Středoškolská odborná činnost

Izometrické zobrazení v HTML5

Antonín Vlček

Brno 2014

Středoškolská odborná činnost

Obor SOČ: 18. Informatika

Izometrické zobrazení v HTML5

Autor: Antonín Vlček

Škola: Gymnázium Brno-Řečkovice

Konzultant: Ing. Ondřej Žižka

Mgr. Jan Herman

Brno 2014

Prohlášení

Prohlašuji, že jsem svou práci vypracoval(a) samostatně, použil(a) jsem pouze podklady (literaturu, SW, atd.) citované v práci a uvedené v přiloženém seznamu a postup při zpracování práce je v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) v plném znění.

V Brně dne 24. 2. 2014 podpis: .....

Poděkování

Děkuji panu Ing. Ondřeji Žižkovi a panu Mgr. Janu Hermanovi za obětavou pomoc o podnětné připomínky, které mi během práce poskytli. Dále bych rád poděkoval Jihomoravskému centru pro mezinárodní mobilitu (JCMM), za finanční podporu mé práce.

Anotace

Tato práce se věnuje problematice vytvoření JavaScriptového enginu pro izometrické zobrazení. V první části práce je stručně objasněn princip izometrické projekce a jsou zde popsány technologie, pomocí kterých je možné tohoto cíle dosáhnout. Další část je věnována popisu vývoje enginu a ukázce jeho využití v praxi na příkladu známé hry Snake. Do příloh je umístěna podrobná dokumentace všech metod. Přínosem by tato práce měla být především pro začínající vývojáře her, kteří se chtějí seznámit s problematikou izometrie.

Klíčová slova

Izometrické zobrazení, Canvas, JavaScript, HTML5, Dlaždicové hry

Annotation

This thesis is dedicated to the topic of isometric view JavaScript engine creation. Its first part consists of a brief explanation of the isometric projection basics and a decription of technologies used to accomplish this task. The following part describes the engine development process and provides a demonstration of its utilization for creation of the well-known Snake game. The addenda consist of a detailed documentation of the all methods used in this thesis. The work can be helpful to beginners in game development who intend to get acquainted with the topic of isometry.

Keywords

Isometric view, Canvas, JavaScript, HTML5, Tile based games

Obsah

Seznam pojmů

DOM

Z anglického Document Object Model. Je to objektově orientovaná reprezentace HTML nebo XML dokumentu.

Sprity

Malé obrázek ze kterých se poskládá celá scéna. Tato technika se využívá z důvodu menší náročnosti na výkon, než vykreslování jednoho velkého obrázku.

JSON

JavaScript Object Notation, je způsob zápisu dat nezávislý na počítačové platformě.

Web Workers

Technologie HTML5, která umožňuje provádění početních procesů na pozadí, mimo hlavní vlákno programu.

Web Sokets

Technologie HTML5 umožňující komunikaci se serverem bez využití dalšího programovacího jazyka, jako je například PHP.

Framework

Framework je softwarová struktura, která slouží jako podpora při programování a vývoji a organizaci jiných softwarových projektů.

API

API označuje v informatice rozhraní pro programování aplikací.

Open-source

Otevřený software, s volně dostupným zdrojovým kódem.

LicenceApache 2.0

Licence pro open source. Podrobnosti a úplné znění nalezenete na oficiálním webu `http://www.apache.org/licenses/LICENSE-2.0`

Úvod

V dnešní době dochází k prudkému rozvoji moderních webových technologií, jako je například HTML5 a CSS3. Pomocí těchto nových prostředků je možné realizovat mnoho projektů s výrazně vyšší efektivitou, a dosáhnout tak výsledků, které bychom si ani nedokázali představit ještě před pár lety, kdy byl internet pouhým prostředkem pro sdělování vesměs textových informací a jednotlivé stránky obsahovaly pouze nezformátovaný text, pár hypertextových odkazů a občas obrázek.

Od této podoby urazil internet velmi dlouhou cestu a i počet uživatelů této počítačové sítě rapidně vzrostl. Vznikla tedy poptávka po nových technologiích, díky kterým bude možné vykreslovat vlastní grafiku. Dostupnost těchto prostředků vytvořila prostor pro vývoj pokročilejších webových aplikací, jako jsou například hry nebo jiné grafické vizualizace, které se snaží interpretovat data tak, že nahradí dlouhé texty grafickými animacemi a podobně.

Ačkoli je práce s těmito metodami poměrně jednoduchá, pokud je cílem vývojáře vytvořit věrohodnější prostředí s určitou prostorovou deformací, je potřeba využít modelu a metod pro převod souřadnic do „reálného prostředí“ a zpět. Z tohoto důvodu existují enginy, jejichž cílem je co nejvíce usnadnit vývojáři tvorbu takové aplikace.

Právě vytvoření takového nástroje a sepsání jeho dokumentace je primárním cílem této práce. Dalšími aspekty práce je snaha přiblížit čtenáři problematiku izometrické deformace a srovnat různé technologie, pomocí kterých je možné dynamicky vykreslovat grafiku v rámci webového prohlížeče. Ke konci práce je čtenáři předložena názorná ukázka využití vytvořeného enginu na světoznámém konceptu hry Snake. Tento a další příklady také demonstrují zjednodušení a zkrácení vývoje podobných aplikací a zároveň představují způsoby, jak je možné tento nástroj využít v praxi.

Princip tzv. zabalení zdrojového kódu do jednoho nástroje, není nijak neobvyklý, spíše naopak, žádný složitější počítačový program se bez podobných enginů neobejde. V sou-

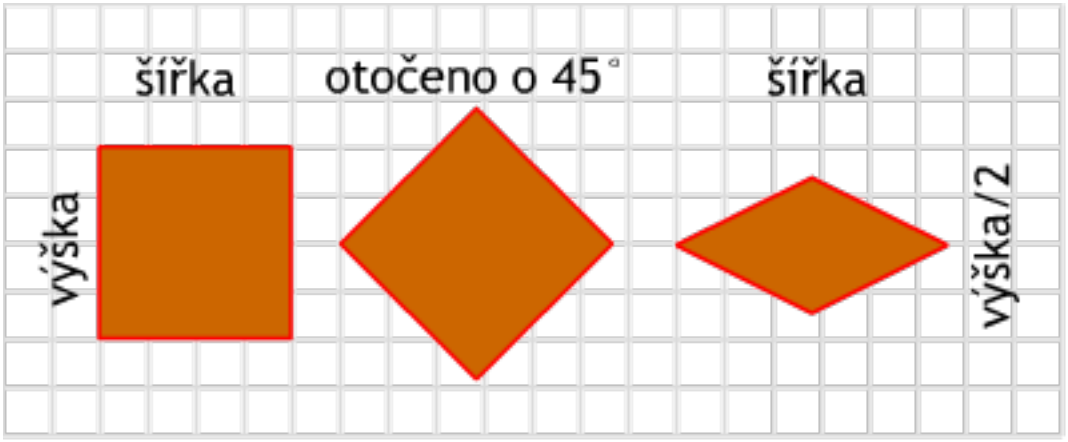
časnosti existuje i nespočet[1] projektů zaměřených právě na vývoj nástroje pro usnadnění tvorby aplikací s izometrickým zobrazením, nicméně drtivá většina z nich je vyvíjena týmy o více lidech, nebo dokonce zaštiťována většími společnostmi. Proto největší přínos bude mít tato práce pro začínající vývojáře, kteří mají zájem o tuto problematiku.

[1] Více v kapitole *Současný stav*

1 Izometrie

Izometrické zobrazení je jedním ze tří druhů[2] axonometrie, které slouží k jednoduchému promítnutí prostorových objektů a trojrozměrných struktur do roviny (2D). V izometrické projekci svírají osy souřadnicového systému úhel 120 ° a objekty jsou nanášeny bez deformace rozměrů stran (v poměru 1:1:1), a tím je vytvářena jednoduchá iluze trojrozměrného prostoru.

