Desenvolvimento de Jogos em HTML5

Jucimar Maia da Silva Jr Emiliano Carlos M. Firmino

Universidade do Estado do Amazonas, Coordenação de Engenharia da Computação, Brasil

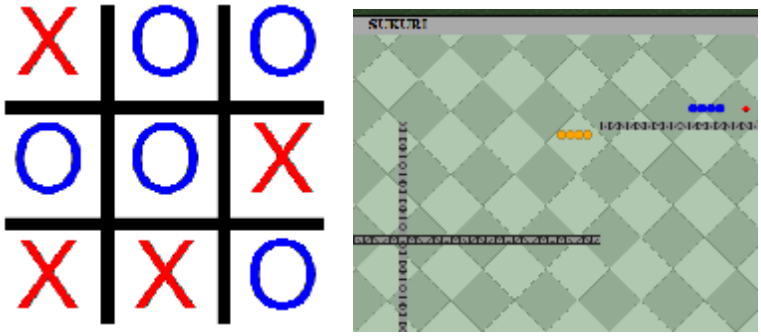

Figura 1: O Jogo da Velha e o Jogo das Cobras Sukuri desenvolvido em HTML5/JavaScript.

Resumo

HTML5 é a próxima geração do HTML. É um padrão aberto e suportado pela maioria dos novos navegadores. Suas novas funcionalidades o tornam uma alternativa à tecnologia proprietária Adobe Flash. Este artigo descreve o uso do HTML5 no desenvolvimento de jogos online dentro do navegador. Foram criados dois jogos, o Sukuri e o Jogo da Velha para investigar e exemplificar a utilização desse novo padrão.

Palavras-Chave: HTML5, JavaScript, Canvas, Desenvolvimento de Jogos

Contato dos Autores:
{jucimar.jr, elmiliox}@gmail.com

1. Introdução

A grande maioria dos jogos incorporados ao navegador é construída usando Adobe Flash. Além de jogos, essa tecnologia é usado em animações, banners, como tocador de vídeo e muitas outras aplicações. A Adobe afirma que 98% dos navegadores possuem sua tecnologia instalada e que a mesma é aberta, segura e com bom desempenho [Adobe 2010]. Isso era aceito pela comunidade web sem grandes controvérsias. Tudo mudou quando a Apple disse que não incorporaria o Adobe Flash aos seus novos produtos iPhone e iPad por considerá-lo com muitos erros, sem segurança e consumidor voraz de bateria [Apple 2010]. A Apple divulgou que o futuro seria o uso de HTML5 em detrimento ao Flash [Apple 2010]. Jogada de marketing ou não, essa opinião foi logo seguida pela Microsoft, Google, Opera e outras empresas. Essa afirmação da Apple acabou deixando o HTML5 em destaque.

HTML5 tem como objetivo padrão facilitar e melhorar a estruturação e a apresentação conteúdo na Web através de novas *tags* e incorporar funcionalidades como o uso de canvas, tocar vídeos, *drag-and-drop* que antes só eram possíveis através do uso de *plugins* proprietários como Adobe Flash e Microsoft Silverlight. Ele é um padrão aberto, livre do pagamento de *royalties*. Apesar do lançamento oficial do HTML5 ser apenas em 2012 [Whatwg 2009], já é possível construir sites usando esse padrão, pois os navegadores novos (Firefox 3.6, Firefox Beta 4, Safari 5, Internet Explorer 8, Opera 10, Chrome 6) são aderentes de alguma maneira ao novo padrão.

O objetivo deste artigo é demonstrar o uso do HTML5 como substituto do Adobe Flash no desenvolvimento de jogos simples incorporados ao navegador. A Seção 2 mostra os trabalhos relacionados; a Seção 3 explica o funcionamento da tag *canvas* do HTML5; a Seção 4 mostra o desenvolvimento do Jogo da Velha; a Seção 5 mostra o desenvolvimento do jogo Sukuri; a Seção 6 mostra as considerações do artigo e trabalhos futuros.

2. Trabalhos relacionados

Os trabalhos estão dentro dos sites das empresas desenvolvedoras de navegadores ou de padrões da Web. Pode-se citar que o site de desenvolvimento do Opera explica como trabalhar com HTML5 nesse navegador [Opera 2010]. O site da *World Wide Web Consortium* (W3C) mostra como funciona a tag canvas do HTML5 [W3c 2010]. O site de desenvolvimento do WebKit da Apple mostra um tutorial sobre o uso de HTML5 [Webkit 2010].

