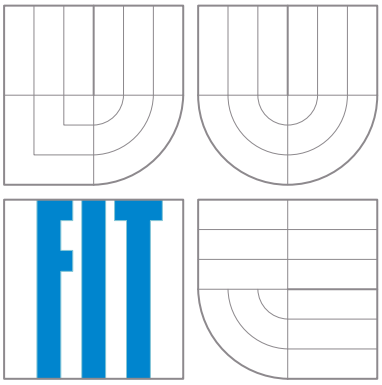


Grafická a zvuková rozhraní a normy

Aktuální vývoj WebGL

27. listopadu 2014

Autor: Pavel Macenauer, xmacen02@stud.fit.vutbr.cz
Jan Bureš, xbures19@stud.fit.vutbr.cz

Fakulta Informačních Technologií
Vysoké Učení Technické v Brně

Charakteristika

WebGL je multiplatformní volně dostupná knihovna, která se využívá pro graficky náročné aplikace na webových stránkách. Díky využití grafického výkonu počítače jsou i tyto aplikace celkem svižné. Pro zobrazení stránek s WebGL stačí uživateli pouze webový prohlížeč, který WebGL podporuje (v současnosti všechny hlavní prohlížeče) a na rozdíl od jiných technologií nemusí instalovat žádné další podpůrné nástroje. Specifikaci WebGL si vzalo na starost konsorcium Khronos Group (členové pracovní skupiny, která se zabývá vývojem standardu WebGL jsou například Google, Mozzila, Opera, Apple a další).

Historický přehled

2006 Vladimír Vukićević vytvořil první 3D prototyp pomocí elementu canvas [2].

2007 Mozzila Firefox a Opera podporují WebGL [2].

2009 Khronos group se zabývá vytvořením oficiální specifikace WebGL.

2011 První oficialní specifikace WebGL 1.0 (OpenGL ES 2.0)[2].

2013 Specifikace WebGL 2.0 (OpenGL ES 3.0) [3].

Vlastnosti

WebGL je nízko úrovňové API, které vychází ze standardu OpenGL ES (určeno pro vestavěné zařízení). Kvůli tomuto je i pro jednoduchou aplikaci potřeba napsat mnoho řádků kódu. Díky mnoha existujícím knihovnám však tento problém částečně odpadá. Přímý přístup k hardwaru také představuje určitá bezpečnostní rizika. V současnosti jsou dostupné dvě verze WebGL a to WebGL 1.0.2, která vychází ze standardu OpenGL ES 2.0 a WebGL 2.0, která vychází ze standardu OpenGL ES 3.0. Pomocí WebGL je možné programovat shadery jazykem GLSL. Na webové stránce je pak aplikace WebGL zobrazena pomocí elementu canvas (je integrováno s DOM) [4].

Výhody

- Velice rozšířené, mnoho knihoven
- Funguje téměř na všech zařízeních s internetovým prohlížečem
- Založeno na standardu OpenGL ES
- Nízkoúrovňové API (rychlost, plně v rukou programátora)
- Není potřeba dalších vývojových nástrojů

Nevýhody

- Nízkoúrovňové API (i pro jednoduchou aplikaci nutno využít mnoho instrukcí)
- Bezpečnostní rizika

Zajímavé projekty

Existuje mnoho ukázkových aplikací či převedených aplikací do technologie WebGL (budou uvedeny níže). Z těch reálně využívaných stojí za zmínku Google Maps [1] a ZygoteBody [5]. Dále spousty příkladů je k nalezení i na Chrome Experiments [6], kde jsou spíše menší projekty a pro počítačové grafiky může být zajímavá galerie shaderů Shadertoy [7].

Budoucnost

- Podpora OpenGL ES 3.1 a výpočetních shaderů [8].
- Dostupnost WebGL ve všech mobilních prohlížečích [8].

Bezpečnost WebGL a podpora IE

V roce 2011 Microsoft oznámil, že WebGL nebude podporovat [26]. Tvrdil, že samotné WebGL je nebezpečné jen svým návrhem. Důvody můžou být následující

- **Undefined behaviour** – např. čtení pixelů mimo framebuffer
- **Přístup do nealokované paměti**
- **Shadery** – shadery musí být před předáním grafické kartě validovány
- **DoS** – scéna trvá velkou dobu vykreslit, počítač tak může přestat reagovat
- **Čtení obrázků a videí z cizích zdrojů** – dle specifikace umožněno pouze, pokud je zdroj validován přes CORS ([14])

Výše zmíněné jsou příklady bezpečnostních rizik, které můžou nastat při implementaci v jednotlivých prohlížečích a počítač uživatele tak může být napaden, typicky přestane reagovat [15].

