UNIVERSIDADE FEDERAL DO PARANÁ
UFPR - DEPARTAMENTO DE INFORMÁTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

INTERFACE WEB UTILIZANDO HTML5

FERNANDO CESAR OTT FILHO

ORIENTADOR: ANDRÉ LUIZ PIRES GUEDES

CURITIBA
2011

FERNANDO CESAR OTT FILHO

INTERFACE WEB UTILIZANDO HTML5

Trabalho de Graduação que visa realizar um estudo sobre os novos recursos da linguagem HTML5, abordando suas novas funcionalidades e validação de seu potencial.

Orientador:
André Luiz Pires Guedes

Curitiba

2011

Lista de Figuras

Sumário

1 Introdução

Se resumíssemos o principal fator que impulsionou a internet a transformar-se em uma rede utilizada por milhões e indispensável no dia a dia poderíamos apenas dizer HTML (Linguagem de marcação de hipertexto). A importância do uso da linguagem HTML encontra-se no seu uso larga escala, por ter alcançado uma eficiente solução para organizar e conectar as páginas web de maneira intuitiva ao internauta. Somente após sua invenção do HTML que a rede www (World Wide Web – sistema de documentos interligados através de hipertexto acessado através da internet) se estendeu dos laboratórios de pesquisa e ganhou o gosto do público final tomando as proporções que possui hoje. Neste contexto, este trabalho procura mostrar a evolução desta linguagem em um estudo detalhado sobre a nova versão do HTML, o HTML5, que já se encontra em uma versão de testes disponibilizado pela W3C (World Wilde Web Consortium). Inicialmente será mostrada a evolução da web juntamente com a linguagem HTML e sua fundamental importância na popularização da rede www. Após, serão descritos as novas funcionalidades e os recursos que o novo HTML traz, através de casos de sucesso que já fazem uso da nova linguagem. Também foram adicionados trechos de código que apresentam uma noção da robustez que a linguagem tomou da versão anterior para a nova. Junto à proposta de estudo, procuramos validar o HTML5 através da criação de uma interface gráfica web para visualização e edição de grafos utilizando apenas HTML5 e JavaScript em sua camada de aplicação. Essa interface se mostra uma alternativa aos Applets Java, Flash e Silverlight para construções gráficas.

2 Fundamentação Teórica

2.1 A evolução da interface web

A primeira especificação pública do HTML foi feita por Tim Berners-Lee (inventor da web) em 1991 em um documento chamado “HTMLtags”, sendo mantido sem custo de licença desde então pela W3C, empresa fundada por Berners-Lee visando criação de normas e recomendações para melhorar a qualidade da web.

Após a especificação do HTML, o primeiro browser baseado nesta tecnologia que conseguiu ser disseminado em larga escala na rede foi o Mosaic, desenvolvido na NCSA (National Center for SupercomputingApplications), lançado em 1993. Por ser o primeiro a possuir uma interface fácil e intuitiva deve-se a ele a popularização da internet. Como é possível observar na figura 2.1 , os principais conceitos de interface desenvolvidos pelo no Mosaic foram tão marcantes que os atuais browsers líderes de mercado (Internet Explorer, Firefox, Chrome) agregam fortemente algumas de suas características.

Figura 2.1: Mosaic - Primeiro Browser Popularizado

Também na década de 90, surgem às entidades controladoras de endereços IP (Internet Protocol) e nomes (DNS – Domain Name System) pela Internet, browsers concorrentes (Internet Explorer, Netscape) e os primeiros buscadores (WebCrawler, Lycos, Yahoo, Altavista, Google), mostrando que a internet poderia ser usada para uso comercial. Estes avanços trouxeram uma popularização exponencial à web que se via obrigada a evoluir visando melhorar a usabilidade do internauta. Este foi o período da Web 1.0 (unidirecional, ou seja, o usuário não interagia com os programas). O próximo avanço virtual seria o inicio da interação usuário – página web, a chamada Web 2.0 que possibilitou a criação de plataformas interativas, como as redes sociais, blogs, compartilhadores onde a geração de conteúdo é fomentada pelo próprio usuário. A Web 2.0 trouxe consigo as primeiras páginas assíncronas (mais de uma requisições ao servidor podem ocorrer simultaneamente), com um modelo de páginas rápidas, com navegação mais agradável e intuitiva ao usuário. Para chegar a este resultado, a Web 2.0 faz intenso uso de pacotes multimídia, Flash, JavaScript e AJAX (asynchronous JavaScript and XML). O uso massivo destas tecnologias e todos os recursos necessários para atender a demanda de uma melhor navegabilidade, foram desenvolvidos utilizando como base o HTML. O curioso é que dentro de tanta inovação, a base de tudo continuou estagnada. As versões posteriores à primeira especificação visava aprimorar o que já existia, com foco quase nulo no sentido de implementarem novas funcionalidades. É neste contexto que o HTML5, tardiamente surge no mercado. Trazer uma base mais sólida e com recursos já embutidos para suportar o refinamento crescente das aplicações web.

2.2 O HTML5

Em 2008 foi anunciada a primeira especificação do HTML5 pela W3C. Além da W3C, outros grupos importantes têm auxiliado e acelerado a construção da sua nova especificação, como Google e Opera. A versão deve ser lançada somente em 2012. O HTML5 foi lançado com a finalidade de facilitar a compreensão e a manutenção do código. Desde 1999 o desenvolvimento da linguagem HTML estava estacionado na versão 4. Neste período, a W3C passou a trabalhar no desenvolvimento do XHTML, um HTML baseado em XML. A nova especificação do HTML ainda esta em desenvolvimento. A proposta era melhorar alguns pontos e adicionar novas funcionalidades e/ou melhorar algumas existentes. Um dos pontos importantes é o dinamismo e segmentação das páginas, que ganhará grande relevância nas otimizações para SEO (Search engine optimization – incentivo ao uso de padrões de boa prática no desenvolvimento web, através de maior relevância da página

web nos mecanismos de busca). Existe também uma mudança no papel do HTML que tenta englobar funcionalidades antes só possíveis através de outras linguagens incluídas ao código HTML padrão (ex.: Flash, Applets Java, Silverlight). A atual e bem sucedida disseminação do HTML5 como padrão entre os browsers e o incentivo de grandes players (ex.: Apple, Google) trazem boas expectativas quanto esta nova linguagem, que ainda se encontra em uma versão de teste. Entre as principais melhorias e novos recursos do HTML5, segundo a W3C, encontramos:

- Melhoria do armazenamento off-line

  O HTML5 contará com uma estrutura muito mais avançada de armazenamento off-line do que o existente atualmente no HTML4, como o cookie. O HTML5 permitira um armazenamento de até 5mb, um aumento significativo comparado ao que os cookies atualmente permitem. Esta nova solução de armazenamento contará com scripts cliente-side que fará com que seja possível excluir o acesso server-side, ou seja, ou servidores web não poderão escrever diretamente no armazenamento local, o que irá fornecer páginas web muito mais rápidas e reduzir a quantidade de banda necessária para acessar o website. As duas principais formas de armazenamento incluídas são a de sessão e a local. Existirá um armazenamento local por website estando disponível para todos os scripts de outros websites já armazenados no mesmo. Já o armazenamento por sessão será limitado ao período de tempo que a atual página estiver aberta, permitindo ainda sessões diferentes para páginas diferentes, recurso inexistente no HTML4.