3. Funcionamento da *tag* canvas HTML5

O Canvas é um novo elemento do HTML5 que permite "desenhar" diretamente dentro uma página Web. Sua *tag* de marcação é: `<canvas></canvas>` e possui apenas dois atributos que são *width* e *height*

que definem as dimensões do canvas no HTML. Se o navegador não suportar a tag, nada será mostrado, nem mesmo mensagem de erro. É possível então colocar um pequeno texto para passar o *feedback*:

```
<canvas width="320" height="240"
id="canvas" >
<p>O seu browser não suporta o canvas</p>
</canvas>
```

Se o navegador suportar a tag, esta pode ser manipulada pelo JavaScript. Por enquanto é possível apenas manipular em 2D (2 dimensões). Está em desenvolvimento o *WebGL* [WebGL 2010] que permitirá criar figuras 3D.

3.1 Desenhando no Canvas

O canvas possui como forma primitiva apenas o retângulo. Para produzir formas complexas é preciso desenhar o *path*, um caminho que será traçado dentro do canvas. Para isso existe uma *Application Programming Interface* (API) específica. A partir de funções simples se desenham formas complexas. Para isso utiliza-se um ponteiro que cada vez que é movido, vai "riscando" por onde passa.

3.2 Utilizando Imagens

O canvas permite utilizar imagens e aplicá-las diretamente. São suportados os formatos de imagem utilizados na Web como `.gif`, `.jpg` e `.png` .

3.3 Utilizando texto

O canvas possui uma API para aplicação de texto desprovida de recursos avançados. Toda a formatação e posicionamento do texto ficam sobre responsabilidade do desenvolvedor. Formatar o texto usando *Cascade Style Sheets (CSS)* produz resultado superior ao que o canvas pode prover.

3.4 Exemplo completo de desenho no canvas com HTML5

Para exemplificar o uso das funcionalidades mostradas nas subseções anteriores, uma bandeira do Brasil foi desenhada diretamente do canvas usando o JavaScript.

Figura 2: Bandeira do Brasil desenhada somente com HTML5 canvas e JavaScript.

Crie uma página HTML com o conteúdo:

```
<canvas width="320" height="240"
id="canvas">
O seu browser não suporta o canvas
</canvas>
```

Crie um script JavaScript com o seguinte código:

```
var canvas =
document.getElementById("canvas");
var ctx = canvas.getContext("2d");
ctx.beginPath();
var width = canvas.width;
var height = canvas.height;
var dx = 10, dy = 10;
//fundo verde
ctx.fillStyle = "green";
ctx.fillRect(0,0,width,height);
//losango amarelo
ctx.moveTo(dx,height / 2);
ctx.lineTo(width / 2,dy);
ctx.lineTo(width - dx,height / 2);
ctx.lineTo(width / 2,height - dy);
ctx.closePath();
ctx.fillStyle = "yellow";
ctx.fill();
//circulo azul
ctx.beginPath();
ctx.arc(width / 2,height / 2,60,0,
2*Math.PI,true);
ctx.fillStyle = "blue";
ctx.fill();
```

4. Desenvolvimento do Jogo da Velha

O Jogo da Velha foi desenvolvido como primeiro experimento com o objetivo de testar as funcionalidades do canvas. É um jogo que não precisa renderização complexa. Uma visão geral das classes em JavaScript envolvidas é mostrada na Figura 3:

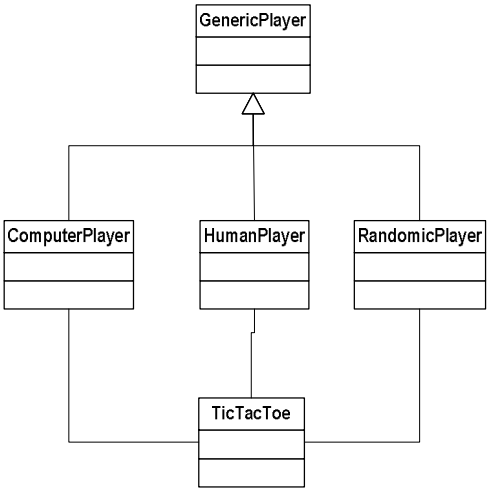

Figura 3: Classes do Jogo da Velha.

A classe *GenericPlayer* é uma classe abstrata para um jogador. *ComputerPlayer* modela um jogador controlado pelo computador e joga usando um algoritmo de MiniMax baseado em [Jones 2008]. O *RandomicPlayer* é um jogador controlado pelo computador e joga aleatoriamente sem a preocupação de vencer. O *HumanPlayer* é o jogador que obedece aos comandos do jogador humano. Como o Jogo da Velha é apenas para dois jogadores, dentro do código-fonte hora é colocado o *ComputerPlayer* ora o *RandomicPlayer*. É possível ainda criar partidas de computador contra computador.

