3D MEDICAL DATA VISUALIZATION IN WEB BROWSER

Tomáš Sychra
Master Degree Programme (2), FIT BUT
E-mail: xsychr03@stud.fit.vutbr.cz

Supervised by: Michal Španěl
E-mail: spanel@fit.vutbr.cz

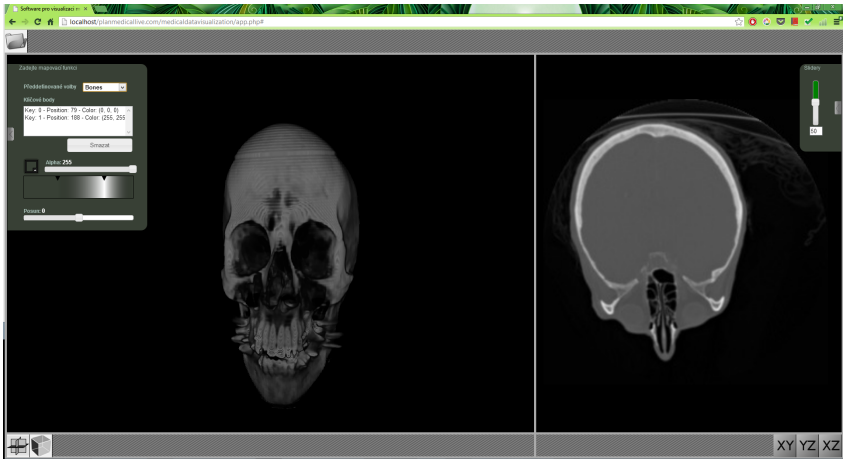
Abstract: The paper discusses possibilities of accelerated 3D visualization of medical data in the Web browser. It deals with WebGL standard and its use in real applications in greater detail. The work analyzes a group of techniques referred to as a volume rendering and implementation of the volume rendering with WebGL standard. This work also provides design of a real and complex application for visualization of volumetric data.

Keywords: 3D graphics, web browser, accelerated graphics, WebGL, medical data, volumetric data, volume rendering, Volume ray casting, data visualization

1 ÚVOD

V oblasti internetových technologií dochází k velkým změnám. Vysoká rychlost internetu současně se vzrůstajícím výkonem osobních zařízení a optimalizací interpretace JavaScriptu vytváří ideální zázemí pro vznik aplikací nativně běžících ve webovém prohlížeči. Jedním z posledních problémů bránících tvorbě náročných programů byla chybějící nativní akcelerace pokročilé 3D grafiky na grafické kartě.

V této práci se věnuji vytvoření komplexní aplikace pro vizualizaci medicínských dat, která bude fungovat v okně webového prohlížeče s využitím stále poměrně nové technologie WebGL. Ač mají současné technologie jistá omezení, lze vytvořit funkční a reálně použitelnou aplikaci.



Obrázek 1: Současný stav aplikace

2 NÁVRH A REALIZACE

Cílová aplikace bude podporovat multiplanární zobrazení, zobrazení jednotlivých řezů a také volume rendering (pomocí algoritmu volume ray casting). Nejvýznamnějším omezením je chybějící podpora

3D textur ve WebGL. S tím se lze do jisté míry vyrovnat, ačkoli výsledné zobrazení není v tak vysoké kvalitě - to ale neznamená, že neexistuje způsob jak vyšší kvality dosáhnout. V neposlední řadě je kladen velký důraz na co nejjednodušší uživatelské rozhraní a minimalizace nepotřebných prvků, které pouze znepřehledňují aplikaci a ubírají místo samotné vizualizaci dat.

2.1 Řešení základních komponent aplikace

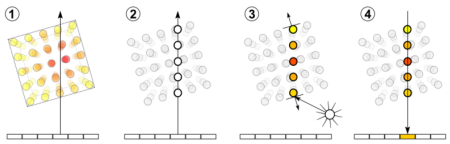
Správa účtů je řešena pomocí Google účtu. Automaticky tak získáme přístup na Google Drive, kde může mít uživatel uložena svá data. Výhodné je také ověření přes OAuth 2.0 a zabezpečený přenos dat z Google Drive ke klientovi (SSL + omezená doba planosti odkazu na soubor), jelikož aplikace pracuje s medicínskými daty, která jsou velmi citlivá.

Uživatelské rozhraní je navrženo co nejjednodušeji. Jádrem je rozdělená obrazovka na dvě poloviny. V jedné lze přepínat mezi multiplanárním zobrazením a volume renderingem. V druhé lze volit mezi sagitálním, axiálním nebo koronárním zobrazením. Velikost jednotlivých částí rozdělené obrazovky lze interaktivně měnit. Realizace návrhu je na obr. 1.

Práce s daty je řešena pomocí File API (načítání z lokálního úložiště) a Google Drive API (načítání z Google Drive).