- Inclusão de pacote multimídia

  Uma das principais carências do HTML4 se caracterizava pela falta de players para vídeo e áudio. Antes das novas tags vídeo e áudio do HTML5, as soluções encontradas para incluir vídeos em páginas web eram, ou utilizar serviços terceiros (ex.: youtube), ou incorporar à página um player, geralmente proprietário (ex.: flash). A primeira opção acabava sendo a mais utilizada. Incluir um player acabava envolvendo também a criação de um layout padrão, custo adicional para garantir qualidade de vídeo e áudio além de não ser a melhor opção em termos de boas práticas de construção web.

Vídeo

Toda esta dificuldade encontrada pode ser resolvida com o HTML5 utilizando apenas uma linha de código.

<video src=”movie.avi” poster=”movie.jpg”>Alt text</vídeo>

Tabela 2.1: Adicionar vídeo utilizando tag HTML5.

A tabela 2.2 irá listar os atributos principais de configuração da tag vídeo.

audio	Define o estado inicial padrão que o audio do vídeo deve apresentar ao carrega-lo
autoplay	Quando presente, o vídeo será iniciado assim que estiver pronto.
loop	Se presente, o vídeo voltara a ser executado a partir do inicio, sempre quando terminado.
preload	Se presente, o vídeo é carregado durante juntamente com o carregamento da página.

Tabela 2.2: Atributos da tag vídeo do HTML5.

Audio

Da mesma maneira que a tag vídeo, o HTML5 traz um player padrão na nova especificação.

<audio src=”musica.mp3” >texto caso não tenha suporte à HTML5</audio>

Tabela 2.3: Adicionar audio utilizando tag HTML5.

Até esta nova versão do HTML não existia players de áudio para web, forçando sempre a inclusão de players proprietários (ex.: Quicktime, Media Player). Os principais atributos da tag áudio, (autoplay, loop, preload) possuem comportamentos similares aos da tag vídeo.

- API's para desenvolvimento gráfico

A partir desta API desenvolvida, o HTML entra também em um novo ramo de atuação, provendo suporte gráfico em páginas HTML5, que até então só poderiam existir com a inclusão de aplicativos desenvolvidos em outras linguagens como Flash, SilverLight e Java Applet. O elemento principal é o canvas. No próximo tópico estudaremos com mais detalhes esta poderosa ferramenta.

Além dos elementos existentes no HTML4, novos elementos foram incluídos. Alguns deles serão descritos na tabela 2.4.

Novos elementos de Media	
<audio>	Comentado na Tabela 2.3
<video>	Comentado na Tabela 2.3
<embed>	Utilizado para incorporar conteúdo, como plug-ins.
Elemento canvas(elementos gráficos)	
<canvas>	Visualizador de elementos gráficos. Saiba mais no próximo capitulo.
Novos elementos de formulário	
<datalist>	Deve ser combinado a um elemento de input, usado para definir listas de opções.
<keygen>	Gera um par de chaves para ser utilizado em formulários. Quando o formulário é enviado, uma chave privada é armazenada localmente, e a pública enviada para o servidor.
<output>	Representa o resultado de um cálculo. Em seu atributo 'onforminput', referenciamos os campos input desejados e as operações que queremos realizar entre eles.
Novos tipos de <input>	**Novos atributos do <input>**
Tel	Autocomplete
Search	Autofocus
url	Formaction
Email	Formenctype
Datetime	Multiple
Date	Required
Month	Step
Week	Pattern
Time	Min/max
Datetime-local	List
Number	Formtarget
Range	Formnovalidate

Tabela 2.4: Novos elementos do HTML5.

2.3 Canvas

2.3.1 Conhecendo melhor o Elemento Canvas

O HTML5 entre seus novos recursos traz o canvas, um elemento fácil e poderoso para desenhos gráficos com a utilização do JavaScript.

Após a instanciação do Canvas este disponibiliza um contexto que é enxergado pelo

JavaScript, possibilitando assim sua manipulação. A grande maioria dos browsers só dá suporte ao 2Dcanvas, porém existem alguns algoritmos já implementados para construir o efeito 3D para as aplicações gráficas criadas no canvas. A API 2D do canvas é bastante intuitiva e de fácil entendimento aos já familiarizados ao JavaScript. Para utilizarmos o canvas, primeiro precisamos coloca-lo na página HTML5. A seguir encontra-se um exemplo de código HTML para a sua adição.

<canvas id=”canvas”width=”760”height=”500”.> Este browser não possui suporte a HTML5 </canvas>
Na parte interna do canvas colocamos um texto default caso o browser não tenha suporte ao mesmo. Toda renderização é feita via JavaScript, fazendo uso da API canvas.

Tabela 2.5: Adicionar canvas

Após inserido o elemento canvas na página, para manipula-lo via JavaScript, precisamos ter acesso ao contexto de renderização do canvas. Com este contexto podemos invocar métodos distintos para acessar as funções de desenho. A partir daí, todo trabalho

var canvas = document.getElementById(”canvas”); var ctx = canvas.getContext(”2d”);
O objeto JavaScript document, contem todos os objetos do cabeçalho e corpo do Html. Ele possui o método getElementById. Através dele conseguimos recuperar o elemento Html através do id do mesmo. Acima, a variável canvas recebe o objeto canvas e na linha seguinte, a variável ctx recebe o contexto para renderização 2d.

Tabela 2.6: Manipular o canvas

fica por conta da manipulação dos objetos, propriedades e atributos da API canvas pelo JavaScript. A tabela 2.7 mostra um exemplo de código JavaScript para desenhar uma reta no Canvas.

ctx.beginPath();
ctx.moveTo(x1, y1);
ctx.lineTo(x2, y2);
ctx.stroke();
ctx.closePath();

Ctx é o contexto do canvas. X1,y1,x2,y2, são as coordenadas de onde até a onde desenharemos a reta.

Tabela 2.7: Desenhar retas no Canvas

Na tabela 2.8 é mostrado através de código JavaScript como é possível escrever texto no Canvas.

ctx.beginPath();
ctx.font = ”10pt Arial”;
ctx.fillStyle = ”#000000”;
ctx.fillText(”UFPR - Fernando Ott - 20071810”, WIDTH, HEIGHT);
ctx.closePath();
ctx.fill();

Definimos a fonte, a cor da fonte e o texto, com os dois parâmetros sendo seu tamanho. O fill() desenha na tela.

