



**UNIVERSIDADE
FEDERAL
DE PERNAMBUCO**



UFPE - UNIVERSIDADE FEDERAL DE PERNAMBUCO

CIN - CENTRO DE INFORMÁTICA

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

**GUIDELINE DE CONSTRUÇÃO DE
JOGOS EDUCACIONAIS PARA A WEB
USANDO HTML 5**

JONATHAN SOARES

RECIFE

2013

JONATHAN SOARES

GUIDELINE DE CONSTRUÇÃO DE JOGOS EDUCACIONAIS PARA A WEB USANDO HTML 5

Monografia apresentada como requisito parcial para a obtenção do Grau de Bacharel em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco

Orientador: **FERNANDO DA FONSECA DE SOUZA, PhD**

RECIFE

2013

Assinaturas

Monografia apresentada como requisito parcial para a obtenção do Grau de Bacharel em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco

Jonathan Soares

Orientando

Fernando da Fonseca de Souza

Orientador

RECIFE

2013

*“Sucesso não é o final, falhar não é fatal: é a
coragem de continuar que conta.”*

Winston Churchil

Agradecimentos

Agradeço a meus pais, Luiz e Sueli por terem me proporcionado toda a base para os meus estudos, por sempre me incentivar e acreditar em mim e serem até hoje um dos maiores exemplos para a minha vida.

A minha irmã Pamela que me incentivava em todas as horas, sempre me fortalecendo com seus conselhos para seguir a diante, com trabalho.

Agradeço muito ao meu orientador Fernando Fonseca, por ter me orientado neste trabalho e me acolhido, tornando-se mais que um professor, mais um conselheiro sincero e amigo que usa a sua maturidade e experiência para dar força nos momentos de angústia.

Aos meus colegas de graduação André Estevão, Cleydyr Bezerra, Filipe Martins, Jacinto Filipe, Jesus Jackson, Luiz Antonio, Osman Torres, Paulo Orlando, Rafael Santos, Tarcísio Coutinho que me ajudaram muito com sua companhia e com os projetos das disciplinas ao longo da minha graduação.

E a Deus por colocar no meu caminho pessoas tão maravilhosas que só me levam ao sucesso.

Muito obrigado a todos pelo apoio!

Índice de Figuras

Figura 2.1 - Classificação dos Jogos e objetivos da Aprendizagem.....	9
Figura 2.2- Características e Sub características de Qualidade Externa e Interna	11
Figura 3.1 - Comportamento das <i>tags margin, border e padding</i>	25
Figura 3.2 - Diferença entre <i>Outline e Border</i>	27
Figura 3.3 - A utilização do <i>float:right</i>	28
Figura 4.1 - Tela Inicial do Browserquest em um tablet.....	39
Figura 4.2 - Tela do Browserquest em um iPad e em um iPhone	40
Figura 5.1 - Profissionais para desenvolvimento de games em geral.....	43
Figura 5.2 - Equipe Interdisciplinar para jogos educativos.....	44
Figura 5.3 - Fases de Criação do Jogo	45
Figura 5.4 - Etapas de desenvolvimento	47
Figura 5.5 - Modelagem de um jogo educacional	48
Figura 5.6 - Básico para o HTML 5	50
Figura 5.7 - Diferenças entre as <i>tags</i> de texto	51
Figura 5.8 – Resultado do código do Quadro 5.3.....	52
Figura 5.9 – Resultado do exemplo do Quadro 5.4.....	53
Figura 5.10 – Exemplo do exemplo do Quadro 5.5	54
Figura 5.11 – Resultado do exemplo do Quadro 5.6.....	55
Figura 5.12 - Utilizando CSS externo	57
Figura 5.13 – Resultado do exemplo do Quadro 5.9.....	59
Figura 5.14 - Resultado ao clicar o botão.....	60
Figura 5.15 - Exemplo <i>Drag</i> - Tela Inicial.....	63
Figura 5.16 - Exemplo <i>Drag</i> - Número Gerado	64
Figura 5.17 - Exemplo <i>Drag</i> - Soltando o Número.....	65
Figura 5.18 - Amostra de Canvas	70

