifických vývojových nástrojů a metodologií, které práci jednotlivých členů týmu usnadní a pomohou jim překonat mezioborové bariéry. K vytvoření i ovládnutí těchto nástrojů je zapotřebí věnovat značné úsilí, které se ale nakonec vyplatí.

Propojením různých profesí a technologií se tvorba výukového softwaru stává efektivnější, pozvolna přestává být výsledkem kreativity a pracovitosti jedinců a stále více získává rysy inženýrské konstrukční práce [3,4].

3 Od ControlWebu k Silverlightu – naše původní technologie tvorby webových simulátorů

Během tvorby výukových simulátorů se využitelné technologie doslova měnily pod rukama.

Objevovaly se nové technologie usnadňující tvorbu multimediálních animací, propojitelných se simulačním modelem na pozadí. Zároveň se objevily i simulační nástroje, usnadňující vytváření složitých hierarchicky organizovaných modelů. V průběhu let jsme naši technologii tvorby simulátorů několikrát zásadně změnili.

Tyto změny musely být dostatečně dobře promyšlené, protože každá změna technologie znamená zpočátku zdržení, způsobené tím, že členové vývojového týmu musí nejprve tuto technologii „vstřebat", což určitou dobu trvá.

Pro propojení jednotlivých vývojových nástrojů mezi sebou jsme si často museli vytvořit vlastní softwarové nástroje (např. pro automatizaci konverze modelů, vyvinutých v modelovacích nástrojích Simulink nebo Modelica do cílové platformy vytvářených simulátorů). A postupném vývoji naší technologie jsem pravidelně referovali na seminářích Medsoft.

Přehled námi dosud používaných technologií při vývoji simulátorů uvádí tabulka 1. Porovnání výhod a nevýhod jednotlivých technologií uvádí tabulka 2.

Technologie používané pro vývoj interaktivních simulátorů					
Platforma	**Modelovací nástroje**	**Konverze modelu do simulátoru**	**Nástroje pro vývoj simulátorů**	**Animační nástroje**	**Distribuce simulátoru**
Control Web	Simulink / Matlab	Automatická	vývojové prostředí Control Web, C++	Animační nástroje v prostředí Control Web, Adobe Flash	CD ROM s instalátorem nebo stažení instalačního programu z Internetu
Adobe Flash	Simulink / Matlab	Manuální	jazyky ActionScript (Adobe Flash, Adobe Flash Builder)	Adobe Flash	Internetový prohlížeč (se zásuvným modulem Flash Player
.NET	Simulink / Matlab	Automatická	Microsoft Visual Studio	Adobe Flash	Instalace do lokálního počítače z Internetu
Microsoft Silverlight	Modelica	Automatická	Microsoft Visual Studio	Microsoft Expression Blend, Animtester	Internetový prohlížeč (se zásuvným modulem Silverlight).
BodyLight.js	Modelica	Automatická	JavaScript (ECMAScript 6),		
Composer	Adobe Animate	Internetový prohlížeč			

Tabulka 1 – Technologie, které jsme používali při tvorbě interaktivních simulátorů.

Porovnání technologií pro vývoj interaktivních simulátorů		
Platforma	**Výhody**	**Nevýhody**
Simulátory na platformě Control Web	Jednoduché vytvoření uživatelského rozhraní z rozsáhlé nabídky virtuálních přístrojů. Automatické generování zdrojového programu pro simulační jádro ze Simulinku	Pracuje pouze pod operačním systémem MS Windows Příliš technický tvar uživatelského rozhraní, připomínající spíše velín průmyslového zařízení než obrázek z fyziologické učebnice. Omezený počet možných flashových animací. Nutnost distribuce s instalátorem provozního programu Control Web. Nutnost instalace na počítači klienta.
Simulátory na platformě Adobe Flash	Běží přímo v internetovém prohlížeči (s instalovaným zásuvným modulem Flash Player) na různých operačních systémech Bohaté možnosti vytváření animovaného uživatelského rozhraní.	Nutnost manuální konverze modelu ze Simulinku do jazyka ActionScript. Relativně pomalé simulační jádro (vhodné pro relativně malé modely)

Porovnání technologií pro vývoj interaktivních simulátorů		
Platforma	**Výhody**	**Nevýhody**
Simulátory na platformě .NET	Automatické generování zdrojového programu pro simulační jádro ze Simulinku. Rychlé simulační jádro umožňuje vytváření výpočetně náročných simulátorů.	Běží pouze pod MS. Windows. Nutnost instalace na počítači klienta (pomocí technologie ClickOnce je však možno aplikaci nainstalovat a spustit z internetového prohlížeče)
Simulátory na platformě Silverlight	Běží přímo v internetovém prohlížeči (s instalovaným zásuvným modelem Silverlight) na různých operačních systémech. Rychlé simulační jádro umožňuje vytváření výpočetně náročných simulátorů.	Microsoft ukončil podporu Silverlightu, Silverlight není v současných prohlížečích podporován, plugin lze nainstalovat pouze do Internet Exploreru
Simulátory na platformě Silverlight	Deklarativní tvorba modelů (pracující přímo s rovnicemi) v jazyce Modelica. Automatické generování zdrojového programu pro simulační jádro z Modeliky. Společné prostředí pro simulační jádro i interaktivní animace. Při vytváření interaktivních animací nástroj Animtester poskytuje rozhraní mezi výtvarníky a programátory	
Simulátory na platformě BodyLight.js	Běží přímo v internetovém prohlížeči (bez zásuvných modulů) Rychlé simulační jádro s využitím standardu WebAssembly umožňuje vytváření výpočetně náročných simulátorů. Deklarativní tvorba modelů (pracující přímo s rovnicemi) v jazyce Modelica. Automatické generování zdrojového programu pro simulační jádro z Modeliky. Společné prostředí pro simulační jádro i interaktivní animace. Při vytváření interaktivních animací nástroj Composer usnadňuje vytváření výsledné aplikace	Nutnost licenčních vývojových nástrojů (Adobe Animate) pro vývoj animačních komponent

Tabulka 2 – Porovnání technologií, které jsme používali při tvorbě interaktivních simulátorů.

