Università degli Studi di Perugia
Facoltà di Scienze Matematiche, Fisiche e Naturali

Corso di Laurea in

Informatica

Tesi di Laurea

Implementazione di un editor per mondi virtuali con tecnologia X3DOM

Laureando:

Federico Frenguelli

Relatori:

Prof. Osvaldo Gervasi

Anno Accademico 2010–2011

Indice

Elenco delle figure

Introduzione

Questo progetto si pone l'obiettivo di sviluppare un editor di scene virtuali come web application utilizzando le ultime tecnologie a disposizione per il 3d nel web, frutto di un processo evolutivo e di standardizzazione iniziato circa venti anni fa, per testarne le potenzialità e comprenderne il grado di maturità raggiunto.

Si pensi ad un supermercato. Si immagini questo ambiente riprodotto in uno scenario virtuale. Due sono gli attori principali in gioco: il cliente, che si muove all'interno dell'ambiente, osserva e sceglie tra gli oggetti esposti cosa acquistare, e il gestore, che si occupa di predisporre ed organizzare gli scaffali con i prodotti a disposizione. Il gestore dunque configura la scena per il cliente. In uno spazio virtuale il gestore necessita di uno strumento, un editor che gli consenta di inserire, posizionare ed eliminare gli oggetti (in questo caso i prodotti del supermercato) e di salvare le proprie modifiche rendendole disponibili all'utente.

Il semplice caso d'uso descritto delinea un'applicazione di tipo gestionale, in cui diversi utenti accedono con vari livelli di autorizzazione ai servizi

messi a disposizione. È facile dunque pensare da subito ad un software per il desktop come alla soluzione più naturale e semplice da implementare, vista anche la necessità di rappresentare una scena 3d con la quale l'utente possa interagire. Negli ultimi anni si è però assistito ad un fiorire di web applications sempre più complesse, dalle "office suite" fino ai gestionali per aziende, programmi per il photo ed il video editing.

Seguendo dunque questo trend di migrazione delle applicazioni dal desktop al web, si fa strada l'idea di avere un editor 3d come quello descritto interamente in un browser. Possibilità questa interessante per tutti i vantaggi che una web application porta con se, quali la compatibilità cross-platform, nessuna installazione e update lato utente grazie all'utilizzo del browser come thin-client, facilità di integrazione con servizi web e possibilità di funzionalità avanzate (e.g. editing collaborativo). Sorgono però alcune domande. Quali tecnologie ci sono a disposizione per il 3d nel web oggi? Sono abbastanza mature per realizzare un applicazione affidabile e performante? Quali svantaggi presentano?

Il compito di trovare una risposta a queste domande è affidato alla prima parte di questo lavoro. Dapprima verranno introdotte le possibili soluzioni presenti nell'era ante web 2.0 quali i linguaggi VRML prima e X3D poi, integrati nei browser tramite plugins di terze parti, e verranno analizzati i motivi per i quali queste non hanno avuto il successo sperato. Da questo piccolo fallimento si vedrà poi come nasce l'idea di un 3d bel web senza plugin, "nativo", e quali tecnologie lo rendono possibile. Si parlerà dunque della specifica HTML5, ancora in via di definizione che grazie alle novità

introdotte pone le basi per il 3D nel web, di WebGL, un nuovo standard promosso dal Khronos Group che definisce un interfaccia di programmazione di basso livello per la creazione di contenuti 3D all'interno di un browser web, e di x3dom, un motore javascript che consente l'introduzione di elementi X3D come parte del DOM di una pagina HTML5 e che fa uso dell'API WebGL per visualizzarli.

L'unico modo di testare efficacemente questo sempre più fiorente intreccio di tecnologie è effettuare una prova sul campo. Il software Medea, fulcro di questo lavoro, nasce esattamente a questo scopo. Nella seconda parte si analizzeranno le specifiche funzionali e le caratteristiche tecniche del software. Si vedrà come sono state utilizzate ed interconnesse le tecnologie di cui sopra per raggiungere gli obiettivi prefissati. Si cercherà di valutare l'efficienza e l'usabilità della soluzione proposta, al fine di individuare e comprendere quali siano i pregi e quali i difetti e i limiti di questa implementazione di 3D nel web. Obiettivo finale è comprendere se queste tecnologie siano pronte e mature per supportare un'applicazione di produzione completa, performante, cross-platform e user-friendly.