Índice de Quadros

Quadro 2.1 - Classificação dos jogos	7
Quadro 2.2 - Classificação de Callois	8
Quadro 2.3 - Uma proposta de classificação dos jogos sob uma perspectiva informativa.....	9
Quadro 3.1 - Alguns elementos de HTML 5	17
Quadro 3.2 - Alguns atributos globais	19
Quadro 3.3 - Algumas propriedades de CSS.....	22
Quadro 3.4 - Propriedades de CSS para posição e afastamento.....	26
Quadro 3.5 - Exemplo utilização do float:right.....	28
Quadro 3.6 - Implementando a tag <audio>	30
Quadro 3.7 - Inserindo JavaScript no HTML.....	34
Quadro 3.8 - Elementos para formulários	35
Quadro 3.9 - Alguns atributos para a tag <input>.....	35
Quadro 3.10 - Quadro de atributos globais de evento	36
Quadro 5.1 - O código básico para HTML 5	50
Quadro 5.2 - Código de exemplo para o uso de texto	51
Quadro 5.3 - Utilização da tag de imagem.....	52
Quadro 5.4 - Exemplo de utilização da tag de link	53
Quadro 5.5 - Utilizando CSS em linha.....	54
Quadro 5.6 - Utilização de CSS interno.....	55
Quadro 5.7 - Exemplo de CSS Externo.....	56
Quadro 5.8 - Importando arquivo CSS.....	56
Quadro 5.9 - Formulário de login e senha.....	58
Quadro 5.10 - Adicionando JavaScript ao formulário	59
Quadro 5.11 - Exemplo de uso de evento de mouse parte 1	61
Quadro 5.12 - Exemplo de uso de evento de mouse parte 2	62
Quadro 5.13 - Básico para a utilização do Canvas.....	66
Quadro 5.14 - Exemplo de utilização de Canvas parte 1	67
Quadro 5.15 - Exemplo de utilização de Canvas parte 2	68
Quadro 5.16 - Exemplo de utilização de Canvas parte 3	69

Lista de Abreviações

HTML5 - HyperText Markup Language (Linguagem de Marcação de Hipertexto)

CSS - Cascading Style Sheets (Folha de Estilos em Cascata)

URL - Uniform Resource Locator (Localizador-Padrão de Recursos)

API - Application Programming Interface (Interface de Programação de Aplicativos)

RPG - Role-playing game (Jogo de Interpretação)

www - World Wide Web (Teia Mundial)

HTTP - HyperText Transfer Protocol (Protocolo de Transferência de Hipertexto)

W3C - World Wide Web Consortium (Consórcio da Teia Mundial)

DOM - Document Object Model (Modelo de Objetos de Documentos)

SVG - Scalable Vector Graphics (Gráficos Vetoriais Escaláveis)

MMORPG - Massively Multiplayer Online Role Playing Game (Jogo online de multijogador em massa de interpretação)

Sumário

Assinaturas	iii
Agradecimentos.....	v
Índice de Figuras	vi
Índice de Quadros	vii
Lista de Abreviações	viii
Sumário	ix
Resumo.....	xi
Abstract	xii
1 Introdução	1
1.1 Objetivo	1
1.2 Motivação.....	2
2 Jogos na Educação	3
2.1 História	3
2.2 Importância e classificações.....	5
2.3 Qualidade em jogos.....	10
2.4 Considerações Finais	12
3 HTML 5.0	14
3.1 Definição	14
3.2 Representação dos elementos de HTML	16
3.3 Elementos básicos para uma página em HTML	16
3.3.1 <i>Tags</i> para Texto	18
3.3.2 <i>Tag</i> para Imagem	18
3.3.3 <i>Tag</i> para Hiperlinks	19
3.3.4 Atributos Globais.....	19
3.4 Utilizando CSS.....	20
3.4.1 Definindo Cores.....	23
3.4.2 Posicionando os elementos da página	23
3.4.3 Organizando os elementos.....	29
3.5 <i>Tag</i> para Áudio	29
3.6 Programando com JavaScript.....	30

3.6.1	Definição.....	31
3.6.2	Utilizando JavaScript	33
3.6.3	Criando formulários	34
3.6.4	Criando eventos	36
3.6.5	Usando números aleatórios	37
3.7	Elemento <i>Canvas</i>	37
4	Browserquest.....	39
4.1	Considerações Finais	41
5	Guideline.....	42
5.1	Profissionais de desenvolvimento de games	42
5.2	O Processo de Design de um Jogo.....	44
5.3	<i>Design</i> de Jogos Educativos – <i>Design</i> Instrucional	45
5.3.1	Análise e Planejamento.....	47
5.3.2	Modelagem	47
5.4	O Processo de Implementação de um Jogo	48
5.4.1	Programação do jogo	49
5.5	Testes	71
5.5.1	Testes com especialistas	71
5.6	Considerações Finais	72
6	Considerações Finais.....	73
6.1	Contribuições	73
6.2	Limitações	73
6.3	Trabalhos futuros	73
	Referências Bibliográficas	75

Resumo

O jogo exerce um papel importante na educação, pois na sua essência ele promove o prazer em aprender, com base na diversão, na qual o aluno desperta maior interesse e motivação no processo de aprendizagem. Na amplitude da sua aplicação, o jogo vem colaborando nas diversas áreas do conhecimento. O jogo se torna mais interessante quanto mais as experiências vivenciadas através dele estiverem próximas da realidade.