3.1 Control Web

V první verzi technologie v polovině devadesátých let jsme simulátory vytvářeli ve **vývojovém prostředí Control Web**, původně určeném pro řídící a měřící aplikace v průmyslu. Modely jsme navrhovali, odlaďovali a identifikovali v tehdy relativně novém prostředí **Matlab/Simulink**. Když jsme nakonec vytvořili softwarový nástroj, který automaticky generoval zdrojový kód ovladače virtuální měřicí/řídící karty pro Control Web, měli jsme možnost snadné a pohodlné aktualizace simulačního jádra vytvářených výukových simulátorů. Touto technologií jsme mimo jiné vytvořili simulátor fyziologických funkcí Golem [5,6]. V atlase fyziologie je touto metodou vytvořen simulátor ledvin [7], viz *http://physiome.cz/atlas/sim/Ledvina/Virtual_Kidney-Install.msi*.

3.2 Adobe Flash

Časem se ale ukázalo, že naše vývojové simulátory vytvářené ve vývojovém prostředí ControlWeb připomínaly spíše velín automatizované průmyslové linky, než elektronický nástroj pro lékařskou výuku. Zároveň se objevila možnost pomocí programu Macromedia Flash (později **Adobe Flash**) vytvářet ovladatelné animované obrázky, propojitelné pomocí technologie ActiveX se svým okolím. Navázali jsme proto úzkou spolupráci se Střední uměleckou školou Václava Hollara a věnovali velké úsilí naučit pracovat s tímto nástrojem profesionální výtvarníky. Iniciovali jsme založení Vyšší odborné školy se zaměřením na obor interaktivní grafika, kde nyní také učíme (*http://www.hollarka.cz*). To nám otevřelo možnosti vkládat do uživatelského rozhraní výukových simulátorů

graficky atraktivní obrázky, které jako loutky na nitích ovládal simulační model na pozadí. Naše simulátory pak obsahovaly interaktivní obrázky jak z lékařské učebnice. Touto technologií jsme začali vytvářet náš internetový Atlas fyziologie a patofyziologie [2,8], který je volně dostupný na adrese *http://www.physiome.cz/atlas*. Pro realizaci simulátorů jsme používali vývojové prostředí jazyka **ActionScript** a **FlashPlayer v internetovém prohlížeči**.

Simulátory realizované v prostředí Adobe Flash pak bylo možné spouštět přímo v okně internetového prohlížeče. Příkladem je simulátor svalu [9,10] – viz *http://www.physiome.cz/atlas/sval/svalCZ/svalCZ.html* nebo simulátor acidobazické rovnováhy plazmy [11] – viz *http://www.physiome.cz/atlas/acidobaze/02/ABR_v_plazme1_2.html*.

3.3 Microsoft .NET

Protože modely jsme vytvářeli ve vývojovém prostředí Matlab/Simulink, museli jsme pak odladěné modely ručně převádět do jazyka ActionScript. Navíc jazyk ActionScript je interpretovaný nikoli kompilovaný, proto simulační jádro modelů vytvořených v ActionScriptu je relativně pomalé. Platforma AdobeFlash je proto vhodná pouze pro simulátory využívající relativně jednoduché modely. Pro vývoj složitějších simulátorů jsme začali používat vývojové prostředí **Microsoft .Net** a animace vytvořené v prostředí AdobeFlash jsme se simulačním jádrem propojovali přes technologii ActiveX. Pro automatizaci převodu modelu z vývojového prostředí Matlab/Simulink do prostředí .NET jsme vytvořili nové softwarové nástroje [12,13]. V této technologii byl např. vytvořen simulátor přenosu krevních plynů, který dodnes využíváme ve výuce [14]. Simulátor je volně dostupný jako jedna z aplikací našeho internetového Atlasu fyziologie a patofyziologie – *http://physiome.cz/atlas/sim/BloodyMary_cs/*. Pomocí této technologie jsme také vytvořili interaktivní výukový program propojující text a simulační hry věnovaný výkladu obecných principů regulačních obvodů a jejich aplikaci na fyziologické regulace [15], program je dostupný na *http://physiome.cz/atlas/sim/RegulaceSys/*.