Asi nejznámějším bezpečnostním problémem WebGL se stalo vykradení grafické paměti uživatele [21] a tím pádem pořízení screenshotů citlivých dat.

S IE11 Microsoft již oznámil, že WebGL podporovat bude [19]. Pravděpodobně pod tlakem konkurence (Firefox, Chrome, Opera), která již technologii dávno podporovala. WebGL v IE11 běží zapouzdřené pod DirectX, což umožňuje v případě, že nastane některá z výše zmíněných bezpečnostních děr, obnovit systém uživatele, v případě, že vypadne.

Alternativy

Existují některé projekty (Unity3D, SilverLight 3D, ...), které je možné po doinstalování pluginů a jiných knihoven umístit i na webovou stránku, většinu z nich však není možné zprovoznit na mobilních zařízeních. Pro vykreslení grafiky na webu je možné využít tyto technologie:

DOM 2D objekty lze ukládat přímo do DOM struktury webové stránky. Využívá se k tomu HTML5 elementů jako **rect, circle, svg, path**, které jsou ukládány přímo do DOM-u. Při malém množství objektů se jedná o nejrychlejší řešení. Při vyšším se však už zahlcuje DOM a jsou lepší jiná řešení.

Canvas Další možností pro 2D je využít HTML5 elementu canvas a jeho API. Oproti DOM-u, který se dá považovat za retained mode, se jedná o immediate mode kreslení, kdy nic není ukládáno a vše se rovnou vykresluje pomocí javascriptových metod (např. arc(), fill(), lineTo(), stroke() atd.). Výhodou je, že se DOM nepřehltí v případě velkého množství objektů. Na druhou stranu vše je vykreslováno v jednom elementu canvas a je tak těžší na jednotlivé objekty vázat nějaké handlery.

Srovnání WebGL s Canvas2D a DOM

Výhody WebGL

- rychlost u vyššího počtu objektů (10-100x)
- více možností - stínovaný, světla, zoom, ...
- hardware akcelerace pro 3D

Nevýhody WebGL

- programovací náročnost - nehodí se na jednoduché aplikace
- horší kompatibilita, dnes už pouze Android Browser dělá problémy

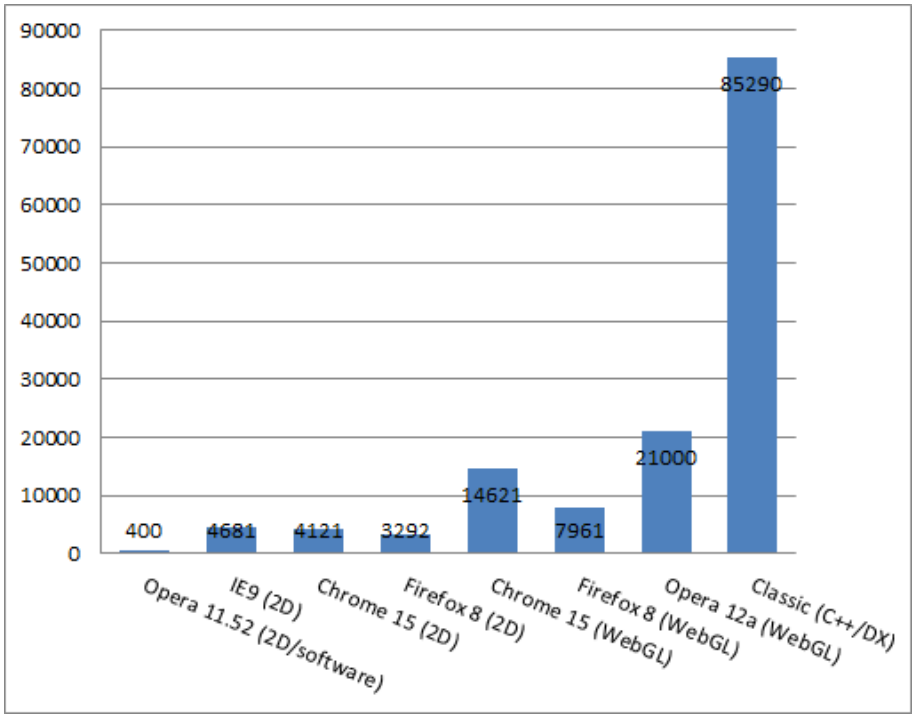
Herní enginy

Jednou z nejpopulárnějších oblastí, zažívající svůj růst v posledních letech kolem WebGL, jsou herní enginy, technologie a různé další frameworky. Výhodou webu jako platformy pro počítačové hry je, že je závislá pouze na internetovém prohlížeči. Nevyžaduje žádné speciální runtimy, není závislá na operačním systému a není třeba nic instalovat. Stačí pouze otevřít prohlížeč, napsat url adresu a hrát. Jedná se tak o řešení vhodná především pro výpočetně jednodušší úkoly a tedy hlavně 2D hry.