Pravděpodobně nejlepším způsobem, jak si izometrické zobrazení představit, je podívat se, co se stane se čtvercem, pokud ho takto transformujeme. Viz obrázek 1. Na tomto obrázku si, mimo jiné, můžete všimnout, že se délky stran původního čtverce nezměnily.

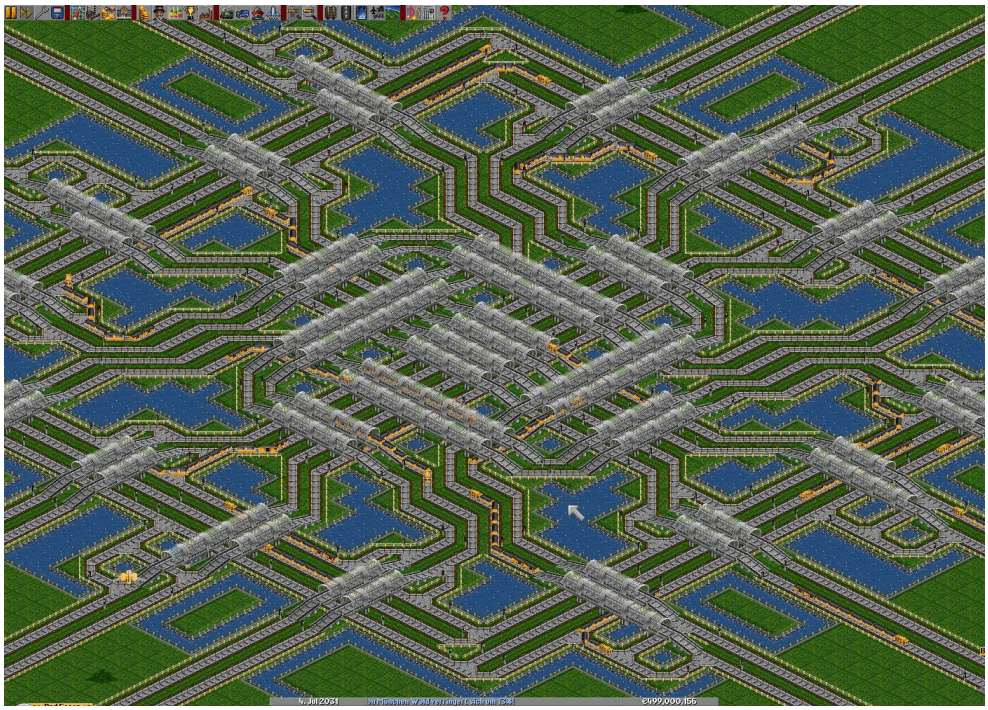


Obrázek 1: Postup transformace do izometrického zobrazení

Tento způsob zobrazování je hojně využíván pro nákresy součástek v technických oborech, ale také v počítačové grafice především u tzv. dlaždicových her (tile based games), jako je například *Open TTD*, což je open-source klon velmi populární hry Transport Tycoon Deluxe (na obrázku 2 je zachycen screenshot ze hry).

V hrách, nebo jiných aplikacích, se izometrie využívá především díky tomu, že je s její pomocí možné poměrně jednoduše vytvořit relativně realisticky vypadající prostředí s ho-

[2] Dalšími jsou dimetrie a trimetrie



Obrázek 2: Screensohot ze hry OpenTTD

rizontální rozdíly v krajině[3].

[3]Například vícepatrové domy, patra skladů nebo prohlubně.

2 Dostupné technologie

Tato kapitola bude věnována popisu nejdůležitějších nástrojů, pomocí kterých je možné projekt realizovat. Metody, které byly zahrnuty do užšího výběru, byly především na JavaScriptu založený Canvas a XML využívající SVG. Jsou zde i další způsoby, jak realizovat tuto práci, o důvodech proč jejich služeb nebylo využito se dozvíte dále v této kapitole.

Prvním úkolem bylo zvolit jednu z těchto metod a vzhledem k tomu, že jsou značně odlišné, to není jen formální záležitost. Právě toto rozhodnutí totiž může zásadně ovlivnit průběh i výsledek práce. To je také důvod, proč je této problematice věnována celá kapitola. Boudou zde popsány výhody, nevýhody a rozdílnosti jednotlivých technologii.

2.1 Flash

Flash je pravděpodobně první technologie, nad kterou by uvažovala většina vývojářů zaměřených tvorbu her v prohlížeči. Už jen to, že do veřejného povědomí vstoupily termíny jako „flashová hra“ nebo „flashovka“ naznačuje popularitu této technologie. Další výhodou může být relativně nenáročná tvorba výsledné aplikace, v tomto případě hry. Je to především díky tomu, že Flash obsahuje grafické rozhraní, ve kterém nakreslíte jednotlivé objekty hry, tak jak potřebujete, a až později k nim dodáváte actionscriptový kód. I přes popularitu a relativní nenáročnost vývoje výsledného produktu tuto technologii nepoužijeme, a to především proto, že není závislá pouze na prohlížeči, ale je potřeba doplňující software – Adobe Flash Player. Tento doplněk je sice velmi populární, ale existují i významné výjimky (např. zařízení od firmy Apple), které tento software nepodporují. Navíc, nejdůležitějším cílem této práce je vytvořit engine (tedy ne výsledný produkt – hru, vizualizaci skladu, atp. . .), který bude závislý pouze na podpoře v prohlížeči (o tom později).

Pokud byste se chtěli o vývoji dlaždicových izometrických her dozvědět více, navštivte webovou stránku
`http://www.tonypa.pri.ee/tbw/tut16.html` [26.1.2014] (anglicky), odkud byla

také čerpána inspirace a nápady při realizaci práce.

2.2 HTML5 & CSS

V dnešní době zažívají tyto technologie obrovský boom a již nyní existuje velké množství projektů[4], které se zabývají vytvářením nových her, nebo pomocí těchto technologii přepisují hry již existující, a umožňují tak téměř všem uživatelům internetu hraní přímo v prohlížeči.

Pokud tvoříte hry tímto způsobem, můžete využívat struktury DOM, stejně jako při tvorbě libovolné webové stránky. Jednotlivé objekty uložené ve struktuře DOM můžete později animovat pomocí CSS3 a aplikovat na ně libovolné javascriptové kódy (například knihovny pro implementaci fyzikálních zákonů).

Ukládání všech objektů daného prostředí je zároveň výhodou, protože umožňuje snazší přístup ke všem objektům, ale také nevýhodou z hlediska výpočetní náročnosti. Pokud bude prostředí dostatečně velké, bude jeho vykreslení a práce s ním velmi náročná (podobný problém má i SVG, o kterém bude řeč později). Další nevýhodou může být prozatímní absence pevně daných standardů HTML5 a CSS3.

2.3 WebGL

WebGL je javascriptové API pro interpretaci 3D a 2D grafiky v jakémkoli z kompatibilních prohlížečů bez použití doplňků.

Jedná se o v současnosti nejmodernější a nejefektivnější způsob zobrazení 3D grafiky v prohlížečích. Využívá HTML5 element Canvas a rozhraní DOM a je obsluhováno pomocí zdrojového kódu JavaScriptu. Je založeno na OpenGL ES 2.0 a umožňuje přenést výpočty na grafickou kartu počítače.

WebGL vzniklo z experimentů s elementem Canvas v rámci organizace Mozilla Fundation.

[4] Více v kapitole *Současný stav*

V prohlížečích Mozilla Firefox a Opera je implementováno od roku 2007. Nyní je spravováno neziskovou organizací Khronos Group.