3.4 Microsoft Silverlight

V roce 2007 se objevila nová technologie **Silverlight,** kterou Microsoft reagoval na tehdy velmi rozšířený Adobe Flash. Nová technologie od Microsoftu svými možnostmi Flash v mnohém překonala. V Silverlightu bylo možné vytvářet numericky náročné simulátory s přitažlivým grafickým rozhraním spustitelné přímo v internetovém prohlížeči. To se jevilo velmi slibné, a proto jsme v roce 2008 zásadně změnili technologickou bázi řešení tohoto projektu. Abychom ale novou technologii mohli využít, museli jsme naučit naše spolupracující výtvarníky pracovat ve vývojovém prostředí **Microsoft Expression Blend** (které je pro ně náročnější než graficky více intuitivní prostředí Adobe Flash). Zároveň jsme vytvořili softwarový nástroj **"Animtester"**, který umožnil oddělit vývojové prostředí určené pro výtvarníka od prostředí pro programátora. Tento nástroj výtvarníkům usnadnil tvorbu animací snadno propojitelných se simulačním modelem na pozadí. Na trhu se objevila i nová (tzv. akauzální) simulační prostředí, která umožňují jednotlivé části modelu popisovat přímo jako soustavu rovnic a nikoli jako algoritmus řešení těchto rovnic. To přineslo zásadní průlom v tvorbě modelů. Ukázalo se, že je daleko efektivnější začít naše modely vytvářet v akauzálním prostředí využívajícím simulační jazyk Modelica, než se spoléhat jen na nové akauzální knihovny v prostředí Simulink. Při vývoji simulačních modelů jsme se proto přeorientovali z vývojového prostředí Matlab/Simulink na vývojové prostředí pro jazyk **Modelica**. Odladěné modely v jazyce Modelica pak překládáme v prostředí Open Modelica do jazyka C# (generátor kódu do C# jsme vytvořili v rámci našeho členství v mezinárodním konsorciu Open Source Modelica Consortium (*http://www.ida.liu.se/labs/pelab/modelica/OpenSourceModelicaConsortium.html*). Modely v C# (kombinované s numerickým řešičem) pak umožní propojit model do vyvíjené aplikace na platformě Silverlight, a simulátor pak může být distribuován prostřednictvím internetu a spouštěn přímo v prostředí internetového prohlížeče se zásuvným modulem SilverLight. Výsledkem tedy byla nová technologie tvorby webových simulátorů (včetně vytvoření sady softwarových nástrojů, umožňujících „bezešvé" propojení vývojového nástroje pro tvorbu modelů využívajícího akauzální modelovací jazyk Modelica, vývojových nástrojů pro tvorbu interaktivní počítačové grafiky a vývojového prostředí pro tvorbu webových aplikací) [16,17]. V této technologii byl např. vytvořen simulátor krevního oběhu [18,19], který využíváme ve výuce studentů [14]. Simulátor je dostupný na adrese *http://physiome.lf1.cuni.cz/SimpleCirculation/*.

4 Tvrdý náraz

Technologie Microsoft Silverlight nám umožnila vytvářet multimediální interaktivní simulátory spustitelné přímo v internetovém prohlížeči. Modely jsme přitom mohli pohodlně vyvíjet v akauzálním prostředí jazyka Modelica. Pomocí námi vyvinutého Animtesteru mohli grafici vytvářet interaktivní animace v prostředí Microsoft Expression Blend snadno napojitelné na vstupy a výstupy modelu na pozadí. Výsledkem pak byly animované obrázky řízené modelem na pozadí a celá aplikace potřebovala pouze internetový prohlížeč se zásuvným modulem Silverlight.

A právě v tom byl zakopaný pes. Microsoftu se nepodařilo prosadit rozšíření svého Silverlightu na jiné platformy.

Společnosti Microsoft, která nakonec v roce 2015 dotáhla Silverlight do páté verze, oznámila ukončení podpory tohoto produktu. Takže dnes již do nového "microsoftího" prohlížeče Microsoft Edge zásuvný modul Silverlight nenainstalujete. Silverlightové aplikace (např. náš výukový model krevního oběhu) jsou spustitelné pouze ve starém Internet Exploreru.

Krom toho, společnost Adobe ohlásila konec podpory zásuvného modulu Flash Player v roce 2020.

Znamenalo to, že při tvorbě webových simulátorů jsme se ocitli opět na začátku, a stáli jsme před úkolem vytvořit zcela novou technologii, která nám umožní v tvorbě webových simulátorů pokračovat.

5 Naše nová technologie – BodyLight.js

Od počátku tohoto století je zřejmé že webová platforma je jediný společný prvek uživatelských výpočetních zařízení – od chytrých telefonů, přes tablety až k počítačům bez ohledu na operační systém, na kterém pracují. Chceme-li vytvořit na platformě nezávislou technologii simulátorů, jsou internetové prohlížeče jedinou možností, jak tento cíl realizovat.

Internetové prohlížeče se v posledních několika letech zásadně změnily. Webový prohlížeč a vykreslovač se v současné době staly tím, čím byl operační systém v minulém století. Není tedy žádným překvapením že jsme si pro implementaci simulátorů zvolili právě webové technologie.

Náš přístup ke skládání webových simulátorů jsme nazvali **Bodylight.js**. Jedná se o kombinaci:

- moderních internetových technologií,
- modelovacího jazyka,
- simulačního runtimu
- a grafických vizualizací.

5.1 Internetové technologie

5.1.1 HTML 5 a ECMAScript 6

Na poli internetových technologií se v posledních letech dějí velké změny, poháněny novými verzemi internetových standardů, jako například standard HTML 5 v roce 2014, který kromě mnoha dalších věcí, přidal vektorové a rastrové kreslící plátno. V roce 2015 vznikla norma ECMAScript 6 výrazně zlepšující syntaktickou příjemnost jazyka JavaScript.

Tyto technologie změnily dnešní prohlížeče k nepoznání, pryč jsou dny kdy jsme byli nuceni spoléhat na proprietární řešení od třetích stran jako Flash nebo Silverlight, jenom proto abychom mohli v prohlížeči nasazovat aplikace podobné těm desktopovým.

Díky vytrvalému mezinárodnímu úsilí dobrovolníků, standardizačních skupin a velkých internetových společností je dnes možné implementovat webové aplikace jako nikdy předtím. Jak HTML tak ECMAScript se nadále vyvíjejí, tyto standardy jsou v dnešní době na ročním cyklu vydávání nových verzí, které jsou adoptovány prohlížeči v rekordních časech.