O jogo em si fica dentro da classe *TicTacToe*. A cada turno, o tabuleiro é transformado em um vetor e passado como parâmetro para o jogador controlado pelo computador. Um algoritmo então calcula a próxima jogada que é renderizada no canvas.

4.1 Desenhando o tabuleiro

O tabuleiro é desenhado usando linhas desenhadas diretamente no canvas:

```
TicTacToe.drawGrid = function () {
    var grid = this.ctx;
    var tileY = this.canvas.height / 3;
    var tileX = this.canvas.width / 3;
    var linX, linY, delta = 20;
    grid.save();
    //formata linha
    grid.lineCap = 'square';
    grid.lineWidth = 12;
    //renderiza Grid
    grid.beginPath();
    for(var i = 1; i < 3; i++){
        linY = i * tileY + 0.5;
        linX = i * tileX + 0.5;
        grid.moveTo(0+delta,linY);
        grid.lineTo(this.canvas.width-
delta,linY);
        grid.moveTo(linX,0+delta);

grid.lineTo(linX,this.canvas.height-
delta);
    }
    grid.stroke();
    grid.restore();
}
```

4.2 Desenhando os “O” e “X”

As jogadas são marcadas através de “O” e “X”. Ao invés de desenhar um círculo ou duas retas cruzadas para fazer o X , optou-se por usar a funcionalidade do canvas de trabalhar com textos. O “O” é a letra O e o “X” é a letra X usando a fonte Arial 65:

```
TicTacToe.drawMark=function(pos,content){
var symbol, color;
switch(content){
    case SQUARE.EMPTY:
        return;
    case SQUARE.O:
        symbol = "O"; color
="blue";break;
    case SQUARE.X:
        symbol = "X"; color =
"red";break;
    default:
        symbol = "?"; color = "black";
}
var x = pos % 3, y = Math.floor(pos / 3);
var grid = this.ctx;
grid.save();
grid.font = "65pt Arial";
grid.textAlign ="center";
grid.textBaseline = "middle";
grid.fillStyle = color;
grid.fillText(symbol,
    this.tileX * x + this.tileX/2,
    this.tileY * y + this.tileY/2,
    this.tileX);
grid.restore();
}
```

5. Desenvolvimento do Jogo Sukuri

O jogo Sukuri foi desenvolvido com o propósito de analisar como o HTML5 se comportaria diante de um jogo que necessite controlar eventos e renderização constantes. Ele consiste no jogador controlar uma cobra/minhoca que se movimenta dentro de um labirinto. Seus adversários são outras cobras controladas pelo computador (cobras-robô). Cada uma delas tem o objetivo de comer uma fruta que aparece aleatoriamente no labirinto. Cada vez que uma fruta é comida, a cobra que comeu aumenta de tamanho. Se a cobra colide com a parede ou com outra cobra ela morre. As fases do jogo mudam à medida que os pontos forem avançando e com isso as cobras tornam-se cada vez mais rápidas. Essa lógica foi modelada usando as classes da Figura 4:

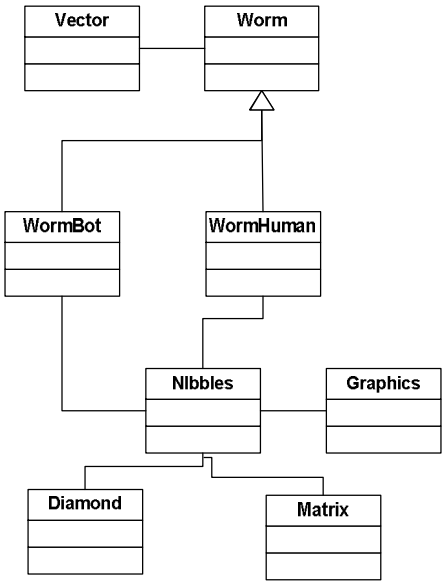

Figura 4: Classes do Jogo Sukuri.

A classe *Vector* representa uma coordenada no mapa, controla as operações vetoriais e é usada para o

"corpo" da cobras. A classe *Worm* é uma classe abstrata para criação das cobras e serve de base para criar a classe controlada pelo jogador (*WormHuman*) e a controlada pelo computador (*WormBot*). A classe *Nibbles* é o jogo em si. Ele utiliza a classe *Graphics* que encapsula o uso do canvas. A classe *Diamond* controla as frutas que a cobra come. A classe *Matrix* mapeia os labirintos.