Tabela 2.8: Escrever texto no Canvas

Para desenhar círculos, segue um exemplo de código na tabela 2.9.

ctx.beginPath();
ctx.arc(x, y, w, 0, Math.PI * 2, true);
ctx.closePath();
ctx.fill();
ctx.stroke();

Ctx é o contexto. X,y são as coordenadas de onde desenharemos o círculo no canvas, w o raio, 0 a rotação e o Math.PI * 2 para criarmos um círculo inteiro.

Tabela 2.9: Desenhar círculos no Canvas

2.3.2 O Potencial do HTML5 + Canvas

Com a API canvas, o HTML5 entra na concorrência contra o Flash e SilverLight. A API permite a construção de jogos, aplicações sofisticadas, interfaces multimídia como seus concorrentes, mas com um grande diferencial por ser leve, algo fundamental para as aplicações web atuais. Abaixo serão mostrados exemplos do potencial que pode ser feito utilizando o elemento Canvas.

A figura 2.2 mostra a visualização por satélite 3d de planetas.

Outro exemplo de uso é na Industria de Jogos. A figura 2.3 mostra a tela do jogo Quake II, portado totalmente para web pela equipe do Google, com todos os recursos HTML5, áudio, vídeo, armazenamento locais e gráficos.

Esses exemplos mostram assim a gama de funcionalidades que podem ser criadas a partir deste novo elemento fornecido pelo HTML5. Outros incentivadores para a utilização

Figura 2.2: Visualização por satélite 3D de planetas

Figura 2.3: Jogo Quake II

em maior escala do HTML5 é a Apple que já não apresenta suporte flash em seus novos produtos e o Google, um dos maiores incentivadores, que já apresenta uma alternativa beta de vídeos em HTML5 para o youtube.

2.4 JavaScript + HTML5

Para utilização de todos os recursos do HTML5, é necessário o uso em larga escala de JavaScript. Sem o conhecimento desta linguagem não é possível trabalhar com os novos recursos disponíveis. O JavaScript é uma linguagem de programação cliente-side para navegadores web, concebida para ser uma linguagem script com orientação a objeto baseada em protótipos, funções de primeira classe e tipagem fraca e dinâmica. Possui suporte à programação funcional e apresenta recursos como fechamentos e funções de alta ordem comumente indisponíveis em linguagem mais populares como Java e C++. Originalmente desenvolvido por Brendan Eich da Netscape sob o nome de Mocha, logo após

já em seu lançamento oficial em 1995 foi renomeado para LiveScript. Seu nome atual vem de uma estratégia de marketing aproveitando-se da grande popularidade que o Java dispunha na época o causou certa confusão, dando a impressão de que a linguagem foi baseada em Java. A linguagem adquiriu rapidamente uma ampla aceitação do mercado como linguagem de script cliente-side de páginas web. Após alguns anos estacionada, com o advento do Ajax, o JavaScript teve sua popularidade de volta e passou a receber mais atenção profissional o que resultou na criação de inúmeros frameworks e bibliotecas, melhores praticas de programação e sua utilização fora do ambiente web, como o uso de plataformas de JavaScript server-side. Atualmente o JavaScript é baseado na linguagem de programação ECMAScript padronizada pela Ecma International nas especificações ECMA-262 e ISSO/IEC 16262 e é atualmente a principal linguagem de programação cliente-side em navegadores web. Algumas características da linguagem JavaScript especificadas na ECMA-262:

- **Sintaxe**
  A linguagem JavaScript herda suas construções principalmente da linguagem Java (que os havia herdado das linguagens C e C++).

- **Linguagem interpretada**
  Não existe a necessidade de compilação como em outras linguagens. Os comandos JavaScripts são executados diretamente por um interpretador. A função de interpretar o JavaScript é do navegador, que é executado a medida em que encontrar os comandos JavaScript dentro dá página HTML.

- **Plataforma**
  Como o JavaScript é uma linguagem interpretada, diferente das outras linguagens compiladas que geram executáveis compatíveis apenas com a plataforma que foi gerado, a linguagem pode ser interpretada por qualquer browser que o suporte independente da plataforma que o browser está sendo rodado.

- **Linguagem baseada em objetos**
  É dada a expressão "baseada em objetos" e não orientada a objetos ao JavaScript pois ela já possui vários objetos embutidos por padrão, como objetos para trabalhar com datas e horas já pronto para ser usado.

- **Eventos**
  A notação de eventos é uma das características mais importantes da linguagem. É a partir dela que criamos dinamismo à página web. O JavaScript nos permite criar

códigos que respondam de acordo com a ocorrência de um destes eventos, que pode variar por exemplo à uma letra digitada no teclado, um arrastar de mouse à um clique de submissão de formulário.

- **Funções de primeira classe**
  No JavaScript, funções são de primeira classe, ou seja, podem ser passadas como parâmetros, serem atribuídas a variáveis ou retornados como qualquer outro objeto.

- **Tipagem**
  O JavaScript possui tipagem fraca e dinâmica. Os tipos são associados com valor, não com variáveis. Isto significa, por exemplo, que uma variável qualquer pode ser um número e mais tarde ser associada a uma string.

- **Criação de código em tempo de execução**
  Uma função de grande importância pertencente ao JavaScript é o **eval**. Com ele, conseguimos transformar e executar em tempo de execução, strings em comandos da linguagem.

3 GRAPHWEB

3.1 Definições básicas de grafos

A aplicação implementada engloba os conceitos de grafos simples, sendo o conhecimento de sua teoria básica o suficiente para utilizar a aplicação, como é mostrado na figura 3.1. Grafos: Um grafo G = (V, E) é um conjunto V de vértices e um conjunto E de arestas (edges em Inglês) onde cada aresta é um par de vértices (Ex.: (v, w)). Um grafo pode ser representado graficamente usando bolinhas para vértices e retas ou curvas para arestas, como representado na figura 3.1

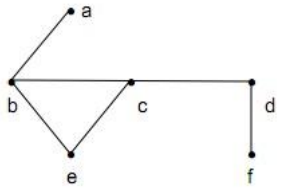

Figura 3.1: Exemplo de um Grafo

- Grafos Simples: grafo não direcionado, sem laços e que existe no máximo uma aresta entre quaisquer dois vértices (sem arestas paralelas).
- Grafo Completo: grafo simples em que, para cada vértice do grafo, existe uma aresta conectando este vértice a cada um dos demais.
- Vértices: unidade fundamental para criação de grafos.
- Arestas: Graficamente as arestas são representadas, no caso dos grafos sem direção como uma linha conectando os dois vértices. Se o grafo é dirigido, então a aresta é representada como uma flecha, que parte do nó de origem e aponta para o nó de destino.