Existe a importância de explorar seus componentes, pois o jogo agrega a possibilidade de formar indivíduos criativos, críticos e autônomos para atuarem na sociedade. Compreende-se, também, ser necessário que o educador tenha conhecimentos da sua utilização para que seja incorporada nas suas práticas com objetividade e essência.

O presente trabalho tem por objetivo mostrar a importância e ação dos jogos educacionais no processo de aprendizagem e educação. Além disso, seu foco é desenvolver um estudo prático do processo da criação de um jogo para web utilizando a linguagem HTML 5, resultando em um *Guideline* para estudos posteriores. Dessa forma, o material servirá como fonte bibliográfica para a realização de todas as etapas da produção de um jogo para estudantes, profissionais, acadêmicos e educadores.

Palavras Chaves: Jogos Educativos, HTML5, HTML, CSS, JavaScript.

Abstract

Playing games has an important role in education because in their essence, games promote pleasure in learning, based on fun. This helps students to arouse a greater interest and motivation in the learning process. In the extent of their application, games collaborate in various areas of knowledge. They become more interesting as closer to the reality the experiences lived through them are.

It is important to explore games' components, because they help to educate creative, critics and independent individuals to act on society. In addition, it is understandable that it is necessary educators have the knowledge of its use, to incorporate them into their practices with objectivity and essence.

This work aims to show the importance and action of educational games in learning and education. In addition, the focus of this project is to develop a practical study of web games development using HTML 5, resulting in a Guideline for further studies. Thus, this material will serve students, professionals, academics and educators as a bibliographic source for all the steps to the development of a game.

Key words: Educational Games, HTML5, HTML, CSS, JavaScript.

1 Introdução

Os avanços tecnológicos fazem com que os jogos tenham cada vez mais importância didática sobre a educação, na sociedade. Jogar é uma atividade gratificante que serve também para motivar o auto aprendizado e o aprendizado em conjunto com os colegas. Com a sua utilização na educação é constatado que o jogo é fundamental para a construção do pensamento da criança e para a aquisição da leitura, da escrita e do raciocínio lógico [VILELA 2011].

Os jogos participam da vida das pessoas não só na fase da infância, mas também em outros momentos. Eles podem ser usados como ferramentas instrucionais eficientes que divertem enquanto motivam o aprendizado. Têm atuado efetivamente na facilitação do aprendizado e no aumento da capacidade de retenção do que foi ensinado, exercitando principalmente as funções mentais e intelectuais do jogador [HAGUENAUER et al. 2007]. Em sua dinâmica, os jogos despertam nos jogadores a autonomia, a criatividade, a originalidade e a possibilidade de simular e experimentar situações potencialmente perigosas e proibitivas do nosso cotidiano [TAROUCO et al. 2004; FALKEMBACH et al. 2006].

Recentemente, os criadores do browser de internet Firefox, Mozilla, desenvolveram um jogo utilizando HTML 5 [VALENTYN 2012] e JavaScript [ROUGET 2012], o Browserquest. Ele é uma demonstração convincente de como a tecnologia pode ser usada para criar um jogo de browser. O jogo utiliza o elemento Canvas para renderizar o mundo 2D, API de áudio em HTML 5 para efeitos sonoros, WebSockets para facilitar a comunicação com o servidor *backend* e localStorage para armazenar o progresso dos jogadores [PAUL 2012].

1.1 Objetivo

Este trabalho tem como objetivo desenvolver um *Guideline* para a produção de jogos educativos visando seu uso na plataforma web, utilizando a linguagem HTML 5, recentemente aprimorada [LUCENA 2012]. Serão mostrados a motivação e os passos necessários para o desenvolvimento de um jogo.