Názornou a velmi efektní ukázkou využití WebGl může být například projekt z řady Chrome Experiment od vývojářů Google: `http://www.ro.me`

2.4 SVG

SVG je zkratka pro Scalable Vector Graphics (škálovatelná vektorová grafika). Jedná se o značkovací jazyk pro vytváření dvourozměrné vektorové grafiky v XML. Krom toho jste možná slyšeli o SVG jakožto typu souboru, který slouží k ukládání této vektorové grafiky. Podobně jako například v jazyce HTML nebo XHTML vkládáme do zdrojového kódu elementy, nicméně zde místo jednotlivých prvků stránky (odkaz, odstavec, atd.) reprezentují jednotlivé prvky grafiky. SVG ukládá všechny tyto elementy do struktury DOM, což má své výhody i nevýhody. Jednoznačnou výhodou je jednoduchý přístup k jednotlivým prvkům grafiky, můžeme snadno přidělovat pravidla kaskádových stylů naprosto stejným způsobem jakoby to byly tagy v jazyce HTML. Nevýhodou je to, že toto zaznamenávání je časově náročnější, takže výsledné aplikace jsou limitovány velikostí (rozsahem).

2.4.1 Podpora v prohlížečích

Toto kritérium je velmi důležité především u komerčních projektů, které jsou zaměřeny na širokou cílovou skupinu, nicméně přestože je práce open-source a necílí ani na velkou uživatelskou obec, ale směřuje spíše do řad nadšenců, je pro nás tento parametr důležitý především proto, že se snažíme vytvořit obecný a jednoduše rozšiřovatelný engine.

Obecně by se dalo říct, že všechny moderní prohlížeče a jejich nejnovější verze s touto technologií nemají problémy, nicméně nesmíme zapomínat na stále početnou skupinu uživatelů, kteří pracují se staršími verzemi prohlížečů.

Prohlížeč	Podporuje od verze
Internet Explorer	9 beta (částečně)
Mozzila Firefox	3.5 (problémy s animací)
Opera	11 (bez problémů)
Google Chrome	8.0 (včetně animací)

Tabulka 1: Podpora SVG v prohlížečích [8]

2.5 Canvas

Na začátek by byla vhodná citace autora knihy o této metodě a HTML obecně: „O rozhraní Canvas jazyka HTML5 by bylo možné napsat celou knihu (a nebyla by malá)“[3]. Takže informace v této kapitole uvedené budou pouze špičkou ledovce a spíše velmi stručným shrnutím všech důležitých aspektů této techniky. Canvas je rozhraní umožňující dynamicky generovat a vykreslovat grafiku, grafy, obrázky, texty i animace. V českých publikacích se často můžeme setkat také s překladem *plátno*. Použijeme-li element `<canvas>` ve struktuře souboru HTML5, výsledkem bude bílá plocha[5], jejíž obsah poté tvoříme a vykreslujeme pomocí funkcí JavaScriptu. Jednotlivé objekty umísťujeme do pomyslné sítě souřadnic s počátkem v levém horním rohu.

2.5.1 Historie

Princip plátna původně představila firma Apple ve svém WebKitu pro Mac OS X za účelem vytvoření widgetů pracovní plochy. Před příchodem plátna bylo možné v prohlížeči realizovat malování pouze skrze zásuvné moduly, jako jsou například Flash od Adobe, SVG a VML anebo jiné, čistě javascriptové techniky.

[5] Ve výchozím nastavení má rozměry 300 na 150 pixelů, což lze samozřejmě jednoduše změnit

2.5.2 Podpora v prohlížečích

Dá se říci, že je podporován všemi moderními prohlížeči a nebude fungovat pouze ve starších verzích Internet Exploreru, které v době psaní této práce tvoří jen minoritní podíl na českém internetu, a do budoucna se počítá spíš s jejich dalším úpadkem. I tato dobrá podpora v prohlížečích byla jedním z důvodů, proč bylo jakkoli rozhodnuto využívat tuto metodu. Jedním z dalších důvodů výběru právě této metody byl článek `http://www.zdrojak.cz/clanky/svg-nebo-canvas-vyberte-si/` [26.1.2014] uveřejněný na webu zdrojak.cz. Především pak tabulka srovnání SVG a Canvasu. Pokud vás zajímá více k tomuto tématu, nebo se právě i vy rozhodujete mezi SVG a Canvasem, tento článek vám s velkou pravděpodobností rozšíří obzory a třeba i pomůže správě vybrat.

Prohlížeč	**Podporuje od verze**
Internet Explorer	9
Mozzila Firefox	1.5
Opera	9.0
Google Chrome	1.0 (včetně animací)

Tabulka 2: Podpora Canvas v prohlížečích [9]

2.5.3 Základy práce s prostředím canvas

Pokud chceme začít pracovat s canvasem, je nejdříve potřeba vytvořit element `<canvas>`:

```
1   <canvas id="canvasID" style="border: 1px solid red">
2       Tento text se zobrazí v prohlížečích, které Canvas nepodporují
3   </canvas>
```

Zdrojový kód 1: Vytvoření nového elementu canvas

Jak můžete vidět v ukázce, tomuto elementu (stejně jako všem ostatním v HTML) můžete přidávat parametry a psát pro něj pravidla v CSS. Nejdůležitějším parametrem je `id`, jelikož pomocí něj se budeme na tento element odvolávat z našich javascriptových kódů.

Na jedné stránce může být těchto elementů i více.

Máme-li takto připravenou stránku, můžeme začít psát JavaScript, pomocí kterého budeme canvas plnit obsahem. Dříve než začneme kreslit, je potřeba najít element `<canvas>` ve struktuře DOM a poté pomocí metody `getContext("2d")` načíst jeho obsah do proměnné.

```
1 var c=document.getElementById("canvasID");
2 var ctx=c.getContext("2d");
```

Zdrojový kód 2: Načtení kontextu

Nyní se můžeme pustit do kreslení na plátno. Základní metodou je kreslení rovných čar pomocí metod `moveTo(x,y)` a `lineTo(x,y)`.

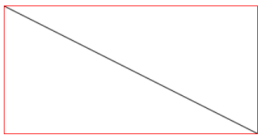
Nejjednodušším způsobem vysvětlení jejich fungování je příklad: Představte si, že před sebou máte prázdný bílý papír a v ruce tužku. Metoda `moveTo()` reprezentuje pohyb tužkou nad papírem, zatímco `lineTo()` zaštiťuje tažení tužky po papíře a kreslení čáry.

Nicméně všechno, co vytvoříme pomocí těchto dvou metod, je stále jen jakýmsi náčrtkem. Abychom docílili vykreslení obrazce[6], musíme použít metodu `stroke()`.

```
1 var c=document.getElementById("canvasID");
2 var ctx=c.getContext("2d");
3 ctx.moveTo(0,0);
4 ctx.lineTo(200,100);
5 ctx.stroke();
```

Zdrojový kód 3: Vykreslení diagonály

Výsledek tohoto kódu bude vypadat takto:



Obrázek 3: Diagonála vykreslená pomocí moveTo() a lineTo()

[6] tento proces je možné si představit například jako obtažení výsledku

Výčet dalších metod pro vytváření obsahu:

- `arc(x, y, r, start, stop)` – kreslení kruhu
- `fillText("Hello World", 10, 50)` – vypisování textu
- `drawImage(cestaKSouboru, x, y)` – vykreslení obrázku

Jak bylo řečeno na začátku tohoto oddílu, možnosti canvasu jsou velmi rozsáhlé, pro více informací doporučuji například tuto webovou stránku: `http://www.w3schools.com/html/html5_canvas.asp` [26.1.2014] (anglicky), nebo odbornou literaturu jako třeba *HTML5 audio a video kompletní průvodce* od Silvie Pfeiferové. Mimo jiné i tyto zdroje byly využity k realizaci práce.