3.2 Introdução ao GraphWeb

Para validarmos o HTML5, foi proposto a implementação de um aplicativo que fizesse forte uso da nova linguagem. Entre os contextos possíveis optou-se por criar uma interface web para visualizar e editar grafos. A escolha foi feita pensando nas próprias matérias ministradas pelo curso que envolvem grafos, possibilitando mais uma opção freeware para o auxiliar no estudo do assunto. Nesta primeira versão, contamos com recursos 3D, atalhos de teclado, drag-and-drop, entre outras funcionalidades que são facilmente atingíveis com o HTML5, tendo sido adicionadas ao projeto visando uma melhor usabilidade do aplicativo pelo usuário.

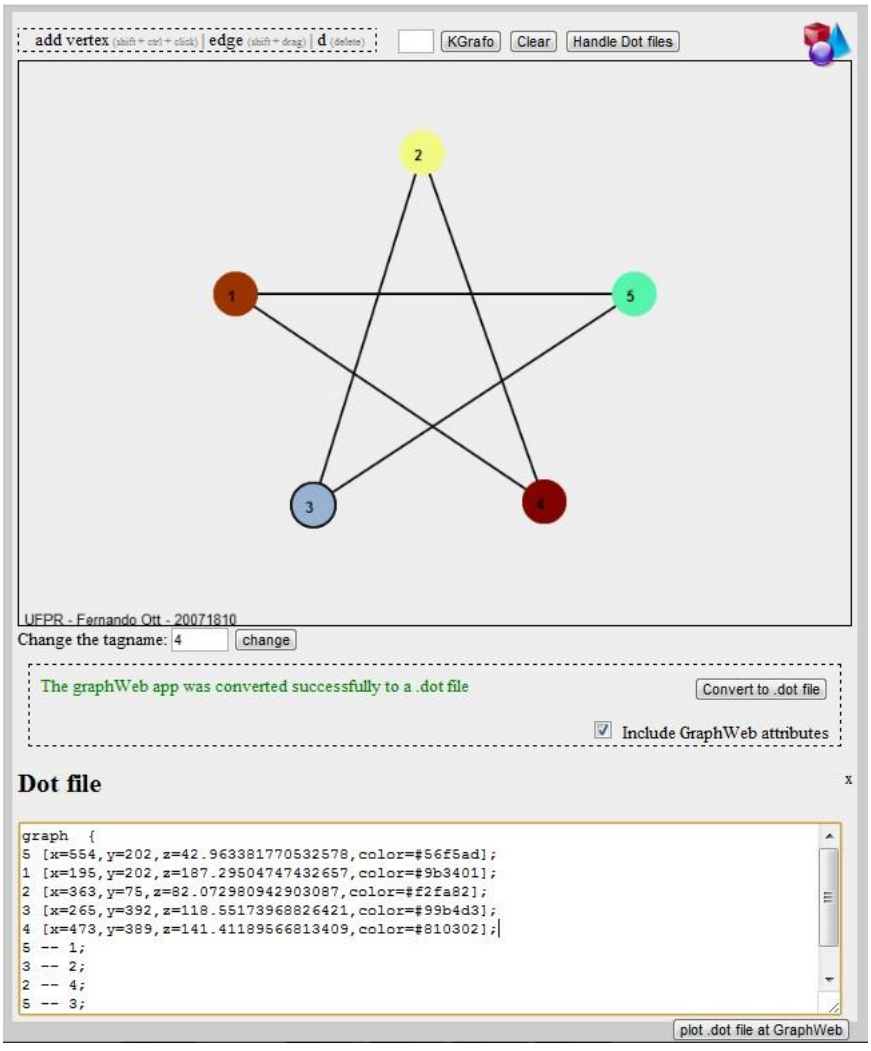

Figura 3.2: PrintScreen do aplicativo web de manipulação de grafos GraphWeb

3.3 Linguagens e bibliotecas utilizadas

O GraphWeb é construído como interface o HTML5, com extenso uso de JavaScript, fazendo uso de jquery (biblioteca JavaScript cross-browser) que ajuda a organizar o código, facilitando o script além da biblioteca js3d usada para o visualizador 3D. A parte Server-side, necessária para interpretar arquivos Dot é implementada em C#, sendo extensível para qualquer outra linguagens, sendo necessário apenas respeitar o modelo de retorno Dot padrão para o GraphWeb.

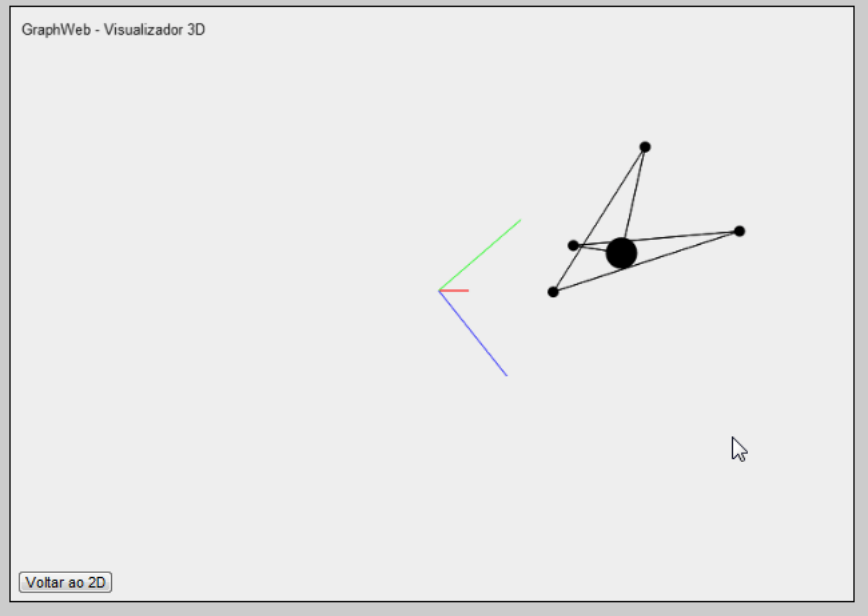

Figura 3.3: Print do visualizador 3D do GraphWeb

3.4 Recursos do GraphWeb

A primeira versão do GraphWeb possui:

- Criação de vértices
- Movimentação de vértices e arestas
- Exclusão de vértices (e respectivas arestas)
- Criação de arestas
- Criação de K-Grafos

- Limpar tudo
- Visualização 3D
- Renomear nomes de vértices
- Manipulação de arquivos Dot

  Exportar grafos um arquivo no formato dot

  Importar grafo a partir de um arquivo com formato dot

A plataforma não possui:

- Exclusão de arestas específicas
- Edição 3D
- Zoom
- Tratamento para arquivos dot com comentários
- Elementos avançados de grafos não são tratados.

1. **Criação de vértices**

   A criação de um vértice único é feito ao posicionar o mouse no local que desejado e pressionando **Ctrl + Shift + clique**.

   Ao manter pressionado o **Ctrl + Shift** veremos que o cursor do mouse alterasse para a figura de uma mão indicativa. Quando o cursor estiver com esta imagem, qualquer clique na área gráfica criará um novo vértice.