Capitolo 1

Tecnologie utilizzate

Portare il 3d nel web non è cosa di poco conto. In un continuo processo evolutivo, gli sforzi di molte aziende ed utenti si sono sommati fino a creare il complesso ecosistema di tecnologie che oggi ci consentono di avere un pizzico di virtualità all'interno del browser. Nel seguito vengono illustrati uno ad uno questi strumenti, in ordine di apparizione, per avere una visione d'insieme chiara e completa. Alcuni di questi sono stati creati e sono evoluti pensando al web. Altri sono stati riadattati per essere integrati nel browser e per interagire con il World Wide Web.

Le tecnologie trattate di seguito sono tutti standard aperti[1], le cui specifiche sono disponibili pubblicamente e gratuitamente e chiunque può contribuire al processo di decisione e di sviluppo. Questo lavoro non può e non

[1]Open Inventor. (2011, July 12). In *Wikipedia, The Free Encyclopedia*. Retrieved 09:05, August 25, 2011, from `http://en.wikipedia.org/w/index.php?title=Open_Inventor&oldid=439077866`

vuole considerare soluzioni chiuse e proprietarie.

1.1 Extensible 3D Markup Language

In principio era VRML. Il Virtual Reality Modeling Language nasce nel 1994 e viene ratificato come standard ISO nel 1997 con il nome VRML97[2]. Questo linguaggio consente di rappresentare una scena 3d in un semplice file di testo con estensione "wrl", detto anche "world". Vertici, spigoli, materiali, parametri per il texturing ed effetti, gestione luci, animazioni e suoni sono specificati in una struttura ad albero di tipo "scene graph" con una sintassi[3] semplice ed intuitiva. Grazie a nodi sensore ed eventi è gestita l'interazione con l'utente. Si possono inoltre inserire degli script, codificati in java o javascript, in dei nodi appositi, garantendo la massima flessibilità ed adattabilità della scena disegnata.

Come in ogni nuovo standard, vi erano in VRML delle evidenti lacune che andavano colmate. Una delle maggiori critiche sollevate da utenti e sviluppatori era la mancanza di una naturale integrazione con HTML. Per dirlo con le parole di Chris Phillips, sviluppatore per uno standard concorrente di VRML conosciuto come Chromeffects, *"VRML has no integration with HTML. When the VRML guys built VRML [...] it was all about 3D. They didn't work with the World Wide Web Consortium. They didn't even think about it. They were too busy getting the 3D to work.* A questo si andavano ad aggiungere altre esigenze come, ad esempio, la necessità di trovare uno standard solido e ampiamente diffuso tra gli sviluppatori su

[2]ISO/IEC 14772-1:1997 – Per maggiori informazioni si veda `http://www.web3d.org/x3d/specifications/vrml/`

[3]La sintassi di un file VRML si basa sul formato standard Open Inventor, prodotto da SGI

cui basare il linguaggio. Questo avrebbe garantito facilità di integrazione in altre applicazioni (e.g. sistemi di authoring e di playback) ma anche un comportamento uniforme e consistente delle scene disegnate tra software di diversi produttori. Inoltre, vi erano numerose richieste di nuove features da parte degli utenti che andavano analizzate e selezionate per essere poi implementate.

A partire da questi requisiti nasce X3D. Naturale evoluzione di VRML, mantiene molte delle precedenti caratteristiche (garantendo retrocompatibilità) e allo stesso tempo amplia la specifica con nuove features[4] e capacità, prima fra tutte la possibilità di definire le scene utilizzando una sintassi XML[5]. La scelta di introdurre nella specifica l'Extensible Markup Language come nuovo formato va a colmare quel vuoto lasciato dalla mancata integrazione con HTML e il World Wide Web. XML è diventato la lingua franca del web ed, inoltre, è utilizzato da molte applicazioni per la rappresentazione e lo scambio dei dati, grazie anche alle numerose interfacce di programmazione per l'accesso ai dati, esistenti per ogni linguaggio, che ne rendono semplice la gestione, il controllo e la validazione.