3 Současný stav

Jak již bylo řečeno v úvodu, v současné době existuje velké množství projektů zabývajících se touto problematikou. V této kapitole boudou zmíněny některé z nich a stručně popsány jejich přednosti, případně nedostatky. První vadou na kráse, ale zároveň i výhodou většiny v současnosti existujících technologií je jejich komplexnost a s tím spojená složitost zdrojových kódů. Pokud chcete daný engine použít pro konstrukci své nové hry, pak jistě oceníte, že efektivním způsobem implementuje vše, na co si jen vzpomenete. Nicméně pokud se snažíte pochopit, jak doopravdy funguje práce s izometrickým prostředím, pak se v rozsáhlých zdrojových kódech budete jen špatně orientovat. Proto je jedním z cílů této práce vytvořit funkční engine s co možná nejsrozumitelnějším zdrojovým kódem.

- `http://www.jsiso.com`

  Tento engine by se dal označit za nejjednodušší ze zde zmíněných, obsahuje funkce na detekci vstupů z klávesnice, kliknutí myši i klepnutí na dotykovém zařízení. Prostředí je vykreslováno do předem inicializovaných vrstev. Vykreslenou mapu je možno otáčet, přibližovat a oddalovat. Obsahuje jednoduchý pathfinding s využitím Web Workers. Pro svou inicializaci a funkčnost využívá jiného javascriptového frameworku a to RequireJS.

- `https://sheetengine.codeplex.com/`

  Jedná se o složitější nástroj, který se neomezuje pouze na izometrické zobrazení. Využívá rozhraní canvas a umožňuje vertikální řazení objektů, má dobře implementované funkce na vytváření stínů, jednotlivé objekty je do sebe možné vnořovat a podporuje i komunikaci se serverem.

- `http://www.isogenicengine.com/`

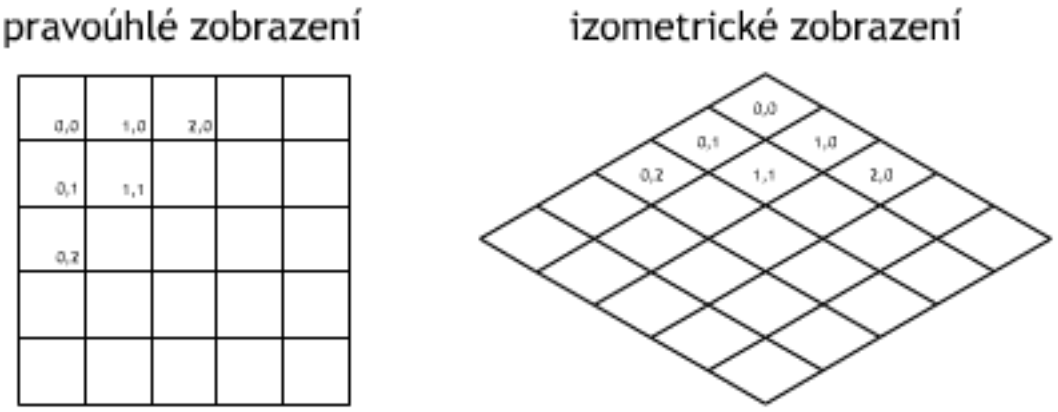
  Tento nástroj je pravděpodobně nejsložitější. Není vyvíjen jednotlivcem (stejně jako předchozí), ale firmou Irrelon Software Limited. Jedná se o nástroj zaměřený přímo na hry. Podporuje muliplayer v reálném čase pomocí Websockets. Má přehledně napsanou dokumentaci a implementuje i fyzikální baličky.

4 Postup vývoje

Na úplném začátku práce bylo nezbytné vybrat technologii, pomocí které bude naprogramován výsledný engine. Více o problematice výběru nejvhodnější technologie se můžete dočíst v kapitole *Dostupné technologie* na straně 12. Na základě vybrané technologie, tedy rozhraní `<canvas>`, bylo třeba nastudovat odbornou literaturu a pochopit objektový způsob programování v JavaScriptu, jelikož bez něj není možné vytvořit intuitivní engine. Tyto teoretické informace byly čerpány především z publikace *JavaScript kompletní průvodce* od Davida Flanagana. Po nastudování principů objektového programování (dále jen OOP) přišla řada na podrobnější obeznámení se s konkrétními možnostmi plátna. Na základě všech těchto nabytých poznatků bylo možné začít navrhovat strukturu a začít programovat.

Základním aspektem, nezbytným pro jakoukoli další práci, bylo sestavení soustavy rovnic pro převod souřadnic mezi modelem a „reálným“ prostředím canvas, kde byly objekty transformovány do izometrického zobrazení.

Pokud bychom pracovali pouze s pravoúhlým zobrazením, stačilo by pro získání správné pozice na plátně vynásobit x-ovou souřadnici buňky v modelu její šířkou, to stejné u y-ové souřadnice s výškou, a jsme na správném místě. U izometrického zobrazení je nutno tuto základní rovnici pozměnit.



Obrázek 4: Izometrické vs. pravoúhlé zobrazení

Výsledné rovnice vypadají takto:

- Převod souřadnic z modelu do izometrie

```
isoX = ((x-y)* (tile.width/2))+Map.offset.x;

isoY = (((x+y)*(tile.height/2))+Map.offset.y; ——
```

- Převod souřadnic z izometrického zobrazení do modelu

```
x -= offset.x;

y -= offset.y;

modelX = Math.round(((x/(tile.width/2))+ (y/(tile.height/2)))/2 - 1);

modelY = Math.round(((y/(tile.height/2))- (x/(tile.width/2)))/2);
```

Každá z těchto dvou soustav byla „uzavřena“ v oddělené metodě, jak můžete vidět na ukázce 4:

```
1  TRANSFORM: {
2              toIso: function (x,y){
3                  var isoX = ((x-y)* (iwe.tile.width/2))+iwe.Map.offset.x;
4                  var isoY = (((x+y))*(iwe.tile.height/2))+iwe.Map.offset.y;
5                  return {x: isoX, y: isoY};
6              },
7              toModel: function(x,y){
8                  x -= iwe.Map.offset.x;
9                  y -= iwe.Map.offset.y;
10                 var modelX = Math.round(((x/(iwe.tile.width/2)) + (y/(iwe.tile.height/2)))
                       /2 - 1);
11                 var modelY = Math.round(((y/(iwe.tile.height/2))- (x/(iwe.tile.width/2)))
                       /2);
12                 return {x: modelX, y: modelY};
13             }
14         },
```

Zdrojový kód 4: Metoda TRANSOFRM

Více o využívání těchto metod v praxi se dozvíte v dokumentaci.

Když bylo nyní vyřešeno přepočítávání, přišlo na řadu naplňování a vykreslování mapy.

Za účelem testování byly vytvořeny jednoduché sprity[7] o rozměrech 52 na 30 px. Tyto obrázky byly vykreslovány odshora dolů v závislosti na modelu, který byl reprezentován polem polí[8]. Pokud by byly sprity vykreslovány z jakéhokoli jiného směru, mohlo by dojít k problémům při vykreslování vyšších objektů (vícepatrové budovy atp.).

Jakmile je možné vykreslovat mapu a měnit základní parametry jednotlivých spritů, musíte se v tomto prostředí nějakým způsobem pohybovat. Je potřeba posouvat celý obsah. Tato funkčnost byla zajištěna přidáním parametru `offset` k instanci objektu Map a následnou drobnou úpravou již zmíněných rovnic[9].

O dalších funkcích a jejich využití v praxi se můžete více dozvědět v přiložené dokumentaci.

[7] Malé obrázky ze kterých se poskládá celá scéna. Tato technika se využívá z důvodu menší náročnosti na výkon, než vykreslování jednoho velkého obrázku.

[8] V JavaScriptu, na rozdíl od C,C++ a dalších, neexistuje tzv. dvojrozměrné pole, ale je nahrazováno právě polem polí.

[9] Tato úprava je již do rovnic zahrnuta.

5 Ukázky využití

Nejlepším způsobem, jak demonstrovat, k čemu je možné tento nástroj v praxi využít, je vytvoření několika jednoduchých příkladů. Je důležité upozornit, že ve chvíli kdy tuto práci čtete, se mohou příklady příklady lišit, protože jsou umístěny serveru GitHub.com a pravděpodobně budou dále rozvíjeny.