1.2 Motivação

Os jogos educacionais são motivadores da aprendizagem, proporcionando um estudo gratificante e fazendo com que o aluno estude sem perceber e com que articule tanto a prática como a teoria. Ainda que os jogos venham desempenhando um papel importante no processo educacional, há muita discussão sobre o que são jogos educacionais. DEMPSEY, RASMUSSE e LUCCASSEN (1996) citados por BOTELHO (2004) definem que os jogos educacionais “se constituem por qualquer atividade de formato instrucional ou de aprendizagem que envolva competição e que seja regulada por regras e restrições”.

Com o seu uso, são estimuladas inúmeras habilidades como: persistência nos desafios, desenvolvimento de tarefas, aumento do raciocínio lógico para resolver problemas, estimulando os jogadores enquanto se divertem e isso faz com que a aprendizagem seja potencializada. As características que os tornam educativos são o desafio, a fantasia e a curiosidade.

“Os jogos, sob a ótica de crianças e adolescentes, se constituem a maneira mais divertida de aprender” [TAROUCO et al. 2004; FALKEMBACH et al. 2006].

Entretanto, o processo de criação e desenvolvimento de um jogo pode se tornar um entrave e um desafio nos projetos tanto para profissionais quanto estudantes e demais interessados em jogos. Isso ocorre pois tal processo requer várias etapas que podem ser exaustivas e difíceis, afastando pessoas que apenas assimilam o entretenimento final de um jogo. Para facilitar a compreensão e a execução de suas etapas fundamentais, a utilização de um *guideline* pode contribuir significativamente para a compreensão dessas etapas de sua criação, bem como da definição dos passos necessários para o desenvolvimento dos referidos jogos.

2 Jogos na Educação

O papel dos jogos, desde a antiguidade até os dias de hoje, vem se mostrando de grande importância na educação, produzindo ambientes de interação entre os indivíduos e promovendo a socialização, a diversão, a alegria, o prazer e a descontração, atuando na construção do pensamento, visando aquisição da leitura, da escrita e do raciocínio lógico-matemático. Brincar é uma atividade não só das crianças, mas também dos adultos, desde o início da civilização. “A brincadeira se faz presente na vida adulta das pessoas em forma de jogos” [WAJSKOP, 1995].

2.1 História

Os primeiros jogos direcionados à educação e ao desenvolvimento se deram início em 3.000 a.C., sendo aplicados na China para a preparação de combates de guerra. Passado o período das guerras, suas aplicações saíram da competição no campo de batalha para aplicação de negócios, onde se deu origem aos jogos empresariais [ROSA E AZUAYA, 2006].

Já na Grécia antiga, Aristóteles (385-322 a.C.) incentivava aplicação dos jogos na preparação da criança para a vida adulta, eles eram comparados à felicidade e à virtude. Platão (427-347 a.C.) divulgava o valor e a importância de apreender brincando, em oposição à prática da violência e da repressão, sendo necessário incentivar tais práticas para as crianças desde pequenas [FONTOURA & PEREIRA 2010]. Nessa época os jogos foram limitados na qualidade de recreação, sua importância era destinada a preparos físicos, formação estética e espiritual, mas não destinados ainda para o ensino da leitura e do cálculo [KISHIMOTO 1999].

Na Idade Média, não houve uma expansão dos jogos na educação, nas salas de aulas a educação era disciplinadora sendo imposta de dogmas, nos quais os alunos tinham que fazer silêncio absoluto e o professor possuía um domínio autoritário, sendo assim: “Os jogos foram considerados delituosos, à semelhança da prostituição e embriaguez” [KISHIMOTO 1999; TEIXEIRA 2012].

No século XVI, com a origem do Instituto dos Jesuítas, chegaram os jogos educativos publicados por um dos líderes dessa instituição educacional, Ignácio de Loyola. A meta era engrandecer as ações didáticas, por meios de jogos de exercícios de

caráter lúdico, onde as crianças vivenciaram uma metodologia educacional diferente do ensino escolástico [KISHIMOTO 1999].

Já na época do Renascimento, surgem novas concepções pedagógicas sendo que nesse período foram consideradas as possibilidades dos jogos passarem a ser utilizados para a educação, sendo vetados alguns jogos e foram classificados como “maus”. [WAJSKOP 1995].

Ao longo dos séculos XVII e XVIII, porém, estabeleceu-se um compromisso que anunciava a atitude moderna com relação aos jogos, fundamentalmente diferente da atitude antiga. Esse compromisso nos interessa aqui porque é também um testemunho de um novo sentimento da infância: uma preocupação, antes desconhecida, de preservar sua moralidade e também de educá-la, proibindo-lhe os jogos então classificados como maus, e recomendando-lhe os jogos então reconhecidos como bons. (ARIÈS 1981, p. 59).