Přehled zajímavých knihoven a enginů

Proč používat jinou knihovnu a ne rovnou WebGL? Programování ve WebGL je srovnatelné s OpenGL, kdy na vykreslení jednoduché scény s pár krychlemi je třeba stovky řádků kódu - sypete alokujete buffery, sypete do nich bod po bodu, přiřazujete barvu, textury - zkrátka stáráte se o vše a následně. Externí knihovny jako je třeba Three.js umožňují tuto práci výrazně zjednodušit.

Na trhu dnes existuje obrovské množství různých knihoven, především pro tvorbu 2D grafiky. Vybrány jsou perspektivní projekty, které bychom doporučili používat.



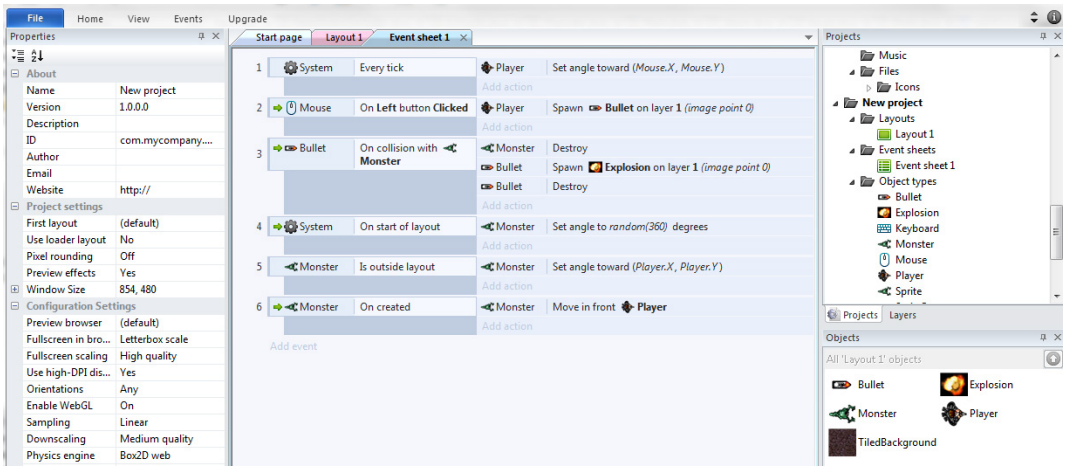
Obrázek 1: Srovnání vykreslení počtu sprajtů při 30 FPS (Game Engine Construct2) z roku 2011

O3D

- Engine pro 3D aplikace
- Usnadňuje práci s grafem scény, texturami, shadery, ...
- Vytvořen v roce 2009 firmou Google (nutné doinstalovat plugin)
- V roce 2010 přepsán jako Javascriptová vrstva nad WebGL
- Od roku 2011 se nevyužívá (je využívána spíše knihovna Three.js)

Construct 2 [9]

- komerční pro komerční účely, jinak zdarma
- 2D herní editor
- exportovatelné na platformy: HTML5, Chrome, Facebook, Windows Phone nebo přes wrappery jako CocoonJS na Android či iOS
- časově nenáročný vývoj vhodný pro prototypování nebo neprogramátory (není třeba prog. znalostí)
- jednoduché hry (Space Invaders, Pac-Man, Tetris, ...)



Obrázek 2: Uživatelské rozhraní Construct 2

Při tvorbě aplikace v editoru vytváříte objekty a těm přiřazujete podmínky a následné akce. Nejčastěji se jedná o sprajty, tedy 2D obrázky, ale jako objekty se chovají i vstupní zařízení jako myš nebo klávesnice nebo formulářové prvky. Editor umí pracovata i s AJAXem nebo WebSockets, umožňuje tedy i multiplayer hry a používání databází.