2.2 Volume rendering

Srdcem celé aplikace je vizualizace dat pomocí volume renderingu. Aplikace konkrétně používá algoritmus volume ray casting. Jedná se o "image order"algoritmus, což znamená, že jsou postupně počítány hodnoty výsledného obrazu (pixel po pixelu). Princip je následovný (lze sledovat na obr. 2). Z kamery jsou vyslány paprsky do scény skrz zobrazovací rovinu (skrz každý pixel). Sledujeme pohyb každého paprsku skrz objekt, následně hodnoty jednotlivých voxelů sčítáme (popřípadě násobíme) a aplikujeme na ně mapovací funkci. Výslednou hodnotu pak přiřadíme příslušnému pixelu, skrze který jsme paprsek vyslali. [1, 3]



Obrázek 2: Základní kroky Volume ray castingu [1]

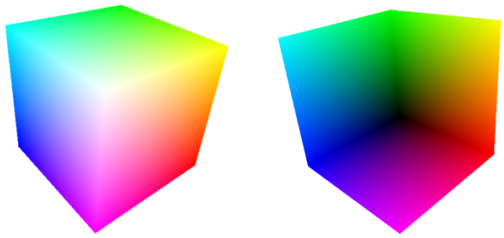
2.3 Realizace

Bez použití 3D textur se situace poměrně komplikuje, jelikož nemáme jak dostat data do paměti graf. karty tak, abychom skrze ně mohli prostorově procházet a akumulovat informace o intenzitě jednotlivých voxelů objemových dat. Řešení ale existuje. Data do shaderu musíme dostat jinou formou. Ideální je využít fragment shader a data do něj předat jako velkou texturu (vize textury na obr. 3). Nevýhodou je snížení kvality výsledného obrazu, jelikož velikost textury, jenž lze do fragment shaderu předat, je v dnešní době nejčastěji omezena na 4096x4096px. Rozměry textury a počet řezů je třeba do shaderu předat také. Je samozřejmě možné použít více textur, ale je to velmi výpočetně i paměťově náročné. Pro vykreslení je použit následující postup. [2]

[1] `http://en.wikipedia.org/wiki/File:Volume_ray_casting.png`

- Vytvoření obalového tělesa (bounding box) a jeho obarvení jako je na obr. 4. Barvy přesně určují souřadnice na krychli. Ty korespondují se souřadnicemi v objemových datech. Výhodou je, že i při transformacích zůstávají souřadnice krychle a objemových dat konzistentní.
- Vykreslení zadních a následně předních stran krychle s identickým nastavením scény. Při odečtení překrývajících se barev jež jsou viditelné z kamery je získán vektor, který určuje, jak přesně se paprsek vyslaný z kamery skrz zobrazovací rovinu v objemovém tělese pohybuje.
- Pohyb v objemovém tělese (v našem případě simulovaný 2D texturou) probíhá obdobně jako u použití 3D textur.

Obrázek 3: Reprezentace objemových dat velkou 2D texturou



Obrázek 4: Levo: přední stěny krychle; Pravo: zadní stěny krychle [2]

3 ZÁVĚR

Navržená aplikace již funguje, ale zatím pouze v rámci raného prototypu. Výsledná kvalita volume renderingu je dostatečná, ale dále pracuji na jejím zlepšení. Ostatní funkce, jako multiplanární zobrazení, koronální, axiální a sagitální řez, se renderují rychle a spolehlivě. Vygenerování snímku v multiplanárním zobrazení se podařilo snížit pod 20ms při rozměrech dat 512x512x512 pixelů. Aplikace je vyvíjena na notebooku s následujícími parametry: grafická karta - ATI Mobility Radeon HD5730 s 1 GB vlastní paměti (renderuje objemová data); procesor - Intel Core i7 720QM s frekvencí 1,6 GHz (generuje textury pro multiplanární zobrazení); paměť RAM - 8GB. Program funguje bez problému na běžném počítači s průměrným hardwarem a minimálně 4GB RAM (doporučená 8GB). Aplikace je primárně vyvíjena pro Google Chrome, jelikož má nejrychlejší interpretaci JavaScriptu. Funguje i v prohlížeči Mozilla Firefox.

PODĚKOVÁNÍ

Tento příspěvek vznikl za podpory grantu FIT-S-11-2 a výzkumného záměru MSM 0021630528.

REFERENCE

[1] Pawasauskas, John. "Volume visualization with ray casting."CS563—Advanced Topics in Computer Graphics, Feb 18 (1997): 15.

[2] Interactive visualization of volumetric data with WebGL in real-time. CONGOTE, John, Alvaro SEGURA, Aitor MORENO, Jorge POSADA a Oscar RUIZ. *Proceedings of the 16th International Conference on 3D Web Technology* [online]. 2011. vyd. Paris, France: ACM, 2011, 137 - 146 [cit. 2013-03-19]. ISBN 978-1-4503-0774-1.

[3] FERNANDO, R. GPU gems. Vyd. 1. Boston: Addison-Wesley, 2004, 765 s. ISBN 03-212-2832-4. Dostupné z: `http://http.developer.nvidia.com/GPUGems/gpugems_pref01.html`