Essa classificação, que agregou ao século XVII, atribuiu devido às concepções dos adultos sobre a infância em uma atitude moral contraditória com relação aos jogos e as brincadeiras [ARIÈS 1981]. Ainda segundo ARIÈS, a educação das crianças nesta época era de desenvolver no homem um espírito livre. Na primeira infância eles teriam que estar aptos para coordenar todas as matérias da educação, desde a arte até a ciência, tendo cautela de conservar as brincadeiras e jogos, para resguardar a moralidade das crianças, tidas até então, como adultos em miniaturas.

Por volta de 1600, a especialização das brincadeiras atingia apenas a primeira infância; depois dos três ou quatro anos, ela se atenuava e desaparecia. A partir dessa idade, a criança jogava os mesmos jogos e participava das mesmas brincadeiras dos adultos, quer entre crianças, quer misturada aos adultos. (ARIÈS 1981, p. 49).

Junto ao Século XVIII, o jogo começa a ganhar espaço e valorização no âmbito educacional na medida em que a educação começa a se adaptar à natureza infantil. Com o início do século XIX, que foi marcado com o término da Revolução Francesa, surgiram novas práticas pedagógicas. Nessa época os princípios dos pedagogos ROUSSEAU, PESTALOZZI e FROEBEL foram colocadas em prática. Na concepção de ROUSSEAU, o brinquedo tinha duas qualificações, o objeto e a ação de brincar e a atenção era mais na ação do sujeito por estabelecer uma relação conectada pela inteligência. PESTALOZZI procurou estudar a ação mental da criança. Mas é com FROEBEL que o jogo passa ser parte da história da educação pré-escolar. Fez uso de materiais como bola, cubo e cilindro, para a criança estabelecer relações matemáticas adquirindo noções primárias de Física e Metafísica [KISHIMOTO 1999].

Montessori (1870 - 1952) e Décroly (1871- 1932) contribuíram muito para o ensino da matemática, sendo os primeiros pedagogos a romper com a educação tradicionalista e utilizando jogos e materiais didáticos para uma educação sensorial, transformando em uma educação natural com instintos infantis [WAJSKOP 1999]. Montessori também incorpora os materiais criados por ITARD e SÉGUIN em sua metodologia de ensino, que foi dirigida aos deficientes mentais. Na mesma época, DECROLY, organizava seus jogos intelectuais e motores, também destinados às crianças deficientes mentais. Atualmente crescem as pesquisas, as utilizações e a produção dos brinquedos e dos jogos, para atender às diferentes formas de deficiências da criança [KISHIMOTO 1999].

Desde as primeiras épocas até os tempos atuais, observa-se uma crescente valorização do uso dos jogos na educação como recursos didáticos e sua evolução nos processos de aprendizagem e recreações.

2.2 Importância e classificações

Os Jogos podem resgatar o desejo de conhecimento, tornando a aprendizagem prazerosa e são importantes na vida da criança. Enquanto joga, ela aprende com liberdade e não como obrigação imposta, passando a gostar cada vez mais de aprender. Jogos equilibram seus mundos externos e internos, concentrando suas energias e modificando em prazer, as suas angústias. Sendo assim, proporcionam um estudar gratificante representando em vitória os seus esforços. Os jogos educativos são planejados para divertir e aumentar a chance de aprendizagem de conceitos, conteúdos e habilidades que estão neles embutidos [BARBOSA NETO 2012]. Jogar é participar do mundo de faz de conta, dispor-se às incertezas e enfrentar desafios, traçar estratégias em busca de soluções, atuam no imaginário na qual o abstrato se concretiza, resultando num entretenimento e de um novo conhecimento.

O jogo tem potencial de preparar o indivíduo para o trabalho também, ajudando a inseri-lo no grupo social, quando entrando em contatos com outros faz cumprir suas funções dentro da equipe. Conforme MARRAS (2009, p. 315) citado por LEWINSKI (2011) “quando os indivíduos trabalham em conjunto e compartilham responsabilidades obtém-se como resultado indivíduos comprometidos e envolvidos na solução de problemas”. Como exemplo, os jogos de empresas que são atividades lúdicas, das quais ensinam técnicas, desenvolvem habilidades de relacionamento entre os acadêmicos e

comunicações variadas [LEWINSKI 2011]. Eles podem simular um ou vários departamentos ou setores da economia, com o objetivo de fornecer conhecimento das complexidades das ferramentas e das decisões para os participantes [LEWINSKI 2011].