Three.js [10]

- zdarma (MIT License)
- 3D knihovna
- Využívá především WebGL, ale pro kompatibilitu umožňuje i Canvas/SVG fallback
- Obrovské množství 3D funkcí: materiály, osvětlení, animace, kamera (ortografická, perspektivní), přístup k GLSL, geometrie (plocha, krychle, 3D text, ...), načítání dat (binární, obrázky, přes JSON, ...), matematické funkce a mnohé další

pixi.js [11]

- zdarma (MIT License)
- 2D knihovna
- Scene Graph
- WebGL filtry, blending, tinting a další postprocessing efekty
- Podpora více dotyků – optimalizováno pro mobilní zařízení
- Detekce rendereru – dle podpory zvolí vykreslování pomoci WebGL nebo Canvas2D
- Používají i společnosti jako Google nebo McDonalds

Isogenic [12]

- komerční
- client-server herní engine specializovaný na multiplayer a masivně hrané multiplayer hry
- využívá Node.js, MongoDB, není však náhradou pro HTTP server
- mocná síťová komunikace umožnující na jednotlivé klienty streamovat události nebo i celé grafické objekty
- jednoduché kreslící funkce (DOM-based nebo Canvas based), nevyužívá WebGL, ale to lze zprostředkovat pomocí jiných knihoven

Na závěr ke knihovnám

Každou z předchozích knihoven jsme vyhodnotili jako nejvhodnější pro to co dělá. Pokud máte skvělý nápad a neumíte programovat nebo chcete vytvořit hru za odpoledne, doporučili bychom Construct2. Pro 2D je pak pixi.js, pro 3D naopak three.js. Jako poslední je zmíněn i Isogenic, což je jeden z nejperspektivnějších projektů pro masivně hrané multiplayer hry a lze ho kombinovat i s výše zmíněnými knihovnami.

Další WebGL herní enginy

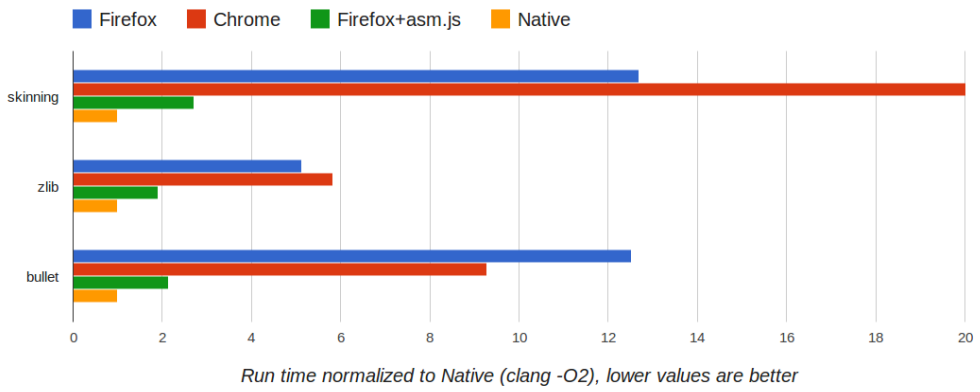
- enchant.js (`http://enchantjs.com/`) – objektově orientovaný 2D/3D engine japonského původu
- PlayCanvas (`https://playcanvas.com/`) – 3D engine s editorem, cloud-server a velká komunita, pro nekomerční účely zdarma
- Phaser (`http://phaser.io/`) – 2D herní engine
- voxel.js (`http://voxeljs.com/`) – pro milovníky krychlí

C/C++ - Emscripten - JavaScript

Za zmínku stojí zmínit i Emscriptem [24]. Jedná se o compiler LLVM kódu do JavaScriptu, především dělaný pro C/C++. Ke kompilaci do LLVM používá Clang, následně LLVM převádí do asm.js nebo JavaScriptu.

Asm.js [13] je programovací jazyk tvořený podmnožinou JavaScriptu, přizpůsobený pro výkon ostatních jazyků na webu. Sám o sobě není výkonnější než-li samotný JavaScript – jedná se o jeho podmnožinu, ale při překladu ostatních jazyků s LLVM výstupem do JavaScriptu vykazují tyto jazyky výkonnostní nárůst. Jedná se o projekt Mozilly, Firefox tak byl prvním prohlížečem, který optimalizace asm.js implementoval. Google Chrome je také podporuje.

Emscriptem je optimalizován pro překlad OpenGL na WebGL a dobře podporuje i SDL, lze tak převést velké množství aplikací původně napsaných v OpenGL na kód běžící v prohlížeči.