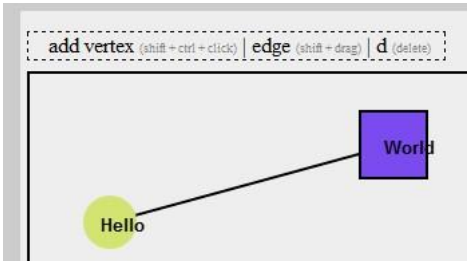

Figura 3.4: Criando vértices no GraphWeb

A versão 1.0 do GraphWeb suporta criação de vértices circulares por padrão e em formato de quadrado. Utilizando o método descrito anteriormente para sua criação, apenas serão gerados vértices circulares, sendo a opção quadrada possível apenas a partir de grafos exportados de arquivos .dot.

2. **Movimentação de vértices**

   A opção de manipulação/edição é a padrão do GraphWeb. Caso nenhuma atalho de teclado esteja pressionado, apenas o cursor mostrará a imagem 3.5 representando a possibilidade de arrastar objetos. Enquanto o cursor estiver apresentando esta imagem, podemos selecionar, arrastar e reposicionar os vértices dentro do espaço gráfico.

Figura 3.5: Figura para manipulação no GraphWeb

3. **Exclusão de vértices**
   Para a exclusão de um vértice, primeiro selecione o vértice desejado. O vértice estará selecionado quando tiver um contorno preto mais acentuado que os outros vértices. Após selecionado o vértice desejado, basta pressionar **delete** para que o vértice seja apagado do grafo.

4. **Criação de arestas**
   Para criarmos uma aresta, posicionamos o mouse no vértice origem desejado, mantemos pressionada a tecla **Shift** e arrastamos o mouse até o vértice destino desejado. Quando mantemos pressionando o **Shift**, o cursor do mouse aparece com a figura 3.6 , o que representa que o modo de criação de arestas esta ativado. Quando o cursor estiver com esta imagem, cliques com o mouse não produziram efeito algum. Apenas ao pressionar o botão esquerdo do mouse e arrasta-lo trará um efeito visual, que no caso representa a criação da aresta. É possível arrastar arestas a partir de qualquer ponto para qualquer ponto, porém elas não serão válidas, caso não tenham como origem e destino um vértice. A versão 1.0 do GraphWeb, não suporta exibe texto às arestas, e não possui suporte à dígrafos, logo a representação de arestas através de setas não é possível.

Figura 3.6: Figura para criação no GraphWeb

5. **Criação de K-Grafos**
   Com o GraphWeb também podemos gerar grafos. Na versão 1.0 o único tipo de grafo criado são os completos. Como mostrado na figura 3.7

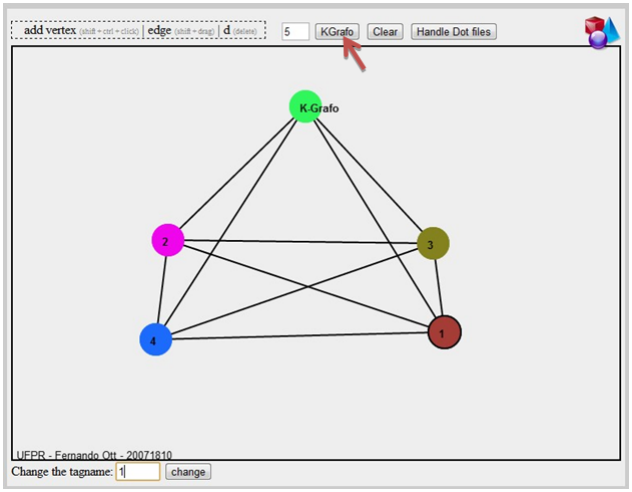

Figura 3.7: Criar K-Grafos

Para criarmos um grafo completo basta selecionar o número de vértices que deseja que seu grafo completo possua e clicar na opção 'KGrafo' do menu superior. O posicionamento dos vértices é escolhido de maneira aleatória.

6. **Limpar Tudo**

   Caso queira iniciar um novo exemplo, no menu superior do GraphWeb existe o botão **Clear**. Ao clicarmos nele, antes verifica-se se o usuário tem certeza que deseja limpar tudo. Em caso afirmativo toda a estrutura de vértices e arestas é excluída e em seguida atualizamos o canvas redesenhando-o, o que trará a área limpa para a criação de um posterior novo exemplo.

7. **Visualização 3D**

   Através da API opensource Js3d, e com algumas adaptações e melhorias, o GraphWeb também apresenta um visualizador 3D dos grafos construídos na tela 2D. Veja na figura 3.8

   A versão 1.0 não possui suporte a edição em 3D, a coordenada z é escolhida randomicamente quando o grafo é criado. A única maneira disponível para edita-lo é exportando o grafo para um arquivo .dot, e alterando o atributo z do vértice desejado.

8. **Renomear Vértices**

   Ao criar um vértice através do atalho **Ctrl + Shift + Clique**, o nome padrão do grafo é o numero incrementado de vértices já existentes, ou seja, quando o terceiro vértice é criado, seu nome será 2 (o primeiro vértice é 0). Como pode ser visto na

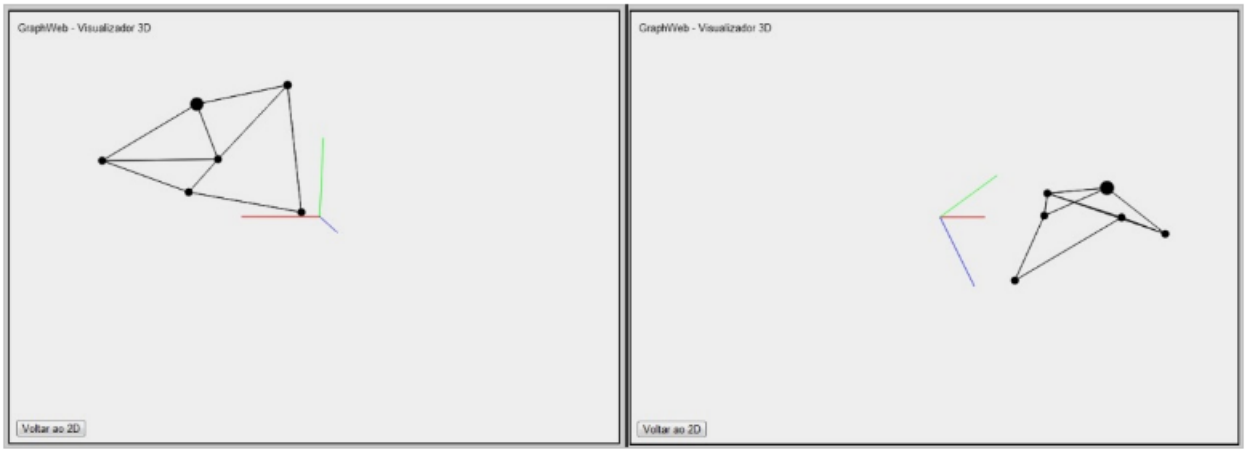

Figura 3.8: Visualização 3D dos grafos

figura 3.9.