5.1 Algoritmo de *Pathfinding*

A classe *WormBot* utiliza um algoritmo de *pathfind* para encontrar a fruta. Optou-se usar o *breadth-first search (BFS)* baseado em [Milligton 2006] por se de implementação simples.

5.2 Técnica de *Double Buffering*

Para melhorar a experiência do usuário e evitar que a tela fique piscando (*flickering*), a técnica de *Double Buffering* citada em [Lamothe 1999] foi aplicada. Ela consiste em criar um canvas secundário, desenhar nele e depois copiar as variáveis para o canvas principal. Isso evita que o jogador perceba a tela sendo desenhada a cada jogada.

6. Considerações Finais e Trabalhos Futuros

Este artigo mostrou a utilização de HTML5 e JavaScript para criação de dois jogos simples. O desenvolvimento ocorreu sem ajuda de plugins proprietários ou mesmo frameworks. Utilizaram-se exclusivamente linguagens e padrões abertos. Isso demonstra que a tecnologia HTML5 está despontando e deve-se ser observada com atenção pela comunidade de desenvolvimento de jogos.

O HTML 5 é uma tecnologia em desenvolvimento e por isso existem diferentes implementações em diferentes navegadores. As técnicas e funções Javascript usadas no desenvolvimento funcionaram sem problemas. Alguns testes verificaram que existe uma diferença de implementação das curvas de Belzier entre o Chrome 6 e o Firefox Beta 4. À medida que o padrão HTML 5 avança para a padronização essas diferenças deverão desaparecer.

O desenvolvimento dos jogos no experimento serviu para selecionar o HTML5 como possível interface de jogos em rede. Os próximos passos serão exploração dos recursos de áudio e vídeo do HTML5, a experimentação do WebGL e a comunicação do HTML5, usando WebSockets [WebSockets 2010], com um servidor de desenvolvido na linguagem Erlang. Estes passos são parte de um trabalho futuro de desenvolvimento de um jogo em rede que está na sua fase inicial de modelagem, onde a implementação da interface com HTML5 pode apresentar resultados satisfatórios. Os jogos e outros experimentos encontram-se disponíveis no site http://www.est.uea.edu.br/ludus/html5.

Agradecimentos

Os autores gostariam de agradecer o apoio da Universidade do Estado do Amazonas – UEA. Jucimar Junior agradece a Cristina Araújo por ajudar na revisão do artigo. Emiliano Firmino agradece a bolsa de pesquisa recebida da Fundação de Amparo à Pesquisa do Estado do Amazonas - FAPEAM.

Referências

APPLE, 2010, THOUGHTS ON FLASH, Disponível em: http://www.apple.com/hotnews/thoughts-on-flash/ [Acessado em 1 agosto 2010]

ADOBE,2010, OPEN ACCESS TO CONTENT AND APPLICATIONS, Disponível em: http://blogs.adobe.com/conversations/2010/02/open_access_to_content_and_app.html [Acessado em 1 agosto 2010]

LAMOTHE, A., 1999. Tricks of Windows Game Programming Gurus. SAMS.

JONES, T., 2008. Artificial Inteligenge: A Systems ApproachT. Infinity Science Press.

MILLIGTON, I., 2006. Artificial Inteligence for Games. Morgan Kauffman Publisher.

OPERA, 2010, HTML 5 CANVAS – THE – BASICS, Disponível em: http://dev.opera.com/articles/view/html-5-canvas-the-basics/ [Acessado em 1 agosto 2010]

WEBGL, 2010, WEBGL, Disponível em: http://www.khronos.org/webgl/ [Acessado em 1 agosto 2010]

WEBKIT, 2010,WEBKIT DOM PROGRAMMING TOPICS: USING THE CANVAS, Disponível em: http://developer.apple.com/mac/library/documentation/AppleApplications/Conceptual/SafariJSProgTopics/Tasks/Canvas.html [Acessado em 1 agosto 2010]

WEBSOCKET, 2010, WEBSOCKETS, Disponível em: http://dev.w3.org/html5/websockets/ [Acessado em 1 agosto 2010]

W3C,2010, HTML5 CANVAS TAG, Disponível em: http://www.w3schools.com/html5/tag_canvas.asp [Acessado em 2 agosto 2010]

WHATWG, 2009, FAQ – WHATWG WIKI, Disponível em: http://wiki.whatwg.org/wiki/FAQ [Acessado em 1 agosto 2010]