[4]Tra quelle non citate, degne però di nota, vi sono:

1. divisione di X3D in componenti, che consente sia di utilizzare un sottoinsieme delle funzionalità offerte quanto più calzante con i requisiti, sia di introdurre in maniera graduale e pulita nuovi componenti garantendo un rapido supporto alle nuove tecnologie
2. miglioramento dell'interfaccia di scripting
3. miglioramento della specifica che definisce il comportamento runtime di X3D, garantisce che una scena X3D si comporti in maniera prevedibile e uniforme su diversi browser/player

[5]L'utilizzo di un formato XML-based è di fatto una possibilità in quanto la specifica, proprio per garantire la retrocompatibilità, ammette l'utilizzo del formato VRML-based

X3D viene ratificato come standard ISO nel 2004[6]. Sebbene abbia suscitato, insieme a VRML, l'interesse di un gran numero di utenti e di aziende e goda tutt'ora di una certa popolarità, non ha mai conosciuto il successo sperato. Secondo molti osservatori, il problema era l'assenza di una qualsiasi offerta al pubblico. In sintesi, al navigatore medio non interessava affatto volare in mondi 3D: ciò che gli interessava era semplicemente l'informazione[7]. Va aggiunto poi che gli strumenti di navigazione quali i player X3D erano spesso distribuiti come plugin per i browsers o in forma standalone, essendo queste le uniche possibilità. Mancava dunque quella parte di "vera" integrazione con la pagina web, che avrebbe consentito all'utente di accedere con immediatezza e facilità ai contenuti 3D, senza dover passare per complessi rompicapo quali la scelta e l'installazione di uno dei tanti player messi a disposizione. Infine, ma non per ultimo, va detto che a limitare la diffusione di X3D ha contribuito anche l'adozione di un modello di sviluppo chiuso e proprietario dei sistemi di playback, di editing e di authoring. Come spesso accade (tralasciando la quasi totale assenza di prodotti cross-platform) molti di questi software non sono sopravvissuti alla scomparsa dell'azienda produttrice, lasciando i propri utenti "con un pugno di mosche in mano".

Mentre X3D viveva la sua "crisi", un'altra tecnologia si stava facendo avanti, un nuovo standard che avrebbe potuto riportare X3D alla gloria di un tempo.

[6]ISO/IEC 19775/19776/19777

[7]Guida a VRML. In *HTML.it*. Retrieved 11:00, August 27, 2011, from `http://xml.html.it/guide/lezione/1812/il-3d-sul-web-storia-e-futuro/`

1.2 Un nuovo standard all'orizzonte: HTML5

Molto, forse troppo, ci sarebbe da dire sulla nascita e l'evoluzione di HTML. Ci si limiterà in questo contesto a ripercorrere brevemente la sua storia, soffermandosi su quei cambiamenti e punti di svolta degni di nota, per poi dedicare ampio spazio a quelle caratteristiche fondamentali per l'evoluzione del 3d nel web.

HTML nasce nel 1989 ad opera di Tim Berners-Lee, oggi ritenuto uno dei padri fondatori del World Wide Web. Berners-Lee cercava uno strumento per descrivere in maniera efficiente e semplice ipertesti, ovvero le future pagine web. Quello che fece fu scrivere la primissima specifica[8] dell'Hypertext Markup Language e il software che la implementava. Da qui, come si usa dire, il resto è storia.