5.1 Map builder

Ať už chcete vytvořit dlaždicovou hru, izometrickou vizualizaci nebo jakoukoli jinou interaktivní aplikaci, je dobré si vytvořit nástroj na jednoduchou úpravu podkladového prostředí. Účelem následujícího příkladu je demonstrovat, jak je toho možné dosáhnout. Kompletní zdrojový kód je možné najít na serveru GitHub.com - `https://github.com/TonnyVlcek/IzometrickeZobrazeniHTML5`, zde budou popsány jen klíčové aspekty této jednoduché aplikace.

V případě, že chcete využít tohoto izometrického enginu, je nejdříve potřeba jej připojit k vašemu souboru:

```
<script type="text/javascript" src="./iwe.js"></script>
```

Dalším nezbytným krokem je vytvoření elementu `<canvas>` s parametrem id. Jakmile existuje tento element, přichází řada na inicializaci, nastavení základních parametrů enginu a nahrání potřebné grafiky:

V tuto chvíli jsme připraveni začít tvořit logiku aplikace. Potřebujeme napsat funkce, které se provedou při určitých událostech. V této konkrétní aplikaci jsou implementovány funkce pro:

- Kliknutí myší, která v závislosti na nastaveném režimu (viz dále) posouvá jednotlivé dlaždice nahoru respektive dolů po pomyslné ose Z a tím vytváří vertikální členitost prostředí.

  Pokud je kliknuto v rozmezí souřadnice, kde se nachází jednoduché menu, pozmění

```
iwe.INIT({
        canvas: 'main', //id elementu canvas
        tile: {width: 52, height: 30},
        startAt: {x: 300, y: 200},
        fullScreen: true
    });

iwe.loadImages('./images/', [   'default.png','block.png','hero.png','highlite.png',
                                    'menu_1.png','menu_2.png'
                                ]);
```

Zdrojový kód 5: Inicializace enginu iwe

se tato funkce a posouvá dlaždice opačným směrem.

- Zvýraznění aktuální buňky - překreslí („obtáhne“) danou buňku zvýrazňujícím spritem (viz obrázek 5).
- Pohybování s celým vytvořeným prostředím. Tato funkce zjišťuje rozdíl souřadnic při stisknutí a uvolnění levého tlačítka myši a v reálném čase nastavuje tento rozdíl jako offset (odsazení pro celou mapu).
- Pohyb s hrdinou. Pomocí šipek je díky této funkci možné měnit pozici hrdiny a tak testovat, zda-li vytvořené prostředí odpovídá požadavkům.

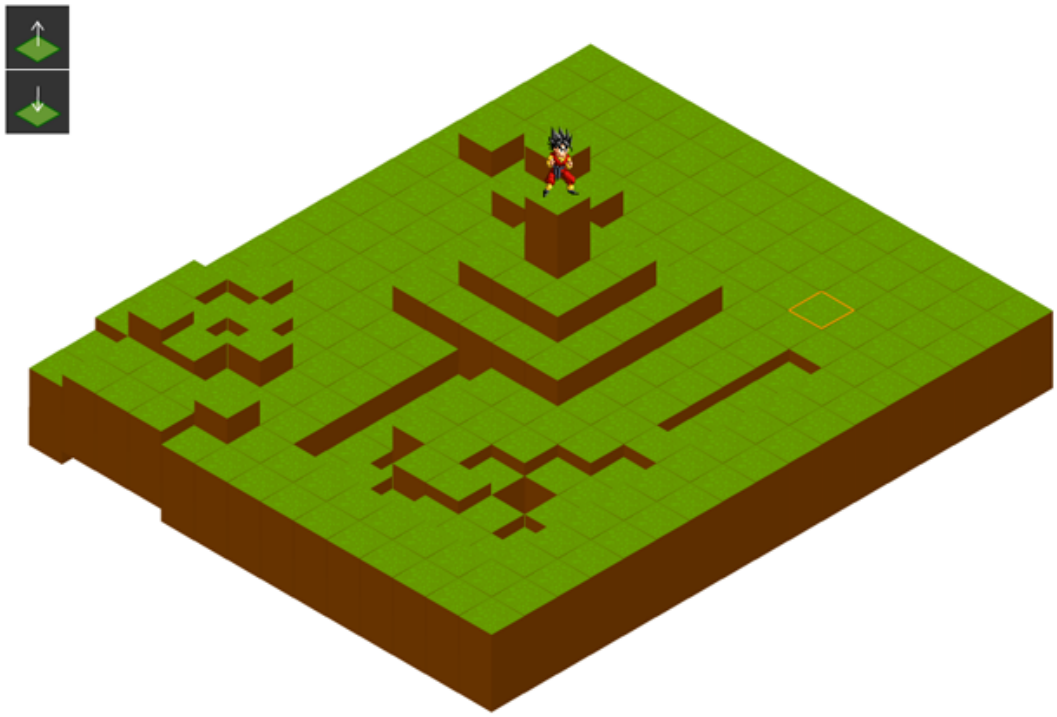
Zde zmíněné funkce jsou pouhým základem pro vytváření mapy pomocí tohoto enginu, není příliš náročné vytvořit mnoho dalších[10].

5.2 Hrdina a kolize

Na tomto jednoduchém příkladu je demonstrována detekce kolizí mezi statickým objektem a uživatelem ovládaným hrdinou.

Na začátku opět inicializujeme engine a načteme potřebnou grafiku. Poté do pole polí uložíme informaci, na jakém místě se má vyskytovat jaká dlaždice. Mapa může být samozřejmě větší, nebo jí není nutné vytvářet ručně, ale můžeme být naplněna pomocí

[10]Např. vkládání jiných dlaždic (voda, stromy, krabice), jedním kliknutím ovlivňovat výšku více objektů, atd.



Obrázek 5: Screenshot z ukázky mapBuilder

určitého algoritmu. Nyní vytvoříme prototypy dlaždic, které poté uložíme do modelu enginu, abychom s nimi mohli pracovat za pomocí vestavěných metod. V následující ukázce kódu si můžete všimnout, že první prototyp má parametr walkable nastavený na true a druhý na false. Tento parametr bude rozhodovat o tom, zda-li může náš hrdina na toto pole vstoupit.

Nakonec tu máme několik funkcí zajišťujících primitivní logiku „hry“. Z nich stojí za zmínku ta, která má za úkol detekovat kolize:

Při pohybování s hrdinou se pokaždé odkážeme na tuto funkci, která se „podívá“ do 2D modelu a na základě parametru *walkable* vrací logickou hodnotu true nebo false.

```
var map = [
                    [0,0,0,0,0,0,0,0,0,0],
                    [0,1,1,0,0,0,0,0,0,0],
                    [0,0,1,1,0,0,0,0,0,0],
                    [0,0,0,1,1,0,0,0,0,0],
                    [0,0,0,0,1,0,0,0,0,0],
                    [0,0,0,0,1,1,1,1,1,1],
                    [0,0,0,0,1,0,0,1,0,0],
                    [0,1,1,1,1,0,1,1,0,0],
                    [0,0,0,0,1,0,1,0,0,0],
                    [1,1,1,1,1,1,1,1,1,1]
            ];
```

Zdrojový kód 6: Mapa prostředí

5.3 Snake

Tato ukázka je pravděpodobně nejkomplexnější demonstrací. Fakticky se jedná o plně funkční hru Snake, tak jak ji znáte z vašich starých mobilních zařízení, nicméně v izometrickém zobrazení.

Opět zde bude zmíněno několik klíčových funkcí pro vytvoření této hry.