5.1.2 WebAssembly

Doposud se v prohlížečích používá JavaScript (v jeho nejnovější specifikaci ECMAScript 2018), který se v prohlížečích interpretuje. Proto rychlost programů, kterou bylo možno v prohlížečích spouštět, je omezena. Spouštět v prohlížečích numericky náročné simulační modely je problematické. Vše se však mění nástupem nového standardu WebAssembly. WebAssembly zavádí binární formát instrukcí, které jsou určeny pro vykonání uvnitř zásobníkového virtuálního stroje [20]. S WebAssembly jsou spojena velká očekávání, protože primární implementace onoho virtuálního stroje byla dosažena ve všech moderních prohlížečích. Dnes je tedy možné psát kód ve vyšších programovacích jazycích, například C/C++/Rust a mít cíl kompilace prohlížeč, kde kód běží rychlostí srovnatelnou s rychlostí kompilovaného kódu v jazyce C.

WebAssembly je formát optimalizovaný ve velikosti a rychlosti načítání, a běží ve stejném "sandboxovaném" prostředí jako JavaScript, které zvyšuje bezpečnost prováděného kódu. Toto tzv. "sandboxované" prostředí má omezený přístup ke zdrojům hostitelského počítače – přístup k disku je typicky omezen na vybrané adresáře, přístup k síti na vybrané servery a porty apod., tak aby nedošlo k proniknutí nebezpečného kódu mimo vymezenou oblast. Sandbox, doslova přeložený jako pískoviště, je vlastně místo, odkud se písek nedostane (nemá dostat) mimo vyhrazenou plochu.

WebAssembly také znemožňuje čtení zdrojového kódu (obfuskace kódu), která vychází z nutnosti přeložit zdrojový kód do binárního formátu. Proto je tedy možné do určité míry zachovat proprietárnost algoritmů, což dosud ve webovém prostředí nebylo možné. Rádi bychom tady podotkli, že to není úplná obfuskace, ale se svými klasickými spustitelnými binárními předchůdci sdílí zranitelnost vůči dekompilaci. S dostatkem času a prostředků je vždy možné použitý algoritmus zpětně reprodukovat jako posloupnost kroků, které musí procesor vykonat a očekává se že se kolem WebAssembly vytvoří produktivní komunita dekompilátorů.

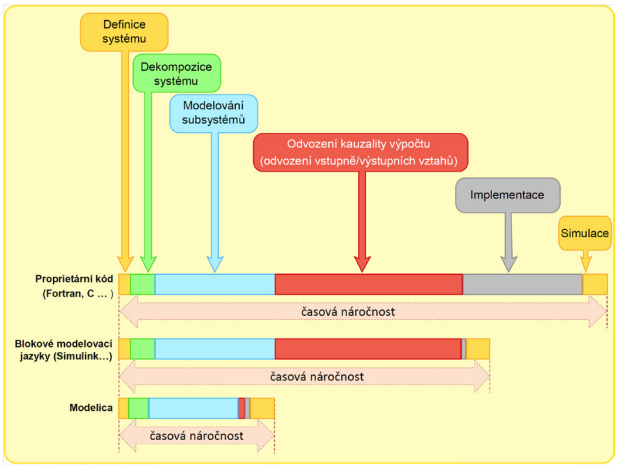
5.2 Modelica – na rovnicích založený modelovací jazyk

Jestliže, obrazně řečeno, kostrou každé výukové aplikace je scénář, svaly výukové simulační aplikace reprezentují interaktivní multimediální komponenty a animace, pak mozkem výukové aplikace je simulační model v pozadí.

Původně se simulační modely programovaly v **klasických programovacích jazycích** (Fortran, C++ apod.).

Počátkem devadesátých let se objevily specializované nástroje pro modelování, využívající výpočetní bloky (sumátory, integrátory aj.), které se počítačovou myší propojují na obrazovce počítače do simulační sítě. Tyto tzv. **blokově orientované simulační jazyky** pracují s propojenými bloky. V propojkách mezi jednotlivými bloky „tečou" signály, které přenášejí hodnoty jednotlivých proměnných od výstupu jednoho bloku ke vstupům dalších bloků. Propojením bloků je možné postupně vytvářet složitější bloky, propojitelné s okolím přes vstupní a výstupní konektory. V blocích dochází ke zpracování vstupních informací na výstupní. Z propojení jednotlivých bloků je pak zřejmé, jakým způsobem se počítají hodnoty jednotlivých proměnných – tj. jaký je algoritmus výpočtu. K nejrozšířenějším blokově orientovaným jazykům patří např. Simulink (*http://www.mathworks.com/products/simulink*) od firmy Mathworks. Dlouhá léta byl Simulink hlavním nástrojem, v němž jsme vytvářeli modely. V Simulinku jsme v minulosti např. vytvořili volně šiřitelnou knihovnu bloků pro modelování fyziologických systémů, která obsahuje též zdrojový kód integrovaného modelu fyziologických systémů, který byl podkladem pro náš výukový simulátor Golem [5]. Výukový simulátor Golem, který jsme vyvíjeli koncem devadesátých let a na přelomu tisíciletí, byl určen k výuce klinické fyziologie poruch homeostázy vnitřního prostředí. Simulátor se využíval na některých našich i zahraničních lékařských fakultách.

Hlavní potíž blokově orientovaných jazyků tkví v tom, že simulační síť složená z hierarchicky propojených bloků zobrazuje grafické vyjádření řetězce transformací vstupních hodnot na výstupní a že při vytváření modelu musíme nadefinovat přesný algoritmus výpočtu, jak ze vstupních hodnot vypočítat výstupní hodnoty modelu. Požadavek pevně zadaného směru spojení od vstupů k výstupům vede k tomu, že propojení bloků odráží postup výpočtu a nikoli



Obrázek 1 – Blokově orientované simulační jazyky přinesly především zkrácení implementace modelu oproti tvorbě modelu v klasických programovacích jazycích. Modelica přinesla další časovou úsporu, protože odvození postupu, jak ze vstupních proměnných modelu vypočítávat výstupní, přenechává kompilátoru.

vlastní strukturu modelované reality. U složitých modelů odvození kauzality výpočtu (tj. odvození algoritmu výpočtu výstupních proměnných ze vstupních proměnných) nebývá jednoduchou záležitostí.