Dopo oltre 20 anni, siamo giunti ad HTML5[9], quinta (ma non ultima!) revisione di quella prima embrionale specifica, ancora in corso di definizione (in gergo *draft*). Quali sono i suoi obiettivi? Come ricorda il W3C[10] stesso nella documentazione di HTML5, *"The main area that has not been adequately addressed by HTML is a vague subject referred to as Web Applications. This specification attempts to rectify this, while at the same time updating the HTML specifications to address issues raised in the past few*

[8] La prima informale specifica era un ristretto documento denominato *HTML Tags*, `http://www.w3.org/History/19921103-hypertext/hypertext/WWW/MarkUp/Tags.html`

[9] HTML5 Specification. In *HTML5 Editor's Draft*. Retrieved 12:20, August 29, 2011 from `http://dev.w3.org/html5/spec/Overview.html`

[10] Il World Wide Web Consortium (W3C) è un'associazione internazionale che si occupa di definire standard aperti per il Web e di assicurarne la crescita ed l'interoperabilità. `http://www.w3.org/Consortium/mission.html`

years". HTML5 fa da "capello" ad un insieme di tecnologie pensate per agevolare la creazione di web application.

In primo luogo il linguaggio è stato arricchito di nuovi elementi per migliorare la semantica delle pagine. I tag `<nav>`, `<section>`, `<article>`, `<header>` e `<footer>`,ad esempio, individuano parti ben precise e immediatamente riconoscibili di un documento HTML5. Il markup è stato snellito con l'eliminazione dei molti elementi deprecati in HTML 4.01 e sono stati introdotti miglioramenti per quanto riguarda le regole di parsing. Vi sono poi i tag `<audio>` e `<video>`, che consentono l'inserimento di contenuti multimediali, senza l'ausilio di plugins esterni, all'interno della pagina. Questi elementi sono parte del DOM e dunque possono essere liberamente modificati e gestiti grazie a JavaScript. Sono state poi aggiunte numerose API per semplificare la vita dello sviluppatore, come WebStorage, un'interfaccia di programmazione, evoluzione dei cookies, per la persistenza dei dati lato client, opure WebSockets, un'API che consente di creare una connessione full-duplex tra client e server.

Per quanto riguarda questo progetto, però, le novità più importanti sono sicuramente quelle introdotte nel campo della grafica e del disegno all'interno della pagina web. Oltre ad avere integrato pienamente la specifica Scalable Vector Graphics (SVG) per la gestione di immagini ed elementi in grafica vettoriale, è stato introdotto un nuovo elemento per il disegno, il tag `canvas`. Quest'ultimo consente di definire un'area da utilizzare come una tela, un foglio sul quale disegnare. Il codice Javascript consente di accedere all'oggetto Canvas e, da questo, ci permette di recupera un oggetto di tipo

Context. Un "contesto" è un oggetto che espone un'API per il disegno. Un Canvas può fornire molteplici Context. L'esempio seguente mostra l'utilizzo del Context "2d", l'unico per ora formalmente definito nella specifica HTML5. Il risultato è mostrato in Figura 1.1.

Listing 1.1: Draw into canvas using 2d Contex

```
<!DOCTYPE html>
<html>
<head>
<title>Canvas Examples</title>
<script type="text/javascript">
window.onload = function () {
    var draw = function (c) {
        c.fillStyle = "red";
        c.fillRect(50, 100, 100, 100);
        c.fillStyle = "green";
        c.fillRect(150, 100, 100, 100);
        c.fillStyle = "blue";
        c.fillRect(250, 100, 100, 100);
    };
    var canvas = document.getElementById("paint");

    if (canvas.getContext) {
        var context = canvas.getContext("2d");
        draw(context);
    } else {
        // fallback
    }
};
</script>
</head>
<body>
    <h1>Canvas</h1>
    <canvas id="paint" width="400px" height="300px" style="border: 1px
        solid">
    </canvas>
</body>
</html>
```