Začátek je znovu stejný jako u ostatních ukázek, jen je potřeba nahrát větší množství grafiky. Dohromady je nutných 16 obrázků:

- 2 obrázky na podkladovou mapu - pro odlišení hraničních dlaždic
- 1 obrázek jako potrava - v našem případě jablko
- 4 obrázky pro každý směr pohybu hlavy, segmentu těla a ocasu (12 celkem)
- 1 obrázek pro znázornění ohybu hada [11]

Po dokončení těchto rutinních kroků je potřeba vytvořit pilotní objekty SNAKE, FOOD a konstruktor cSegment. Pak je potřeba definovat několik funkcí, které budou zajišťovat herní logiku a vykreslování:

[11] pokud bychom chtěli rozlišovat i směr, je opět zapotřebí 4 obrázků

```
var tiles = {
  t0: function (x,y){
    this.x = x;
    this.y = y;
    this.offsetX = 0;
    this.offsetY = 0;
    this.spriteId = 0;
    this.walkable = true;
  },
  t1: function (x,y){
    this.x = x;
    this.y = y;
    this.offsetX = 0;
    this.offsetY = 0;
    this.spriteId = 1;
    this.walkable = false;
  }
};

iwe.fillMode();
```

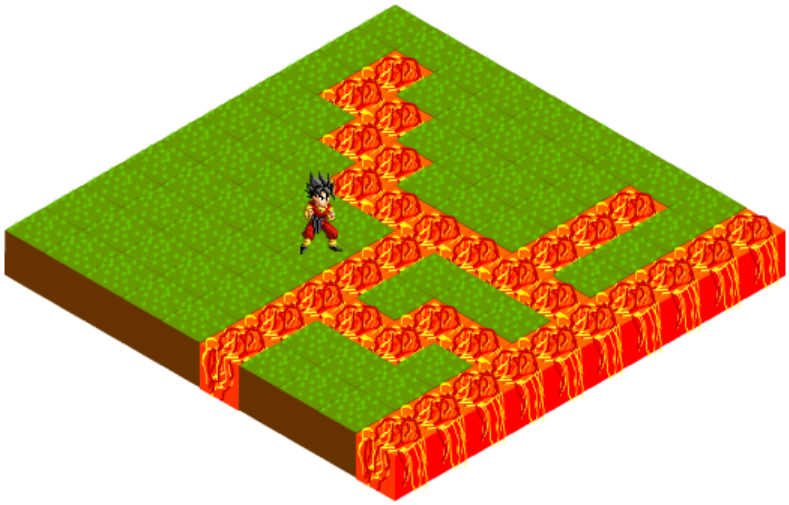
Zdrojový kód 7: Nastavení prototypů

- `pushSnake()` - vykresluje celého hada, na základě směru každého segmentu rozhoduje, jaký sprit mu bude přiřazen
- `moveSnake()` - zajišťuje správný pohyb hada, detekuje kolizi, kdy se had zakousne sám do sebe, nebo do zdi, zároveň také zjišťuje, zda-li had nenarazil na jablko, v takovém případě přičte body a zavolá funkci newFood.
- `pushFood()` - vykresluje jablko
- `newFood()` - má za úkol vygenerovat novou náhodnou pozici pro jablko, pokud bylo minulé požito hadem, jablko se musí vyskytovat mimo hadovo tělo a uvnitř ohraničeného prostředí. Viz ukázka zdrojového kódu 9 na straně 29.

Největším problémem bylo zařídit, aby zbytek těla kopíroval pohyby hlavy bez nutnosti zaznamenávání jednotlivých pohybů do pole. Tento problém řeší funkce `moveSnake()`, a to tak, že při každém pohybu se cyklem projde celé pole, kde jsou uloženy instance objektů

```
function collision (x,y){
            if(!iwe.getTile(x,y).walkable){
                console.log('Collision on '+x+':'+y);
                return false;
            }else{
                return true;
            }
        }
```

Zdrojový kód 8: Nastavení prototypů



Obrázek 6: Screenshot z ukázky Hrdina a kolize

reprezentující jednotlivé části těla a každý segment získá souřadnice a směr předchozího segmentu. To, kterým směrem se had pohybuje, udává hlava, která je ovládána pomocí šipek. Směr jejího pohybu se pak „kopíruje“ na ostatní segmenty. Tuto část zmíněné funkce představuje ukázka zdrojového kódu 10 na straně 29. Tato funkce je periodicky volána pomocí vestavěné funkce JavaScriptu `setInterval()`, což má za následek pohyb hada. Rychlost, jakou se had pohybuje, ovlivňuje parametr `speed` objektu `SNAKE`, který udává počet milisekund, po jakou dobu had zůstane na jednom místě.

```
function newFood (){
  FOOD.x = Math.floor(Math.random() * (iwe.Map.model.length-3)) + 1;
  FOOD.y = Math.floor(Math.random() * (iwe.Map.model[0].length-3)) + 1;
  for(var i = 0; i <= SNAKE.body.length-1; i++){ //food will never appear in the snake
    if(FOOD.x === SNAKE.body[i].x && FOOD.y === SNAKE.body[i].y){
      newFood();
    }
  }
}
```

Zdrojový kód 9: Ukázka funkce newFood()

```
function moveSnake(){
  for(var i = 0; i <= SNAKE.body.length-2; i++){
    SNAKE.body[i].x = SNAKE.body[i+1].x;
    SNAKE.body[i].y = SNAKE.body[i+1].y;
        SNAKE.body[i].direction = SNAKE.body[i+1].direction;
    }
  if (head.direction === ”left”) {
    head.x --;
    head.spriteId = 5;
    head.offset.y = -36;
  }
  ...
  ...
  ...
}
```

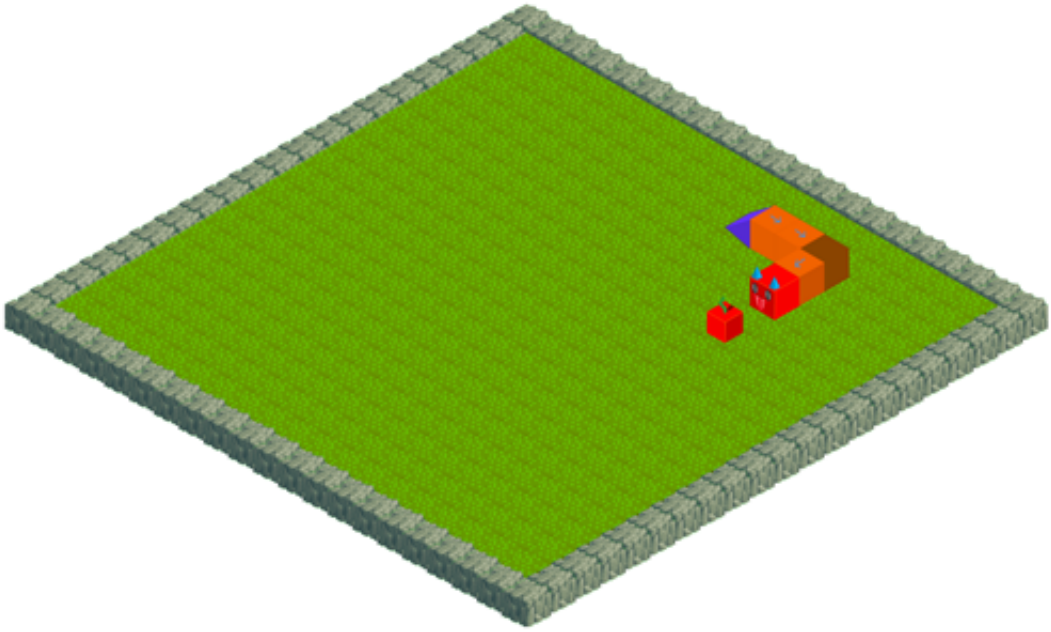
Zdrojový kód 10: Ukázka funkce moveSnake()

6 Možnosti rozšíření

Vzhledem k tomu, že cílem bylo vytvoření obecného nástroje, jsou možnosti rozšíření téměř neomezené. V této kapitole je uvedeno pár příkladů. Některé z těchto rozšíření vyžadují zásah přímo do enginu, jiné je možné realizovat i přímo v aplikaci.