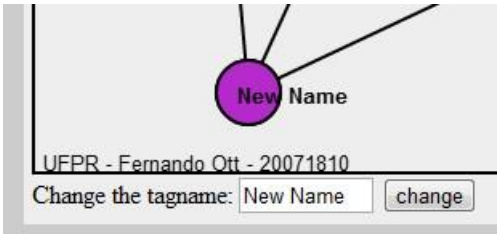

Figura 3.9: Renomear vértices

Esta nomeação padrão pode ser facilmente alterada. Ao selecionar um vértice especifico, no canto inferior esquerdo surgira a opção 'Change the tagname' contendo o nome atual do vértice. Basta incluir o novo nome desejado e selecionar a opção à direita do campo texto, **change** para que o vértice tenha seu novo nome alterado.

9. **Manipulação de arquivos .dot**

   Uma das necessidades que o aplicativo possuía era, após a manipulação dos grafos, salvar o resultado final estudado. A melhor opção encontrada foi utilizar o formato dot (arquivo textual para descrição de grafos), por tratar-se de um padrão largamente difundido entre os aplicativos voltados ao estudo de grafos. Esta primeira versão, já conta com o serviço de exportar e importar grafos a partir de arquivos .dot. Ao acessar o GraphWeb por padrão, será apresentado um grafo Hello World, que é extraído do arquivo .dot mostrado na figura 3.1

   O padrão .dot para modelagem de grafos é bastante rica em detalhes sendo possível definir os mais diversos atributos para definição de um grafo.

Graph{
Hello – World
}

Tabela 3.1: Hello World!

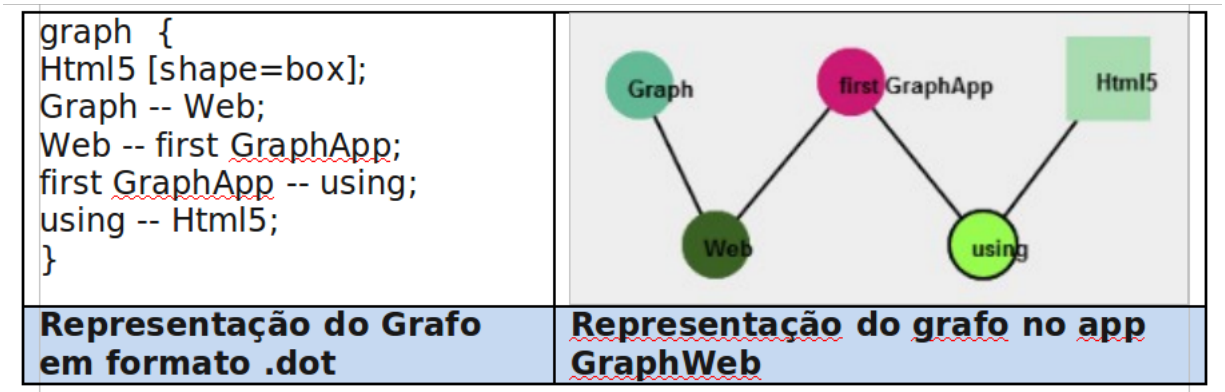

Figura 3.10: Representação do grafo na aplicação

Como podemos ver no exemplo acima 3.10, o vértice **Html5** recebeu o atributo 'shape=box'. Isto implicou no formato de uma caixa quando o grafo foi renderizado no GraphWeb. Para acessar a ferramenta de conversão para formato .dot selecione a opção 'Handle Dot file' no menu superior. Depois de selecionado, o aplicativo abrira na parte inferior uma sessão dedicada à importação e exportação do grafo visual para o formato dot, como podemos ver na imagem 3.11.

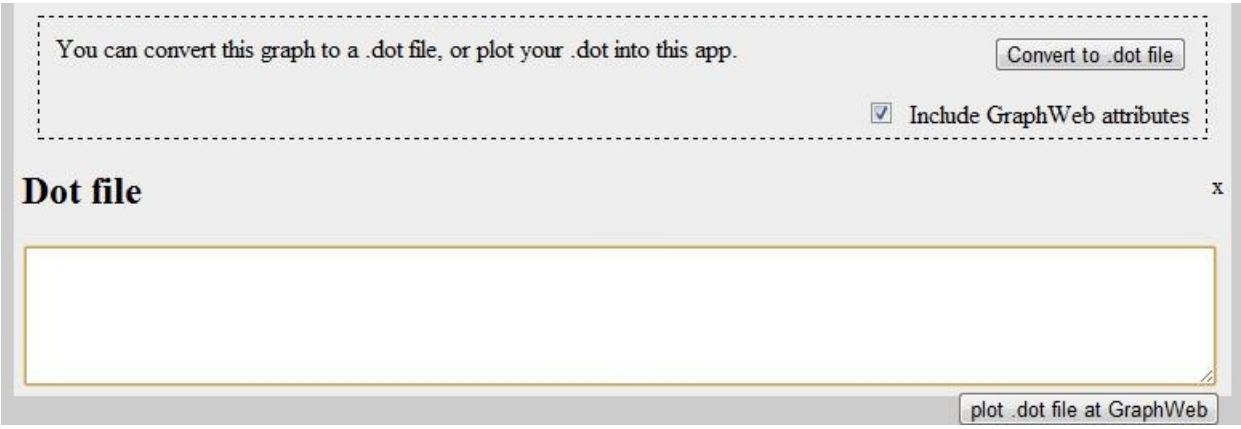

Figura 3.11: Exemplo .dot

- Exportar grafos para um arquivo no formato dot

  A versão 1.0 do GraphWeb conta com uma ferramenta de fácil exportação para o formato dot. Depois de concluído o trabalho visual com o grafo, temos

a possibilidade de salva-lo como texto em formato dot. Para isto, primeiro selecione a opção 'Handle Dot files' no menu superior. Em seguida, selecione 'Convert to .dot file'. Abaixo do botão 'Convert to .dot file', temos a opção selecionada por padrão 'Include GraphWeb attributes'. Caso mantenha esta opção selecionada, o arquivo .dot será criado incluindo atributos utilizados pelo GraphWeb identificar a posição dos vértices e arestas no aplicativo. Se esta opção não for selecionada, o arquivo .dot será criado sem os atributos reservados e ao renderizarmos o mesmo grafo posteriormente, as posições e cores dos vértices serão escolhidas aleatoriamente. Depois de selecionado o botão 'Convert to .dot file', o grafo, agora em versão textual no formato dot será exibido na caixa de texto, podendo ser editado/copiado.

graph {
Html5 [shape=box];
Graph – Web;
 	
Web – 5;
5 – using;
using – Html5;
}
	graph {
	Html5
	[x=474,y=151,z=22,color=#a9dcb0,
	shape=box];
	
	Graph [x=197,y=155,z=152,color=#64b996];
	Web [x=242,y=249,z=111,color=#3c6125];
	5 [x=320,y=149,z=123,color=#cc1973];
	using [x=400,y=249,z=58,color=#9afb51];
	Graph – Web;
	Web – 5;
	5 – using;
	using – Html5;
	}

Sem incluir atributos reservados do GraphWeb	Com atributos reservados do GraphWeb incluídos.