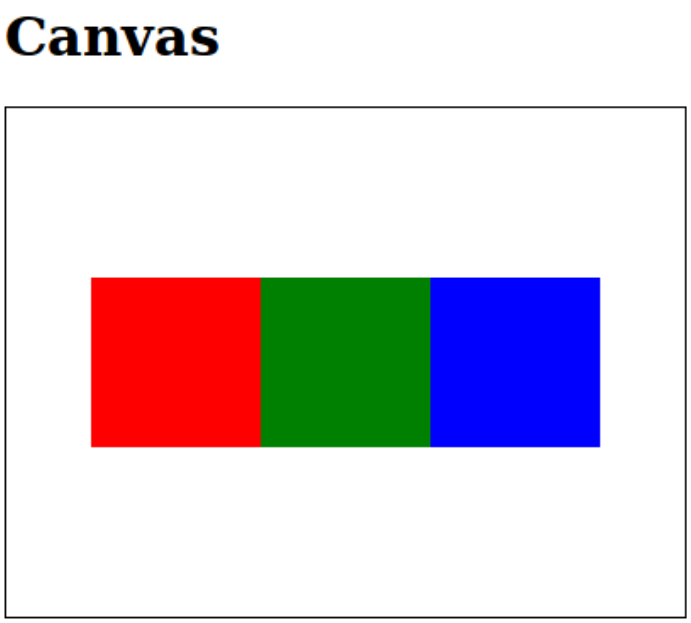

Figura 1.1: Tre rettangoli disegnati nel canvas

Il codice del listato 1.1 è decisamente semplice e autoesplicativo. Il contesto viene recuperato attraverso il metodo `getContext(2d)` dell'oggetto canvas. Qualora questo metodo non sia presente significa che il browser in uso non supporta il tag `<canvas>` e si può dunque scatenare un meccanismo di fallback (ad esempio sostituendone le funzionalità con Flash o simili). L'oggetto Context espone quella che viene definita come la Canvas2D API, dei semplici metodi per il disegno, utilizzati in questo caso per riempire il canvas con tre rettangoli dei colori specificati.

Sebbene il Context "2d" sia il solo definito nella specifica, viene riportato tra i contesti supportati anche "webgl", il cui obiettivo è quello di fornire un'interfaccia di programmazione per il 3D.

1.3 WebGL: il 3d (realmente) nel web

Le prime sperimentazioni di un context 3D per l'elemento canvas iniziano nel 2006 all'interno dei laboratori Mozilla ad opera di Vladimir Vukićević[11]. Qualche anno più tardi, nel 2009, visto l'interesse suscitato fu istituito il WebGL Working Group. Ad oggi è uno standard cross-platform, cross-browser e royalty-free gestito dal consorzio Khronos Group, responsabile anche di altre importanti tecnologie in campo 3d quale, ad esempio, OpenGL. Nel Marzo 2011 è stata rilasciata la versione 1.0 della specifica[12].

WebGL, acronimo di Web-based Graphics Library, è un'interfaccia di programmazione che consente di creare e gestire contenuti 3d all'interno della pagina web. Per mezzo del tag `<canvas>` espone una DOM API di basso livello, utilizzabile, dunque, attraverso qualsiasi linguaggio "DOM-compatibile" (vedi Javascript, Java). Queste caratteristiche consentono di mettere da parte i plugin di terze parti (spesso soluzioni proprietarie) e di accedere ad un 3d nativo del web. Un browser WebGL compatibile si occuperà di fornire l'accelerazione grafica sfruttando l'hardware presente sulla macchina, interfacciandosi direttamente con i driver, con risultati performanti di alta qualità.

Non a caso infatti WebGL poggia su due tecnologie, consolidate e ampiamente diffuse, quali OpenGL ES2, API per la grafica 2D e 3D in sistemi embedded (console, telefono, veicoli, etc...), e l'OpenGL Shading Language

[11] Corso su WebGL. In *HTML5Today*. Retrieved 12:10, August 14, 2011 from `http://www.html5today.it/tutorial/corso-webgl--introduzione`

[12] WebGL Specification. In *Khronos Group*. Retrieved 9:20, July 29, 2011 from `https://www.khronos.org/registry/webgl/specs/1.0/`