Na přelomu milénia se objevila zcela nová kategorie modelovacích nástrojů, která umožňuje nestarat se o způsob výpočtu a v modelovacích blocích psát přímo rovnice. Byl vytvořen speciální objektově orientovaný jazyk, nazvaný **Modelica**. Modelica, která původně vznikala jako akademický projekt ve spolupráci s malými vývojovými firmami při univerzitách v Lundu a v Linköpingu, se záhy ukázala jako velmi efektivní nástroj pro modelování složitých modelů uplatnitelných zejména ve strojírenství, automobilovém a leteckém průmyslu. Vývoj jazyka Modelica proto postupně získal podporu komerčního sektoru. Rychlost, s jakou se nový simulační jazyk Modelica rozšířil do různých oblastí průmyslu a jak si Modelicu osvojila nejrůznější komerční vývojová prostředí, je ohromující. Dnes existuje několik komerčních i nekomerčních vývojových nástrojů využívajících tento jazyk (viz *https://www.modelica.org*). Propojením jednotlivých komponent v Modelice dochází k propojení soustav rovnic mezi sebou. Propojením komponent tedy nedefinujeme postup výpočtu, ale modelovanou realitu. Způsob řešení rovnic pak „necháváme strojům" [21,22] (Obr. 1). Na rozdíl od blokově orientovaných jazyků, kde struktura propojení hierarchických bloků reprezentuje spíše způsob výpočtu, než modelovanou realitu, struktura modelů v Modelice zobrazuje strukturu modelované reality. Proto jsou i složité modely v Modelice dostatečně průzračné a pochopitelné. To má velký význam právě pro tvorbu složitých integrovaných modelů.

V Modelice jsme vytvořili aplikační knihovnu Physiolibrary pro modelování fyziologických systémů (*http://www.physiolibrary.org*) [23–25]. Tuto knihovnu jsme mimo jiné využili při implementaci rozsáhlého modelu lidské fyziologie HumMod (*http://www.physiomodel.org/*) [26] a dnes Modelicu využíváme jako základní jazyk pro tvorbu simulačních modelů.

5.3 Standardizované rozhraní simulačních modelů: Functional Mock-up Interface

Při praktickém využití modelů vyvstává často problém, jak propojit vytvořený simulační model s ostatními programy, které model využívají. Tento problém zvláště ostře vyvstal v automobilovém průmyslu, kde se často kombinují softwarové systémy různých výrobců. Proto se začalo volat po vzniku nějakého standardizovaného rozhraní mezi modelem a jeho okolím, který by dovolil propojit simulační model s okolím: tj. domluvit se jakým způsobem komunikovat se vstupními a výstupními proměnnými, jak zadávat hodnoty parametrů, jak spouštět model, třeba jen na určitý časový krok apod. Vývoj standardu, nazvaného "funkční maketové rozhraní" – Functional Mock-up Interface (FMI) inicioval Daimler AG s cílem zlepšit výměnu simulačních modelů mezi dodavateli a ostatními výrobci programových systémů. První verze, FMI 1.0, byla vydána v roce 2010 a následovala FMI 2.0 v červenci 2014. K dnešnímu dni pokračuje vývoj standardu prostřednictvím účasti 16 společností a výzkumných ústavů pod střechou Asociace Modelica jako projektu sdružení Modelica. FMI je podporováno více než 100 nástroji a používá se v průmyslu a ve výzkumné a vývojové sféře v celé Evropě, Asii a Severní Americe.

FMI je standard, který podporuje jak komunikaci aplikace s modelem, tak i komunikaci více běžících dynamických modelů mezi sebou – tzv. kosimulaci pomocí kombinace xml souborů a C-kódu.

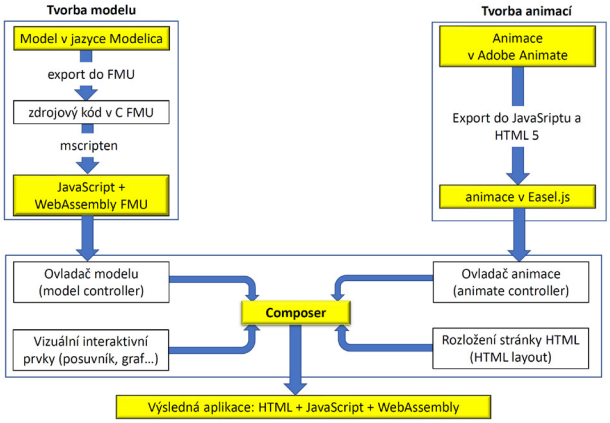
Vývojová prostředí pro jazyk Modelica – Dymola i OpenModelica umožňují model a jeho runtime exportovat podle standardu FMI do tzv. "funkčních maketových jednotek" – Functional Mock-up Unit (FMU) obsahujících zdrojový kód modelu v jazyce C a popisný xml soubor. V naší technologii pak takto vygenerovaný zdrojový text modelu v jazyce C transpilujeme do kódu WebAssembly, což umožňuje, aby simulační model běžel vysokou rychlostí na straně klienta v internetovém prohlížeči.