Tabela 3.2: Atributos reservados no graphweb

No tabela 3.2, quando a opção de incluir os atributos do GraphWeb é selecionada, os vértices ganham os atributos **x, y, z** e **color**. Ao preservarmos estes atributos, podemos garantir a consistência do grafo criado através do aplicativo. Quando houver a necessidade de remodelar o grafo, agora convertido em formato dot, suas posições e cores serão mantidas da mesma maneira que criada.

- Importar grafo a partir de um arquivo com formato dot.
  Além da criação manual de grafos e dinâmica de grafos completos, o GraphWeb permite outra maneira para criação de grafos. A partir de arquivos no formato dot. Para acessar esta opção, selecione o botão no menu superior 'Handle Dot

files'. Após aberto a sessão de importação/exportação para o formato Dot, insira o texto em formato dot no campo texto e selecione a opção 'plot .dot file at GraphWeb'.

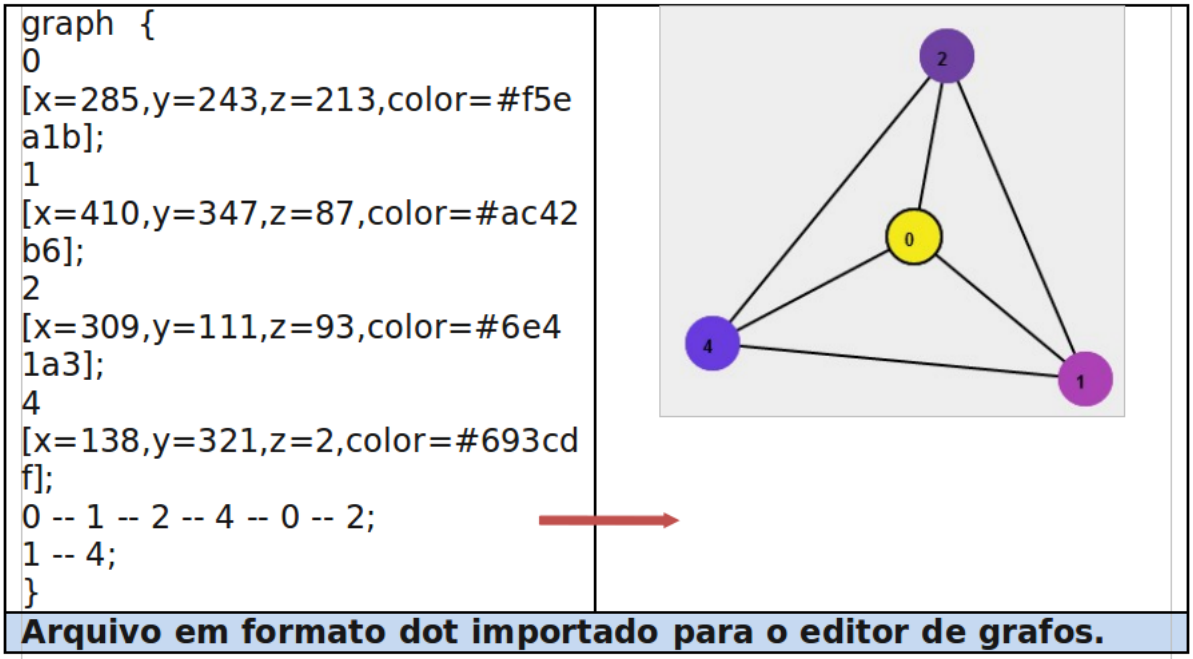

Figura 3.12: Editor de grafos

Existem dois alertas possíveis na importação de grafos figura 3.13.

Figura 3.13: Tipos de alertas

Caso a primeira mensagem em vermelho figura 3.13 seja apresentada, é importante verificar se o arquivo dot copiado está completo e sem erros de sintaxe. Se a mensagem apresentada após a importação for a segunda, em azul, a importação foi realizada e já estará sendo apresentada na área de edição gráfica. Outro detalhe são os comentários. A primeira versão do GraphWeb não aceita comentários nos arquivos importados, logo remova-os caso existam, pois poderá apresentar erros no resultado do grafo importado.

3.5 Estrutura de implementação

O GraphWeb faz intenso uso de JavaScript para realizar suas operações. Ao longo do projeto, foram definidos padrões para facilitar seu eventual desenvolvimento posterior, transformando-o em um framework de fácil utilização.

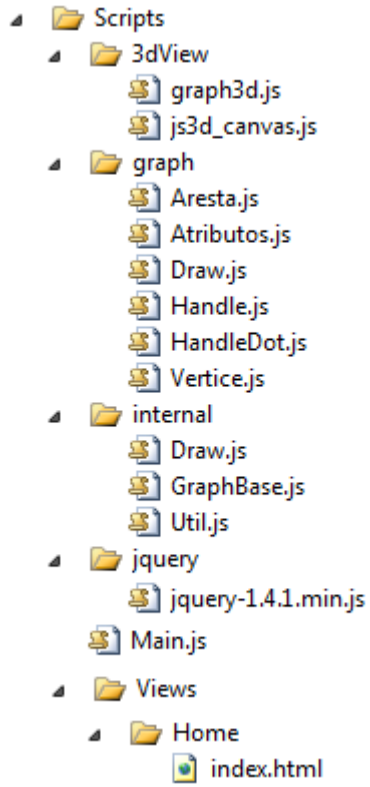

Figura 3.14: Estrutura de implementação

Como é observado na estrutura de pastas da figura 3.14, os arquivos foram divididos em várias pastas, separados por funções e permissões.

- **3dView**: Código responsável para visualização 3D.
- **Internal**: Códigos internos para o funcionamento base do aplicativo. É aconselhável evitar sua alteração, implicando diretamente no funcionamento correto do sistema. Nele guardamos os métodos genéricos para desenho, a instanciação de objetos relevantes do sistema e controle do posicionamento do mouse.
- **Jquery**: Biblioteca utilizada para simplificar o código JavaScript.
- **Graph**: É nesta sessão, de livre acesso aos desenvolvedores, que novas funcionalidades são implementadas.

Aresta.js: Métodos que envolvam arestas, como criar, remover.

Vertices.js: Métodos que atuam na construção de vértices.

Handle.js: Arquivo principal de serviço para a manipulação dos objetos vértices e arestas. Nele concentram-se métodos como arrastar, deletar, limpar entre outros.

HandleDot.js: Nele concentram-se os retornos server-side quando realizamos alguma importação/exportação para arquivos no formato dot.

Atributos.js: Serviços e métodos para consumir os atributos de um vértice/aresta. É a partir dele que definimos as características dos vértices e arestas que devem ser desenhadas.