Figura 1.2: Cubo con texture in un canvas renderizzato con WebGL

(GLSL) per la creazione di "Shader"[13]. L'utilizzo di queste tecnolgie fa di WebGL un'interfaccia di basso livello e sebbene da una parte questo porti enormi benefici sul piano dell'efficienza e della qualità, dall'altra si introduce un'alto livello complessità, anche per la gestione di contenuti semplicissimi come quelli in Figura 1.2: *"WebGL is a low-level API, so it's not for the faint of heart. OpenGL's shading language, GLSL, is itself an entire programming environment. As a result, even simple things in WebGL take quite a bit of code. You have to load, compile, and link shaders, set up the variables to be passed in to the shaders, and also perform matrix math to animate shapes."*. Il bagaglio di conoscenze richieste per utilizzare questa tecnologia è enorme come lo sono, di certo, lo sforzo e il tempo richiesti per apprenderle, malgrado questo sia poi il compito di uno sviluppatore.

In conclusione WebGL non è uno strumento "ready-to-use" e questo potrebbe non giocare a suo favore. Tuttavia, stanno già nascendo librerie di più alto livello che fungono da "wrapper" e agevolano la vita al programma-

[13]Gli shader sono insiemi di istruzioni, altamente flessibili e riutilizzabili, che definiscono in ogni parte l'aspetto finale che l'oggetto avrà dopo la fase di rendering.

tore[14], mascherando la complessa natura di WebGL e consentendogli dunque di concentrarsi pienamente sulla creazione di contenuti per web. Tra queste vi è sicuramente X3DOM, un framework che tenta di unire il vecchio X3D, consolidata tecnologia di ieri, senza dubbio "user-friendly", con il giovane e complicato WebGL.

1.4 X3DOM

Quanto visto finora con WebgGL consente di introdurre e gestire elementi di grafica 3d nel contesto web utilizzado un paradigma di programmazione imperativo. Ciò che manca è un modo semplice per integrare in maniera dichiarativa contenuti 3d all'interno del DOM di una pagina HTML. E' proprio quello che il framework X3DOM cerca di fare, recuperando un percorso già iniziato dall'X3D Working Group per una piena integrazione di X3D con HTML5. L'obiettivo finale è quello di incorporare una scena X3D all'interno del DOM, come già avvenuto per SVG, consentendone la manipolazione semplicemente interagendo con i nodi stessi e grazie al supporto degli eventi HTML per i nodi stessi.

Oltre però ad integrare X3D e HTML5 e necessario poi renderizzare in qualche modo questi contenuti per renderli visibili all'utente finale. Il framework opensource X3DOM

X3DOM è un framework sperimentale ed open source il cui scopo è quel-

[14]GLGE Official Website. In *GLGE*. Retrieved 7:22, August 18, 2011 from http://www.glge.org/

lo di cercare di integrare l'HTML5 con contenuti 3D dichiarativi. La linea seguita durante lo sviluppo è quella di soddisfare le correnti specifiche dell'HTML5 riguardo i contenuti 3D dichiarativi[23] e permettere l'inclusione di elementi X3D come parte di un qualsiasi albero DOM HTML5. X3DOM è basato principalmente su due tecnologie: X3D e HTML5. La scelta dell'utilizzo dell'X3D come modello per lo scene-graph si fonda principalmente su tre motivazioni:

1. X3D è uno standard ISO maturo e consolidato, che già definisce la codifica XML.

2. Le specifiche HTML5 già fanno un uso esplicito dell'X3D per scene 3D dichiarative, anche se stanno in una singola riga:
   13.2 Declarative 3D scenes
   Embedding 3D imagery into XHTML documents is the domain of X3D, or technologies based on X3D that are namespace-aware.
   Non specificano però come l'integrazione dovrebbe avvenire
   e cosa più importante, come accedere ai contenuti dello scene-graph.

3. Esiste già un'interfaccia al DOM tree descritta nel X3D-Programming Language Binding Interface [19]. Come per il punto precedente manca un meccanismo di aggiornamento live dei contenuti dell'albero DOM.