5.4 Vizualizace

Díky technologii WebAssembly do níž byl díky standardu FMI překompilován simulační model máme k dispozici simulační jádro webové aplikace, běžící v internetovém prohlížeči, a nyní potřebujeme vytvořit způsob komunikace s modelem. Měli bychom být schopni zobrazovat výstupní hodnoty modelu a měnit jeho vstupy, resp. parametry modelu.

JavaSript pro zobrazení výstupů modelu poskytuje velké možnosti. Pro zobrazení grafů existuje celá řada JavaScriptových knihoven. Pro zobrazení hodnot v čase jsme si vybrali opensourcovou grafickou knihovnu plotly.js [27], která umožňuje zobrazení grafů v reálném čase. V budoucnu není problém dopsat podporu i pro jiné grafické knihovny, které možná odstraní výkonnostní problémy, kterými Plotly trpí, když je vyžadována rychlá obnovovací frekvence.

Výstupem modelu ale nemusí být jenom graf. Pro výukové aplikace je výhodné, když model je propojen s animovanými obrázky. Animovaný obrázek je pak řízen hodnotami výstupů modelu na pozadí, stejně tak hodnoty vstupů (parametrů modelu) mohou být zadávány interakcí uživatele s animovanou komponentou.



Obrázek 2 – Naše technologie tvorby webových simulátorů "BodyLight.js"

5.4.1 Adobe Animate

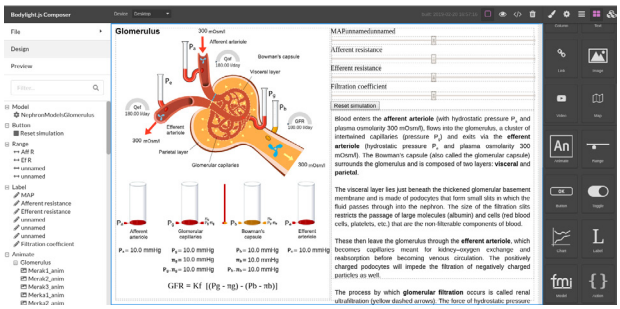
Klíčem k profesionálnímu vhledu výukových simulátorů je zapojení profesionálních výtvarníků do návrhu a tvorby vizualizace vytvářené aplikace. Součástí našeho týmu jsou proto také výtvarníci, kteří ve své práci využívají profesionální nástroje od firmy Adobe. Práci s těmito nástroji také učíme v rámci naší dlouhodobé spolupráce s Vyšší odbornou a Střední uměleckou školou Václava Hollara, kde se podílíme i na výuce oboru "interaktivní grafika".

Jedním z nástrojů pro tvorbu animací je Adobe Animate, který vychází z původního nástroje Adobe Flash. Software Adobe Animate je přední špičkový nástroj pro tvorbu atraktivního interaktivního obsahu pro počítače, smartphony, tablety a televizory, podporuje novější platformy, jako jsou Android™, Apple iOS a Adobe AIR®.

Projekty v Adobe Animate lze také vyexportovat jako JavaScript, s cílem aby se animace chovaly v prohlížečích stejně

jako kdyby to byl normální výstup přehrávače Adobe Flash. K tomu Adobe využívá knihovnu Easel.js [28], která umožňuje zobrazování animací za použití HTML plátna (HTML canvasu).

Naše technologie proto využívá Adobe Animate pro tvorbu interaktivních animací, které propojujeme se vstupy a výstupy modelu. Softwarový nástroj Adobe Animate jsme vybrali proto, že s ním naši grafici umí pracovat, ale v budoucnu není problém rozšířit podporu pro jiné animační knihovny.



Obrázek 3 – Sestavování výukové simulační aplikace v námi vyvinutém softwarovém nástroji Composer.

5.5 Composer – nástroj pro tvorbu finální aplikace

Na tvorbu finální aplikace jsme napsali nástroj nazvaný Composer, který umožňuje vizuální tvorbu webových stránek. Composer je napsaný v JavaScriptovém frameworku React [29] a jako jednostránková aplikace běží v prohlížeči. Composer hojně využívá výsledků opensourcového projektu Grapes.js, který umožňuje jednoduché skládání rozložení (layout) stránky HTML [30].

Composer umožňuje načíst zkompilované WebAssembly FMU a vizualizace z Adobe Animate. Dále obsahuje ovládací prvky pro vstup do modelu; posuvníky, tlačítka a přepínače.

Výstupem z composeru je samostatný HTML soubor, který obsahuje zkompilovaný WebAssembly model a k němu obslužnou logiku výměny dat mezi modelem a animacemi. Dále pak zdrojový kód pro animace a její ovládací logiku.

Návaznost jednotlivých kroků naši technologie zobrazuje obr. 2.

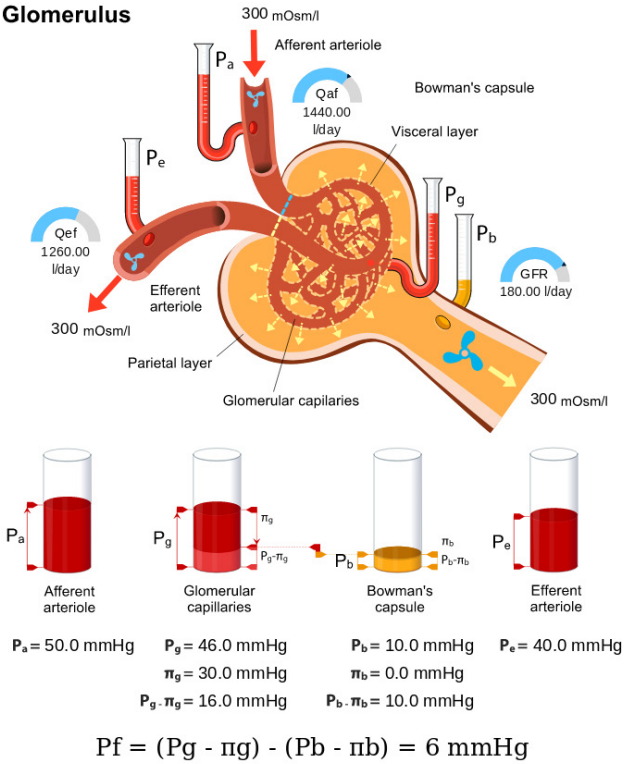
6 Výsledek – výuková aplikace simulace ledvin