Draw.js: Os métodos para desenho em tela concentram-se neste arquivo. Deve-se utilizar ele para implementar novas formas de desenho. O draw.js da pasta internal cuida da renderização geral do aplicativo e deve ser evitado.

- **Main.js**: Neste arquivo, definimos eventos keyboard, ligamos os elementos html aos métodos correspondentes atrelando-os eventos GraphWeb que devem executar.
- **Index.html**: A página html onde será renderizado o aplicativo.

3.5.1 Estrutura dos objetos Vértice e Aresta

O GraphWeb foi desenvolvido utilizando como base principal dois grandes arrays de objetos. Vertices e Arestas. Na figura 3.15 encontramos lista de propriedades que os compõe.

No JavaScript, objetos não possuem tamanho, mesmo sendo arrays. Como existe a necessidade de guardar o numero de vértices e arestas existentes, utilizamos dois contadores globais fortemente ligados aos objetos. O **countV**, que guarda o tamanho do objeto Vertices e **count**, responsável por armazenar o tamanho do objeto Arestas. Outra especificidade interessante destes objetos é sua propriedade **draw(i)**. Esta propriedade é o método responsável por desenhar o objeto ao qual pertence. Em outras palavras, os vértices e arestas possui uma inteligência embutida que os deixa responsável por se auto desenharem na área de edição gráfica do GraphWeb. Desta maneira, podemos facilmente desenhar todo grafo chamando seus métodos internos de desenho, como é possível perceber no trecho de código da figura 3.3.

Vertices

Propriedades	Descrição
x	Coordenada x
y	Coordenada y
z	Coordenada z
color	Cor atribuída ao vértice
tag	Nome escrito ao vértice. Não existe vértice sem tag.
attr	Array de atributos do vértice ex.: attr[0].shape = "box"
draw(i)	Método atrelado à cada vértice que o desenha.

Arestas

Propriedades	Descrição
v1	Objeto vértice
v2	Objeto vértice
attr	Array de atributos da aresta ex.: attr[0].color = "blue"
draw(i)	Método atrelado à cada vértice que o desenha.

Figura 3.15: Tabela de Propriedades

```
//Cria Arestas
for (var j = 0; j < countV; j++) {
Arestas[j].draw(j);
}
//Cria Vertices
for (var j = 0; j < count; j++) {
Vertices[j].draw(j);
}
```

Os loops chamam os respectivos métodos de desenho de cada objeto.

Tabela 3.3: Criar arestas e vértices

4 Conclusões finais

4.1 Lições aprendidas

O HTML5 trouxe uma API poderosa e vasta em termos de novas funcionalidades. Cada novo segmento traz grandes inovações que fazem desta nova versão do HTML uma fonte rica para diversos ramos de estudo. O fato de grande parte destas novas implementações exigirem o intenso uso de JavaScript, seu estudo mais aprofundado despertou-me o interesse em continuar minhas pesquisas nesta área, principalmente por juntos (HTML e JavaScript) conseguirem criar, comparando-os ao Flash e outros, aplicações similares porém muito mais leves e portáveis. Entre os conhecimentos adquiridos, destaco estudo envolvendo a Api 3D. Por estar ainda pouco desenvolvida exigiu um estudo completo de seu código e sua quase reestruturação, proporcionando o aprendizado dos conceitos para criação gráfica 3D. Também destaco o crescimento na qualidade de código cliente-side (JavaScript) com utilização do jquery, o aprendizado de conceitos da linguagem funcionalista, manipulação de objetos não tipados, serialização de objetos, além da incorporação dos conceitos envolvendo o desenvolvimento para interfaces gráficas.

4.2 Conclusão

Através do aplicativo desenvolvido, foi possível ganhar uma visão mais prática e aprofundada sobre os novos recursos que o HTML5 oferece, verificando uma robustez muito maior do que a apresentada em sua versão anterior. O HTML5 passa a ter um papel muito maior e agressivo dentro do contexto das páginas web do que antes apresentava. Deixa de ser apenas uma linguagem de formatação para tornar-se também uma ferramenta gráfica, player de vídeo e áudio entre tantas novas possibilidades. Após seu estudo nota-se que mesmo apenas com uma versão de testes lançada e nem todos seus recursos ainda disponíveis, o HTML5 chega com apoio massivo do mercado, com claro potencial para desbancar os demais aplicativos gráficos, atualmente utilizados em larga escala na

web. Entre suas principais promessas fica a padronização entre browsers onde todos só tendem a ganhar. De um lado desenvolvedores, com a possibilidade de ter seu trabalho rodando com a mesma qualidade em todos os browsers, viabilizando maior foco em uma mesma linguagem e consequentemente sua evolução natural. Por outro lado, o usuário final que tende a se beneficiar com interfaces mais leves, dinâmicas e sofisticadas.

Ressalto também a necessidade futura de uma plataforma para o desenvolvimento gráfico ao HTML5. Em uma rápida comparação entre Flash e Canvas HTML5, é expressiva a diferença quanto a leveza da aplicação do segundo, porém o tempo de desenvolvimento do primeiro também é expressivamente menor por contar com uma plataforma direcionada para o desenvolvimento gráfico.

Referências Bibliográficas

ARTIGO Sobre o Canvas. [s.d.]. Website. http://dev.opera.com/articles/view/HTML-5-canvas-the-basics/.

BERNERS Lee. [s.d.]. Website. www.w3.org/people/Berners-Lee.

Creating a panting canvas. [s.d.]. Website. http://www.w3schools.com/HTML5/HTML5_ref_eventattributes.asp.

DRAGDROP. [s.d.]. Website. http://www.wxs.ca/js3d/.

ESPECIFICACAO do HTML5 pela W3C. [s.d.]. Website. http://dev.w3.org/HTML5/spec/Overview.HTML.

ESPECIFICAçãO do JavaScript ECMA-262. [s.d.]. Website. http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf.

Events HTML5. [s.d.]. Website. http://www.w3schools.com/HTML5/HTML5_canvas.asp.

GRAPHICSJAVASCRIPT - 3D. [s.d.]. Website. http://www.williammalone.com/articles/flash-vs-HTML5-canvas-drawing/.

HTMLHISTORY. [s.d.]. Website. http://www.w3.org/People/Raggett/book4/ch02.HTML.

Introdução sobre HTML5. [s.d.]. Website. http://www.codeproject.com/KB/HTML/HTML5-intro.aspx.

PANTING canvas. [s.d.]. Website. http://dev.opera.com/articles/view/HTML5-canvas-painting/.

PáGINA da NCSA Mosaic browser. [s.d.]. Website. http://www.ncsa.illinois.edu/Projects/mosaic.HTML.

Teoria dos grafos. [s.d.]. Website. http://www2.dc.ufscar.br/~jose/courses/tg/btg.pdf.

Tutorial HTML5. [s.d.]. Website. http://www.w3c.br/cursos/HTML5/conteudo/.