Per lo stesso motivo, l'HTML5 viene utilizzato poiché già utilizza l'X3D per scene 3D dichiarative. Tuttavia, come già detto precedentemente, l'HTML5 non specifica come integrare e accedere ai contenuti dello scene-graph. L'intento è quello di estendere il modello di per produrre delle immagini in

place, piuttosto che all'interno di un plugin aggiuntivo. I nodi X3D sono direttamente integrati con gli elementi DOM e il sottosistema X3D viene principalmente utilizzato per effettuare un render della scena. L'intera manipolazione dello scene-graph avviene servendosi delle interfacce di scripting di un qualsiasi browser standard basato su DOM, come in un tradizionale documento DHTML. Nei paragrafi successivi verranno trattate le componenti principali dell'architettura alla base delll'X3DOM, mostrata anche in figura.

1.4.1 Connettore

Il connettore è il cuore di questa architettura. Consente di connette il frontends del browser con il backends dell'X3D e supporta dei meccanismi per comunicare i cambiamenti nella rappresentazione DOM o in quella X3D. L'architettura non usa direttamente l'albero DOM per il rendering della scena ma crea un albero X3D sincronizzato con l'altro. Il connettore include anche un DOM/X3D adapter per supportare differenti backends e frontends. Gli sviluppatori hanno previsto la presenza di almeno due frontends adapter per Firefox e WebKit e due backends adapter per differenti X3D runtimes (Instant Reality, FreeX3D). La corrente implementazione di X3DOM è basata sulle WebGl, mentre beckends alternativi come plugins X3D già esistenti o Flash saranno implementati successivamente. Nella figura è mostrato il corrente stato di sviluppo del progetto. Il frontends adapter deve essere in grado di accedere direttamente al contenuto dell'albero DOM. Non dovrebbe leggere e analizzare un XML-data-stream, ma leggere e scrivere la rappresentazione del DOM. Di conseguenza il backends adapter ha accesso diretto al contesto X3D runtime che include la scena X3D, la quale riflette l'albero DOM. Il compito principale del connettore è mantenere entrambe le rappresentazioni sincronizzate e distribuire i cambiamenti in entrambe le direzioni.

1.4.2 Model Update

Qualsiasi aggiornamento sull'albero DOM, come la creazione, la modifica e la rimozione di elementi DOM, deve riflettersi sull'albero X3D, usando il backends adapter. Ciò implica che gli elementi DOM rappresentano direttamente nodi X3D, ma anche strutture addizionali come le Route X3D.

1.4.3 ObserverResponse

A seconda del profilo supportato il modello in esecuzione (execution-model) può creare dei cambiamenti nell'albero X3D in base al tempo o all'azione dell'utente. Il connettore deve essere in grado di distribuire questi cambiamenti indietro nella rappresentazione DOM. Questo sarà reso possibile connettendo l'Observer con specifici elementi dell'albero X3D che distribuiranno questi cambiamenti.

1.5 Innovazione introdotta da X3DOM

1.5.1 Assenza di plugin

I sistemi basati su plugin hanno due grandi inconvenienti. Primo, i plugins non sono installati di default sulla maggior parte dei sistemi. Di conseguenza, l'utente si deve occupare dell'installazione del plugin e dei problemi di sicurezza e di incompatibilità con il browser o il sistema operativo. In sec-

ondo luogo, questo tipo di sistemi definisce un'applicazione ed un modello di eventi all'interno del plugin che è disaccoppiato dal contenuto DOM della pagina web.

1.5.2 Programmazione dichiarativa

Grazie all'HTML5 esistono già tecnologie per la visualizzazione di scene 3D come canvas3D. Tuttavia quest'ultima è un'interfaccia di basso livello, dove il programmatore JavaScript deve manipolare direttamente matrici 4x4, al contrario di quello che avviene utilizzando l'X3D per la descrizione di scene 3D.