Os jogos são ferramentas instrucionais, pois eles divertem e motivam, simplificando então o aprendizado, pois aumentam a retenção do que foi ensinado. Com isso eles estimulam e desenvolvem estruturas cognitivas do cérebro, despertando nos jogadores as habilidades de observar e identificar, comparar e classificar, conceituar, relacionar e inferir, além de desenvolver a autonomia, criatividade, originalidade e também permitem simular e experimentar situações potencialmente perigosas e proibitivas do cotidiano das pessoas [TAROUCO et al. 2004; FALKEMBACH et al. 2006].

Conforme SILVEIRA (1998), apud HAGUENAUER et al. (2007):

(...) os jogos podem ser empregados em uma variedade de propósitos dentro do contexto de aprendizado. Um dos usos básicos e muito importantes é a possibilidade de construir-se a autoconfiança. (...) Até mesmo o mais simplório dos jogos pode ser empregado para proporcionar informações factuais e praticar habilidades, conferindo destreza e competência". (...) Os jogos educativos podem despertar no aluno: motivação, estímulo, curiosidade, interesse em aprender (...) o aluno constrói seu conhecimento de maneira lúdica e prazerosa".

O uso dos jogos na educação vem tornando a aprendizagem mais eficiente e seu uso possui um alto valor educativo eficaz e motivador. Mas é importante ressaltar que os jogos não podem ser feitos sem nenhum conhecimento prévio e devem estar atrelado a princípios teórico-metodológicos claros e bem fundamentados. Por isso há necessidade do educador ter conhecimento de como a utilização de um jogo pode ser incorporada às suas práticas com objetividade, mantendo a importância de explorar seus componentes. Neste sentido, ANTUNES (1998, p.37) apud SULZBACH (2013):

Jamais pense em usar os jogos pedagógicos sem um rigoroso e cuidadoso planejamento, marcado por etapas muito nítidas e que efetivamente acompanhem o progresso dos alunos, e jamais avalie qualidade de professor pela quantidade de jogos que emprega, e sim pela qualidade dos jogos que se preocupou em pesquisar e selecionar.

Atualmente jogar em sala de aula proporciona momentos ricos de interações na aprendizagem, auxiliando educadores e educandos no processo da educação.

Os Jogos podem ser divididos em não acadêmicos (base só para diversão) e acadêmicos (base para aprendizagem). Segundo CRUICKSHANK (1980), citado por

VASCONCELLOS (2008) cita:

Os jogos aparecem classificados segundo sua configuração e atitude psicológica em quatro grandes grupos. Para cada um dos grupos Ihote propõe também um antijogo. Por antijogo entendemos a atitude física ou mental que é impeditiva da experiência proposta pelo jogo. E assim, por exemplo, caretas e sinais informam o que deveria ser um prazer descobrir nos jogos de enigmas. Trapaças impedem a sensação de risco que os jogos de sorte ou revés deveriam conter. A confusão impede que jogos de território se atenham às suas regras – é só lembrar o que acontece quando a torcida invade o campo... E por fim, se a função de uma boneca é representar o esquema corporal para que sejam nomeados cabeça, tronco e membros, ela não é capaz de ser, simultaneamente, a “filhinha” dos jogos de casinha.

O quadro 2.2 mostra os quatro grupos propostos por Ihote e exemplos de jogos que se encaixam em cada grupo.