- Stíny

  Canvas umožňuje načtení informací o každém vykresleném pixelu. Tyto informace je také možné měnit, tudíž by bylo reálné na základě určité podmínky (offset okolních



Obrázek 7: Screenshot ze hry Snake

dlaždic) o určité procento zatmavit barvu každého pixelu dlaždice, která leží ve stínu. Další možností je vytvoření speciálního zastíněného spritu pro každou dlaždici. Tento způsob je jednodušší na implementaci, nicméně při vykreslování rozsáhlejších prostředí je vysoce neefektivní.

- Export/import mapy za pomocí XML/JSON

  S trochou úsilí je možné obohatit engine o možnost exportu a importu informací o mapě ze souboru XML nebo JSON. Tato funkce je nad rámec této práce, nicméně její implementace by pravděpodobně nebyla příliš náročná a plánuje se v blízké budoucnosti.

- Pohyblivé sprity

  Vylepšení dojmu z prostředí a většímu přiblížení se realitě by tato funkce určitě prospěla. Prostředí canvas samo o sobě neumožňuje vkládání animovaných obrázků, bylo by tedy potřeba napsat metodu, která by výměnu obrázků zajišťovala sama, a do seznamu grafiky uložit každý snímek animovaného gifu.

- Přibližování a oddalování celého prostředí (zoom)

  Existuje více možností, jak toto rozšíření realizovat. Tím pravděpodobně nejjednodušším je uzavřít prostředí canvas do elementu `<div>` a poté měnit výřez, který bude viditelný. Touto metodou se však s přibližováním bude zhoršovat i kvalita bitmapových textur. Proto další metodou je nakreslení a načtení speciálních textur pro každou úroveň přiblížení.

- Otáčení celého prostředí

  Toto rozšíření je jedním z těch, které nutně nevyžadují změnu v enginu jako takovém. Implementace této funkce můžete dosáhnout i vlastním úsilím až ve zdrojovém kódu vaší aplikace. Princip je v tom, že ve chvíli, kdy chcete celou mapu otočit například o 90 stupňů stačí „přerovnat“ prvky v dvojrozměrném modelu. Otáčení v jiných úhlech (krom 90, 180, 270, 360 stupňů) by pravděpodobně vyžadovalo kreslení a nahrávání speciálních spritů.

- Pathfinding

  Pro zde zmíněné ukázky je tato funkce pravděpodobně zbytečná, nicméně pokud bychom pomocí enginu chtěli vytvořit například onlinovou hru, kde by vaše postava nebyla ovládána šipkami, ale kliknutím na místo určení, stala by se tato funkce nezbytnou.

Závěr

Na rozdíl od velkého množství podobných projektů je tento napsán tak, aby byl jeho zdrojový kód přehledný a co nejjednodušeji pochopitelný. Uživatel–programátor, kterému nebude stačit dokumentace nebo bude chtít být víc než jen uživatelem tohoto enginu a bude chtít zasáhnout do jeho vývoje, nebo začít se svým projektem, má možnost podívat se na veškeré zdrojové kódy na stránce GitHub (`https://github.com/TonnyVlcek/IzometrickeZobrazeniHTML5`), přečíst si dokumentaci a od základu tak pochopit principy a úskalí práce s izometrickým prostředím.

Na základě zde představených ukázek je možné s klidným svědomím říci, že byl splněn hlavní cíl a to vytvoření funkčního enginu pro práci s izometrií. Ve výsledku je tedy možné tento nástroj použít i pro realizaci jednodušších projektů.

Jak již bylo zmíněno v předchozí kapitole, existuje nespočet možností, jak engine vylepšovat a zvyšovat jeho kvalitu. Vzhledem k tomu, že autora práce téma zaujalo, je velmi pravděpodobné, že se dočkáte dalších úprav a vylepšení.

Celá tato práce, všechny zdrojové kódy i prezentace jsou publikovány pod licencí LicenceApache 2.0.

Zdroje

[1] FLANAGAN, David. JavaScript: kompletní průvodce. 2. aktualiz. vyd. Praha: Computer Press, 2002, xiv, 825 s. ISBN 80-7226-626-8.

[2] MAKZAN. Programujeme hry v HTML5. 1. vyd. Brno: Computer Press, 2012, 320 s. ISBN 978-80-251-3731-4.

[3] LUBBERS, Peter, Brian ALBERS a Frank SALIM. HTML5: programujeme moderní webové aplikace. Vyd. 1. Brno: Computer Press, 2011, 304 s. ISBN 978-80-251-3539-6.

[4] PFEIFFER, Silvia. HTML5 - audio a video: kompletní průvodce. Vyd. 1. Brno: Zoner Press, 2011, 350 s. ISBN 978-80-7413-147-9.

[5] BELLANGER Clint. `http://clintbellanger.net/articles/isometric_math/` [24.2.2014] (anglicky)

[6] MALÝ Martin `http://www.zdrojak.cz/clanky/svg-nebo-canvas-vyberte-si/`[24.2.2014]

[7] TONY `http://www.tonypa.pri.ee/tbw/index.html` [24.2.2014] (anglicky)

[8] BUKVIČKA Tomáš `http://studium.vos-sps-jicin.cz/svg/data/podpora_v_prohlizecich.php` [24.2.2014]

[9] SEKERA Jiří `http://www.zdrojak.cz/clanky/canvas-rikejme-tomu-plocha-na-kresleni/` [24.2.2014]

Seznam obrázků

Listings

Seznam příloh

[1] Dokumentace k iwe

[2] Zdrojové kódy zkomprimované do formátu zip a přiloženy na CD

PŘÍLOHY

DOKUMENTACE

Toto je dokumentace k izometrickému enginu iwe, pokud se chcete dozvět více izometrii, vývoji tohoto enginu a dalších technologiích, všechny tyto informace jsou uvedeny v hlavní části této práce.

iwe

typ: objekt

popis: tento objekt zastřešuje veškerý obsah enginu. Všechny ostatní zde zmíněné metody jsou parametry tohoto objektu, tudíž k nim přistupujete pomocí tečkového zápisu. Například `iwe.INIT();`

iwe.INIT(objekt)

typ: funkce vstup: na vstupu očekává objekt, s následujícími parametry:

- canvas - řetězec, odkazující na identifikátor elementu `<canvas>`.
- tile - objekt s parametry width a height, které udávají rozměry jedné izometrické dlaždice.
- startAt - objekt s parametry x a y, které udávají počáteční offset mapy.
- fullScreen - boolean - volitelný parametr, pokud jej neuvedete, canvas bude mít vámi nastavené rozměry, stejně tak jako v případě, že hodnotu tohot parametru nastavíte na `false`. Pokud bude mít tento parametr hodnotu `true`, bude canvas již od začátku zabírat celou plochu prohlížeče.

ukázka využití:

```
1 iwe.INIT({
2   canvas: 'main',
3   tile: {width: 52, height: 30},
4   startAt: {x: 250, y: 15},
5   fullScreen: true
6
7 });
```

Zdrojový kód 11: Ukázka použití iwe.INIT()

iwe.Canvas

typ: Objekt

vystup: na základě indentifikátoru, který jste zdali do iwe.INIT(), se do tohoto uloží všechny informace o elementu `<canvas>`. Pokud změníte některý z parametrů, tato změna se projeví i na tomto elementu.

ukázka využití:

```
1 iwe.Canvas.height = 500;
2 iwe.Canvas.width = 1000;
3 iwe.Canvas.style = 'border: 2px solid red; cursor: pointer';
```

Zdrojový kód 12: Ukázka použití iwe.Canvas

iwe.tile

typ: objekt

vstup: tento objekt má paramtery width (číslo) a height (číslo). Tyto parametry ovlivňují převod souřadnic mezi 2D modelem a izometrickým prostředím. Hodnota těchto parametrů je zadáváná z funkce `iwe.INIT()` na základě hodnot, které jse zadali do objektu tile.