1.5.3 DOM

La modifica di un nodo all'interno dello scene-graph è molto semplice: al nodo si accede infatti come ad un qualsiasi nodo del DOM del browser. Esempio:
Codice HTML/X3D:

```
[...]
<group id="root"></group>
[...]
```

HTML-Script per l'aggiunta di un nodo:

```
root = document.getElementByid("id");
```

```
trans = document.createElement('Transform');
trans.setAttribute("translation", "1 2 3" );
root.appendChild(trans);
```

HTML-Script per la rimozione di un nodo:

```
document.getElementById("root").removeChild(trans);
```

Capitolo 2

L'editor Medea

Conclusioni

A conclusione del lavoro di Tesi, possiamo dire che gli obiettivi che ci si era preposti all'inizio delle attività sono stati in gran parte raggiunti. Infatti, è stato realizzato un editor che consente la creazione di scene 3D adottando un approccio innovativo: l'X3DOM. Grazie a questa nuova tecnologia è stato possibile sviluppare un'applicazione web facile da installare, multipiattaforma e facile da utilizzare da parte anche dell'utente, anche meno esperto.

Nel secondo capitolo è stato mostrato come l'impiego dell'X3DOM ha portato notevoli vantaggi allo sviluppo dell'applicazione, semplificando molto l'interazione con la scena X3D, anche se l'adozione di questa tecnologia, recentemente introdotta, richiede una versione del browser usata dagi sviluppatori, in quanto la versione stabile del browser non supporta X3DOM. La scelta di utilizzare una tecnologia ancora in fase di sviluppo è stata comunque ampiamente ripagata dai vantaggi originati dalla sua adozione.

Nel terzo capitolo è stato poi mostrato come è possibile creare un esercizio funzionante utilizzando l'editor esposto nel secondo capitolo.

Bibliografia

[1] D. Brutzman & L. Daly X3D: extensible 3D Graphics for Web Authors. Morgan Kaufmann publishers, (2007).

[2] Adobe flash: http://www.adobe.com/products/flashplayer/.

[3] http://www.adobe.com/

[4] 3d ccs-transforms for the webkit. http://webkit.org/specs/CSS\-VisualEffects/CSSTransforms3D.html

[5] Canvas 3d js library. http://www.c3dl.org/

[6] Canvax3d. https://labs.mozilla.com/forum/comments.php?DiscussionID=363.

[7] Silverlight. http://www.microsoft.com/SILVERLIGHT/

[8] Directx. http://msdn.microsoft.com/directx/

[9] Ecma-262, ecmascript language specification. http://www.ecmainternational.org/publications/standards/Ecma-262.htm

[10] O3d; an javascript based scene-graph API. http://code.google.com/apis/o3d/

[11] Google chrome experiments. http://www.chromeexperiments.com/

[12] Opengl. http://www.opengl.org/documentation/

[13] Opengl. http://www.opengl.org/documentation/

[14] Opengles. Khronos Group. http://www.khronos.org/opengles/

[15] Taking the canvas to another dimension. http://my.opera.com/timjoh/blog\-/2007/11/13/takingthe-canvas-to-another-dimension

[16] Canvas 3D: Gl power, webstyle. http://blog.vlad1.com/2007/11/26/canvas-3d-gl-powerweb-style/

[17] X3D. http://www.web3d.org/x3d/specifications/

[18] Scene access interface(SAI), ISO/IEC CD 19775-2 ed. 2:200x. http://www.web3d.org/x3d/specifications/ISO-IEC-CD-19775-2.2-X3D-SceneAccessInterface/

[19] X3D Specification http://www.web3d.org/x3d/specifications/

[20] Cascading style sheets. http://www.w3.org/Style/CSS/

[21] HTML 5 specification, canvas section. http://dev.w3.org/html5/spec/Overview.html\#the-canvaselement

[22] Scalable vector graphics. http://www.w3.org/Graphics/SVG/

[23] Declarative 3D scene in HTML5. http://www.w3.org/TR/html5/no.html\#declarative-3d-scenes

[24] WebGL Official Website http://www.khronos.org/webgl/

[25] Corso su WebGL http://www.html5today.it/tutorial/corso-webgl--introduzione

Appendice A

Codice