Jedním z výsledků uplatnění naší technologie BodyLight.js je výukový simulátor ledvin, který pomocí simulačních her se snaží vysvětlit studentům lékařství základní funkci nefronu [31]. Finální verze výukové webové aplikace je sestavována pomocí nástroje Composer (viz obr. 3).

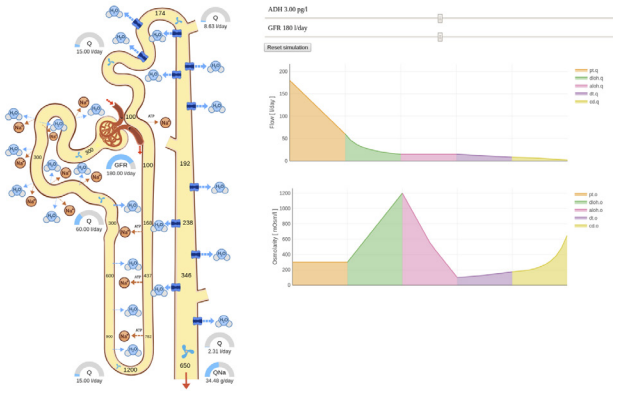
Výukový text (v anglické i české verzi) výsledné aplikace je propojen s grafy a interaktivními obrázky propojenými s modelem na pozadí. Ukázky výstupů zobrazují obrázky 4 a 5.

7 Perspektivy – elektronické učebnice se simulačními hrami

Rychlý rozvoj tabletů, které se začínají využívat i jako médium pro elektronickou distribuci knih a interaktivních výukových materiálů otevírá možnost vytvářet lékařské učebnice zcela nového typu.



Obrázek 4 – Simulátor glomerulu (anglická verze). Simulátor je realizován jako interaktivní obrázek propojený se simulačním modelem v pozadí. Bowmanovo pouzdro (oranžově), aferentní a eferentní arterioly. Červené šipky symbolizují směr proudění krve, žlutá šipka ukazuje směr proudění moči. Pro každou část glomerulusu se vypočítá jak hydrostatický, tak i osmotický tlak. Rozdíl mezi celkovým efektivním tlakem v kapilárách a v Bowmanově pouzdře (vizualizovaný válci) řídí čistý filtrační tok. Uživatel může měnit střední arteriální tlak, rezistence v aferentní a eferentní arteriole a filtrační koeficient. Následně může pozorovat důsledky těchto změn. Ve výukové aplikaci je simulátor doprovázen vysvětlujícím textem.



Obrázek 5 – Jeden z interaktivních obrázků anglické verze webového simulátoru ledvin, zobrazující nefron jako celek – v jednotlivých částech nefronu jsou zobrazeny hodnoty osmolarit a průtoků. V grafu se zobrazuje tok a osmolarita v jednotlivých částech nefronu. Obrázek je propojen s modelem na pozadí. Uživatel může na tomto interaktivním obrázku změnit hladinu ADH a glomerulární filtraci a pozorovat důsledky této změny.

Výukový text může být doprovázen interaktivními animovanými obrázky řízenými podle modelu na pozadí. To dává velké pedagogické možnosti pro vysvětlení složité dynamiky fyziologických procesů.

Jedním z příkladů tohoto přístupu jsou učebnice kardiovaskulární fyziologie a hemodynamiky pro iPad od společnosti PVLoops (*https://harvi.online/site/welcome/*). Do textu jsou začleněny pomocí interaktivních obrázků vyvolávaných tlačítky "nyní si to zkuste". Plná simulace hemodynamiky krevního oběhu je dostupná z libovolného místa v aplikaci prostým otočením iPadu do orientace na šířku. Simulace, spojená s otázkami na konci kapitoly a sadami problémů, umožňuje flexibilní prostředí pro experimentování a objevování metodou "učení praxí" ("learning by doing"), která je z didaktického hlediska velmi efektivní [32].

V budoucnu bychom chtěli prostřednictvím dalšího rozvoje naší technologie vytvářet obdobné výukové aplikace – přitom však vystačíme s prohlížečem a nemusíme celou aplikaci implementovat pro každou platformu.

Poděkování

Vývoj lékařských simulátorů je podporován grantem TRIO MPO FV20628 a FV30195

Literatura

[1.] Comenius JA. Schola ludus seu Encyclopaedea Viva. Sarospartak; 1656.

[2.] Kofránek J, Matoušek S, Rusz J, Stodulka P, Privitzer P, Mateják M, et al. The Atlas of Physiology and Pathophysiology: Web-based multimedia enabled interactive simulations. Comput Methods Programs Biomed. 2011;104: 143–153.

[3.] Kofránek J, Andrlík M, Kripner T, Mašek J, Stodůlka P. "Od umění k průmyslu" – propojení technologií při tvorbě lékařských výukových programů. Medsoft. 2003;15: 43–56.

[4.] Kofránek J, Kripner T, Andrlík M, Mašek J. Creative connection between multimedia, simulation and software development tools in the design and development of biomedical educational simulators. Proceedings of Simulation Interoperability Workshop, Orlando 2003, Position papers, Volume II. SISo Inc.; 2003. pp. 677–687.