iwe.Map

typ: objekt

popis: tento objekt jako parametr obsahuje objekt offset s parametry x (číslo) a y (číslo). Krom toho, že můžete při inicializaci pomocí `iwe.INIT()` nastavit výchozí hodnoty těchto parametrů, jsou klíčové pro pohybování s celým obsahem prostředí canvas.

ukázka využití:

iwe.Model

typ: pole

popis: pomocí metody `iwe.fillModel(map, tiles)` se do tohoto pole uloží další pole, za vniku pole polí, které uchovává instance jednotlivých tříd, jejich konstruktory jsou předány funkci `iwe.fillModel(map, tiles)` v paramteru tiles.

```
1     function mapShift(e){
2             isDraged = true;
3             var coor = getMouseCoor(e);
4             iwe.Map.offset.x -= mouse_x - coor.x;
5             iwe.Map.offset.y -= mouse_y - coor.y;
6             mouse_x = coor.x;
7             mouse_y = coor.y;
8             redraw();
9         }
```

Zdrojový kód 13: Ukázka použití iwe.MAP.offset - pohybování s celým obsahem

iwe.fillModel(pole, objekt)

typ: funkce

vstup: Na vstupu očekává jako první parametr dvojrozměrné pole, které v každé buňce obsahuje číslo odkazující na konstruktor třídy, jako druhý parametr očekává pole jehož parametry budou jednotlivé konstruktory ve formátu t[číslo z pole]. Povinnými parametry konstruktoru jsou: x=číslo, y=číslo, offsetX=číslo, offsetY=číslo, spriteId=číslo(odkaz do pole načtené grafiky iwe.images[]).

V případě, že je vše tak jak má, uloží na správné místo do pole iwe.Model instanci třídy t[číslo].

ukázka využití:

iwe.isVisible(číslo, číslo)

typ: funkce

vstup: na vstupu očekává parametr x a y, jakožto reálné souřadnice nějakého objektu.

popis: na základě `iwe.Map.offset` rozhoduje, zdali je potřeba vykreslovat objekt na daných souřadnicích, nebo jestli je mimo výseč prostředí canvas a tedy by jeho vykreslování bylo zbytečnou zátěží procesoru/grafické karty. Tato funkce je využívána v metodě `iwe.DRAW.image(řetězec,číslo,číslo)`.

výstup: vrací `true`, pokud jsou souřadnice v dané výseči rozhraní canvas. Pokud nejsou, vrací `false`.

iwe.TRANSFORM

```
var map =   [
        [1,1,1],
        [0,1,0],
        [0,1,0]
      ]
var tiles = {
        t0: function (x,y){
                this.x = x;
                this.y = y;
                this.offsetX = 0;
                this.offsetY = 0;
                this.spriteId = 0;
                this.walkable = true;
        },
        t1: function (x,y){
                this.x = x;
                this.y = y;
                this.offsetX = 0;
                this.offsetY = 0;
                this.spriteId = 1;
                this.walkable = false;
        }
    };
iwe.fillModel(map, tiles);
```

Zdrojový kód 14: Ukázka použití iwe.fillModel()

typ: objekt

popis: slouží k logickému zabalení metod `iwe.TRANSFORM.toIso()`, `iwe.TRANSFORM.toModel()`.

iwe.TRANSFORM.toIso(číslo, číslo)

typ: funkce

vstup: Na vstupu očekává souřadnice x a y z 2D modelu, pomocí rovnic popsaných v práci je převede na souřadnice izometrické.

výstup: Objekt o parametrech x a y, které reprezentují izometrické souřadnice

iwe.TRANSFORM.toModel(číslo, číslo)

typ: funkce

vstup: Na vstupu očekává souřadnice x a y z izometrického prostřed, pomocí rovnic po-

psaných v práci je převede na souřadnice 2D modelu.
výstup: Objekt o parametrech x a y, které reprezentují souřadnice v 2D modelu.

iwe.DRAW

typ: objekt
popis: logické zabalení následujících metod: `iwe.DRAW.image()`, `iwe.TRANSFORM.tile()`, `iwe.TRANSFORM.all()`.

iwe.DRAW.image(objekt, číslo, číslo)

typ: funkce
vstup: Na prvním místě vstupu očekává instanci třídy `Image()` a na následujících dvou pozicích x-ovou a y-ovou souřadnici, kam má grafiku vykreslit.
popis: víceméně ekvivalent vestavěné funkce `Canvas.ctx.drawImage(url, x, y)`, s tím rozdílem, že před vykreslením obrázku testuje pomocí funkce `iwe.isVisible()`, zda-li má význam obrázek vykreslovat.
výstup: v případě, že se souřadnice x a y nacházejí v právě viditelném výřezu rozhraní canvas vykreslí grafiku na tyto souřadnice a vrátí true. V opačném případě vrací false.

iwe.DRAW.tile(objekt)

typ: funkce
vstup: jako jediný vstup očekává instanci dlaždice.
popis: načte souřadnice dlaždice, pomocí metody `iwe.TRANSFORM.toIso` je převede na izometrické a poté předá metodě `iwe.DRAW.image()`, aby dokončila vykreslování.

iwe.DRAW.all()

typ: funkce
vystup: žádný
výstup: žádný
popis: postupně projde celé pole `iwe.Model[]` a jednotlivé instance načtené z tohoto pole předává metodě `iwe.DRAW.tile(objekt)`.

iwe.images[]

typ: pole

popis: v tomto poli jsou uloženy instance třídy `Image()`. Toto pole použijete kdykoli se chcete odkázat na přednačtenou grafiku. Je naplněno pomocí funkce `iwe.loadImages(řetězec , pole)`

příklad využití:

```
1 iwe.DRAW.image(iwe.images[tile.spriteId], tile.x, tile.y);
```

Zdrojový kód 15: Ukázka použití iwe.images[

iwe.loadImages(řetězec, pole)

typ: funkce

vstup: Jako první parametr očekává řetězec odkazující na složku, ve které se nachází veškerá potřebná grafika. Na místě druhého parametru očekává pole, v jehož každé buňce je uložen řetězec s názvem souboru a příponou. Veškerou grafiku načte do dočasné paměti a instance třídy `Image()` uloží do pole `iwe.images[]`.

ukázka využití:

```
1 iwe.loadImages('./images/', ['grass.png','unwalkable.png','hero.png']);
```

Zdrojový kód 16: Ukázka použití iwe.loadImages()

iwe.getTile(x,y)

typ: funkce

vstup: očekává parametry x a y reprezentující souřadnice dané buňky v modelu.

výstup: vrací objekt dané dlaždice.

popis: tuto funkci se hodí využít například při detekci kliknutí na určitou dlaždici. Viz příklad využití. ukázka využití:

iwe.fullScreen() typ: funkce vstup: žádný

výstu: žádný

popis: pomocí této funkce je možné nastavit velikost elementu `<canvas>` přesně na velikost

```
function canvasClick(e){
  var mouseCoor = getMouseCoor(e);
  var coor = iwe.TRANSFORM.toModel(mouseCoor.x, mouseCoor.y);
  var tile = iwe.getTile(coor.x, coor.y);
  tile.offsetY += 10;
  redraw();
```

Zdrojový kód 17: Ukázka použití iwe.getTile() - kliknutí

okna v prohlížeči. Tuto funkci můžete volat kdykoli pomocí `iwe.fullScreen()`, nebo je možné nastavit fullscreen implicitně již od inicializace a to nastavení parametru `fullScreen` na `true`. Viz `iwe.INIT(objekt)`.

iwe.CLEAR()

typ: funkce

vstup: žádný

výstup: žádný

popis: zavoláním této funkce smažete veškerý obsah elementu canvas. Tato funkce, spolu s `iwe.DRAW.all()` je užitečná při jakémkoli překreslování.