Quadro 2.2 - Classificação de Callois

<p>O Prazer de Estar Junto <i>Jogos onde as figuras se formam ao jogar</i> Antijogo: Caretas</p>		<p>A Sorte ou Revés <i>Jogos onde a forma é uma figura móvel</i> Antijogo: Trapaças</p>	
<p>Jogos corporais</p> <ul style="list-style-type: none"> • Rodas • Ginástica 	<p>jogos de ilusão</p> <ul style="list-style-type: none"> • prestidigitação • ilusão ótica 	<p>Psicológica</p> <ul style="list-style-type: none"> • pôquer • morra 	<p>adivinhação</p> <ul style="list-style-type: none"> • dados • loterias
<p>Jogos de imitação</p> <ul style="list-style-type: none"> • construção • quebra-cabeças 	<p>jogos de inteligência</p> <ul style="list-style-type: none"> • enigmas • truques 	<p>Destreza</p> <ul style="list-style-type: none"> • três marias • bilhar 	<p>estratégia</p> <ul style="list-style-type: none"> • bridge • dominós
<p>A Magia dos Objetos <i>Jogos onde a forma é uma figura fixa</i> Antijogo: Jogos educativos</p>		<p>A Ordem do Mundo <i>Jogos onde as figuras são ligadas a um terreno</i> Antijogo: Caos</p>	
<p>Figuras humanas</p> <ul style="list-style-type: none"> • boneca • máscaras 	<p>imagem do mundo</p> <ul style="list-style-type: none"> • pião • bola 	<p>Percurso</p> <ul style="list-style-type: none"> • amarelinhas • jogo do ganso 	<p>localização</p> <ul style="list-style-type: none"> • loto • casa da fortuna
<p>Modelos reduzidos</p> <ul style="list-style-type: none"> • soldadinhos • carrinhos 	<p>formas litúrgicas</p> <ul style="list-style-type: none"> • chocalhos • balanço 	<p>Combate</p> <ul style="list-style-type: none"> • boxe • futebol 	<p>combinação</p> <ul style="list-style-type: none"> • xadrez • cubos mágicos

Fonte: IHOTE (1976, p. 37)

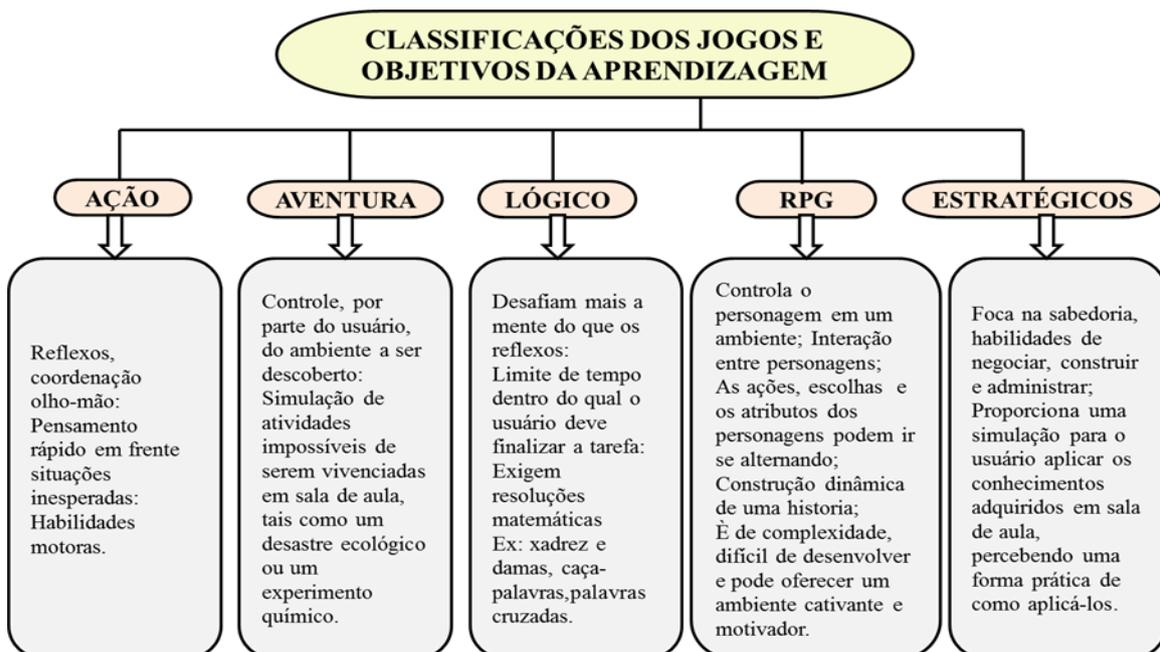
O quadro 2.3 demonstra uma outra forma de classificação dos jogos, que é mais utilizado atualmente e, assim como a Figura 2.1, informa quais habilidades cada grupo de jogos ajuda a desenvolver no jogador.