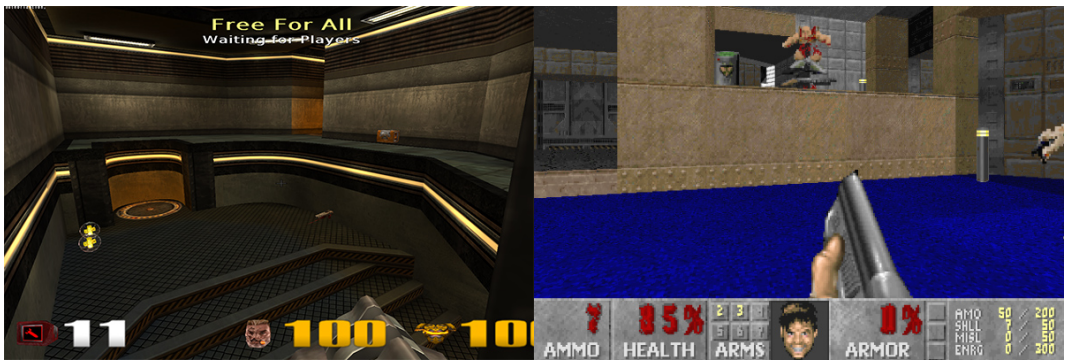
Obrázek 3: Srovnání rychlosti nativního kódu a přeloženého kódu

Zajímavé Emscripten projekty

Pro nás nejzajímavější je pravděpodobně fakt, že byly již přeloženy projekty založené na herních frameworcích - Unity [20] a Unreal. Součástí Unity5 má být i podpora pro web, která bude pravděpodobně řešena právě tímto přístupem.

- DeadTrigger 2 (http://beta.unity3d.com/jonas/DT2/) - přeložený projekt z Unity
- Quake 3 (http://www.quakejs.com/)
- Doom (http://kripken.github.io/boon/boon.html)

Byly portnuty i aplikace založené na Qt nebo i jiných programovacích jazycích [23]: Ruby, Python, Lua, Perl.



Obrázek 4: Příklady portnutých projektů pomocí Emscripten - Quake 3 Arena, Doom

Literatura

[1] http://maps.google.com/.

[2] http://www.students.science.uu.nl/~3685632/content/history.html.

[3] http://learningwebgl.com/blog/?p=5933.

[4] https://www.khronos.org/webgl/wiki/Getting_Started.

[5] http://www.zygotebody.com/.

[6] http://www.chromeexperiments.com/webgl/.

[7] https://www.shadertoy.com/.

[8] https://docs.google.com/presentation/d/1-iML22EjbtlfzRKR5aoFZYr7XvtZKjwESl4D9GADnag/preview#slide=id.g3ed3afcb1_0137.

[9] https://www.scirra.com/manual/1/construct-2.

[10] http://threejs.org/docs/.

[11] https://github.com/GoodBoyDigital/pixi.js.

[12] http://www.isogenicengine.com/docs-manual.html.

[13] http://asmjs.org/.

[14] http://www.w3.org/TR/cors/.

[15] https://www.khronos.org/webgl/security/.

[16] http://books.google.cz/books?id=ud_fAAAAQBAJ&lpg=PA21&ots=M_WgIbNpsh&dq=dom%20vs%20webgl%20vs%20canvas&hl=cs&pg=PA21#v=onepage&q=dom%20vs%20webgl%20vs%20canvas&f=false.

[17] http://msdn.microsoft.com/en-us/library/ie/dn265058%28v=vs.85%29.aspx.

[18] http://html5gameengine.com/tag/webgl.

[19] http://www.winbeta.org/news/webgl-returns-internet-explorer-11-thanks-security-impr

[20] http://blogs.unity3d.com/2014/10/07/benchmarking-unity-performance-in-webgl/.

[21] https://blog.mozilla.org/security/2011/06/16/webgl-graphics-memory-stealing-issue/.

[22] https://www.scirra.com/blog/58/html5-2d-gaming-performance-analysis.

[23] https://github.com/kripken/emscripten/wiki/Porting-Examples-and-Demos.

[24] http://kripken.github.io/mloc_emscripten_talk/#/.

[25] https://developer.tizen.org/dev-guide/2.2.1/org.tizen.web.appprogramming/html/guide/w3c_guide/graphics_guide/performance_comparison.htm.

[26] http://arstechnica.com/information-technology/2011/06/microsoft-no-way-to-support-webgl-and-meet-our-security-needs/.