[5.] Kofránek J, Vu LDA, Snaselova H, Kerekes R, Velan T. GOLEM-multimedia simulator for medical education. Stud Health Technol Inform. IOS Press; 1999; 2001; 1042–1046.

[6.] Kofránek J, Andrlík M, Kripner T, Mašek J, Velan T. Simulation chips for GOLEM – multimedia simulator of physiological functions. Simulation in Health and Medical Sciences. Society for Computer Simulation International, Simulation Councils, San Diego; 2002. pp. 159–163.

[7.] Kofránek J, Tribula M. Control web pro multimediální interaktivní ledvinu. Medsoft. 2007;19: 93–102.

[8.] Andrlík M, Kofránek J, Matoušek S, Stodulka P, Wünsch Z, Kripner T, et al. Internetový atlas výukových multimediálních modelů pro vybrané kapitoly normální a patologické fyziologie člověka. Ukázka předběžných výsledků. Medsoft. 2006;18: 7–12.

[9.] Wünsch Z, Kripner T, Kofránek J, Uk LF. Mechanické vlastnosti kosterního svalu-výukový program. Medsoft. 2004;16: 175–183.

[10.] Wünsch Z, Kripner T, Kofránek J. Realizace výukového programu mechanické vlastnosti kosterního svalu. Medsoft. 2005;17: 213–218.

[11.] Kofránek J, Matoušek S, Andrlík M. Škola (simulační) hrou – využití simulačních modelů acidobazické rovnováhy v e-learnignové aplikaci. Medsoft. 2007;19: 83–92.

[12.] Stodulka P, Privitzer P, Kofránek J, Mašek J. Nové postupy v tvorbě simulátorů-inteligentní propojení Matlabu a Simulinku s platformou .NET a tvorba stavových automatů řídících výslednou aplikaci. Medsoft. 2006;18: 177–184.

[13.] Kofránek J, Privitzer P, Stodulka P. Technologie a trendy tvorby výukových simulátorů. Medsoft. 2008;20: 37–56.

[14.] Kofránek J, Tribula M, Privitzer P. Modely cirkulace a přenosu krevních plynů pro lékařskou výuku. Medsoft. 2018;30: 71–98.

[15.] Wünsch Z, Matúš M, Kripner T, Kofránek J. Modely regulace ve fyziologickém praktiku. Medsoft. 2006;18: 213–218.

[16.] Kofránek J. Webové simulátory. Medsoft. 2010;22: 81–95.

[17.] Privitzer P, Šilar J, Tribula M, Kofránek J. Od modelu k simulátoru v internetovém prohlížeči. Medsoft. 2010;22: 149–169.

[18.] Kofránek J, Mateják M, Ježek F, Privitzer P, Šilar J. Výukový webový simulátor krevního oběhu. Medsoft. 2011;23: 106–121.

[19.] Tribula M, Ježek F, Privitzer P, Kofránek J, Kolman J. Webový výukový simulátor krevního oběhu. Medsoft. 2013;25: 197–204.

[20.] WebAssembly [Internet]. [cited 1 Mar 2019]. Available: https://webassembly.org/

[21.] Kofránek J, Mateják M, Privitzer P, Tribula M. Causal or acausal modeling: labour for humans or labour for machines. Technical computing Prague 2008: 16th annual conference proceedings. Humusoft; 2008. pp. 124–140.

[22.] Kofránek J. Modelica. Medsoft. 2013;25: 64–114.

[23.] Mateják M, Kulhánek T, Šilar J, Privitzer P, Ježek F, Kofránek J. Physiolibrary-Modelica library for physiology. Proceedings of the 10 th International Modelica Conference; March 10-12; 2014; Lund; Sweden. Linköping University Electronic Press; 2014. pp. 499–505.

[24.] Mateják M, Ježek F, Tribula M, Kofránek J. Physiolibrary 2.3-An Intuitive Tool for Integrative Physiology. IFAC-PapersOnLine. Elsevier; 2015;48: 699–700.

[25.] Mateják M. Physiolibrary – fyziológia v Modelice. Medsoft. 2014;26: 165–172.

[26.] Mateják M, Kofránek J. Physiomodel – an integrative physiology in Modelica. 2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). 2015. pp. 1464–1467.

[27.] Plotly.js [Internet]. [cited 1 Mar 2019]. Available: https://plot.ly/javascript/

[28.] EaselJS [Internet]. [cited 1 Mar 2019]. Available: https://www.createjs.com/easeljs

[29.] React [Internet]. [cited 1 Mar 2019]. Available: https://reactjs.org/

[30.] GrapesJS [Internet]. [cited 1 Mar 2019]. Available: https://grapesjs.com/

[31.] Šilar J, Ježek F, Mládek A, Polák D, Kofránek J. Model visualization for e-learning, Kidney simulator for medical students. Proceedings of the 13th International Modelica Conference, Regensburg, Germany, March 4--6, 2019. Linköping University Electronic Press; 2019. pp. 393–402.

[32.] Leisman S, Burkhoff D. Use of an iPad App to simulate pressure-volume loops and cardiovascular physiology. Adv Physiol Educ. Am Physiological Soc; 2017;41: 415–424.

Kontakt

David Polák
e-mail: *david.polak@lf1.cuni.cz*

Filip Ježek
e-mail: *jezekf@gmail.com*

Jan Šilar
e-mail: *jansilar@post.cz*

Jiří Kofránek
e-mail: *kofranek@gmail.co*m
Oddělení biokybernetiky
a počítačové podpory výuky
ÚPF 1. LF UK
Praha U Nemocnice 5 128 53
Praha 2