Quadro 2.3 - Uma proposta de classificação dos jogos sob uma perspectiva informativa

CATEGORIAS	SUB-CATEGORIAS	JOGOS	GÊNEROS	SUPORTES	HABILIDADES	INTERATIVIDADE
SIMULADORES	Voo, corrida, construção, negócios	<i>Combat flight simulator, Racing game, The Sims</i>	Ação, aventura, educacional, esporte	On-line, computador, portátil, mobile, console	viso-motora/ reação física/ Pensamento reversível/simulação matemática	UM-TODOS, UM-UM, TODOS-TODOS - (Nível de Interação básico, intermediário e elevado) sem e com metas definidas, sem e com vencedor ou perdedor; narrativas abertas e negociadas com um grupo
ESTRATÉGIA	(Enredo) RPG, MUD's, quebra-cabeças (puzzle) (Lógica abstrata) cartas, demais quebra-cabeças, tabuleiro, ETR	<i>The Sims, Civilization, God games, The Legend of Zelda, xadrez</i>	Aventura, casual, educacional	On-line, computador, mesa, portáteis, mobile, console	Planejamento estratégico/ Pensamento reversível/ envolvimento, espontaneidade/ simulação matemática	UM-TODOS, UM-UM, TODOS-TODOS - (Nível de Interação básico, intermediário e elevado) sem e com metas definidas, sem e com vencedor ou perdedor; narrativas abertas e negociadas com um grupo
AÇÃO	Corrida, futebol	<i>GTA, FIFA, Counter Strike, MOH</i>	Guerra, esporte, luta	On-line, computador, portátil, mobile, console	Coordenação viso-motora/ reação física/ descoberta e a aprendizagem	UM-TODOS - (Nível de interação básico e intermediário) metas definidas, regras limitadas e jogadas restritas.
RPG	Quebra-cabeças, jogos de cartas, de mesa, tabuleiro	<i>MU, Everquest, WOW, Dungeons & Dragons</i>	Aventura, casual, MMORPG	On-line, computador, mesa, mobile console	Planejamento estratégico/ Pensamento reversível/ simulação matemática	UM-TODOS, UM-UM, TODOS-TODOS - (Nível de Interação básico, intermediário e elevado) sem e com metas definidas, sem e com vencedor ou perdedor; narrativas abertas e negociadas com um grupo
ARCADE	Jogos de console	<i>Space Invaders, Star War</i>	Aventura, guerra, luta, ação	Máquinas de arcade - fliperama	Coordenação viso-motora/ reação física	UM-TODOS - (Nível de interação básico) metas definidas, regras limitadas e jogadas restritas.
MUSICAL	Aprendizagem, educacional	<i>Guitar Hero, PopStar Guitar, Rock Band</i>	Musical	Console (PC e on-line)	Coordenação viso-motora/reação física/ descoberta/ aprendizagem	UM-TODOS (Nível de Interação básico) metas definidas, regras limitadas e jogadas restritas.
NEWSGAMES	Editorial games, mashups, infografia interativa, jogos de análise de notícias	<i>September 12th, Shoes in Bush, Madri</i>	Notícia, informação, Jornalismo	On-line (PC e mobile)	Pensamento reversível/ envolvimento, espontaneidade, agenciamento	TODOS-TODOS - (Nível de Interação elevado) abertura das narrativas; podem ser combinadas, discutidas e negociadas com um grupo maior de jogadores
ADVERGAMES	Marketing viral, Marketing estratégico, Marketing político	<i>Sete Zoom, Zona Incerta, Descubra o Segredo</i>	Alternate Reality Games (ARG) Publicidade e Propaganda	On-line (PC e mobile)	Pensamento reversível, envolvimento, espontaneidade	UM-TODOS (Nível de interação básico) metas definidas, regras limitadas e jogadas restritas.
SERIOUS GAME	Edutainment, Militainment, administração, política, urbanismo	<i>Army Battlezone, Cola-Cola</i>	Jogos sérios, educacionais, Relações Públicas	On-line (PC e mobile), computador, mesa, console	Planejamento estratégico/ Pensamento reversível/ envolvimento, espontaneidade	UM-TODOS (Nível de interação básico) metas definidas, regras limitadas e jogadas restritas.

Fonte: SANTOS (2012)

Figura 2.1 - Classificação dos Jogos e objetivos da Aprendizagem



Fonte: adaptado de TAROUCO et al. (2004)

Atualmente aprender com os jogos tornou o aprendizado gratificante, e para que isso ocorra os jogos precisam ser claros, simples e fáceis. É importante o educador após o término, chegar a uma conclusão com seus alunos, pondo em discussão o conteúdo pedagógico que o jogo explorou e assim poderá avaliar tanto o jogo, como também para fixar melhor o aprendizado que o jogo está propondo. Há pesquisas que mostram que os alunos que jogam em um ambiente de classe são mais entusiasmados e motivados aos que joga isolado longe da escola [BARBOSA NETO 2012].

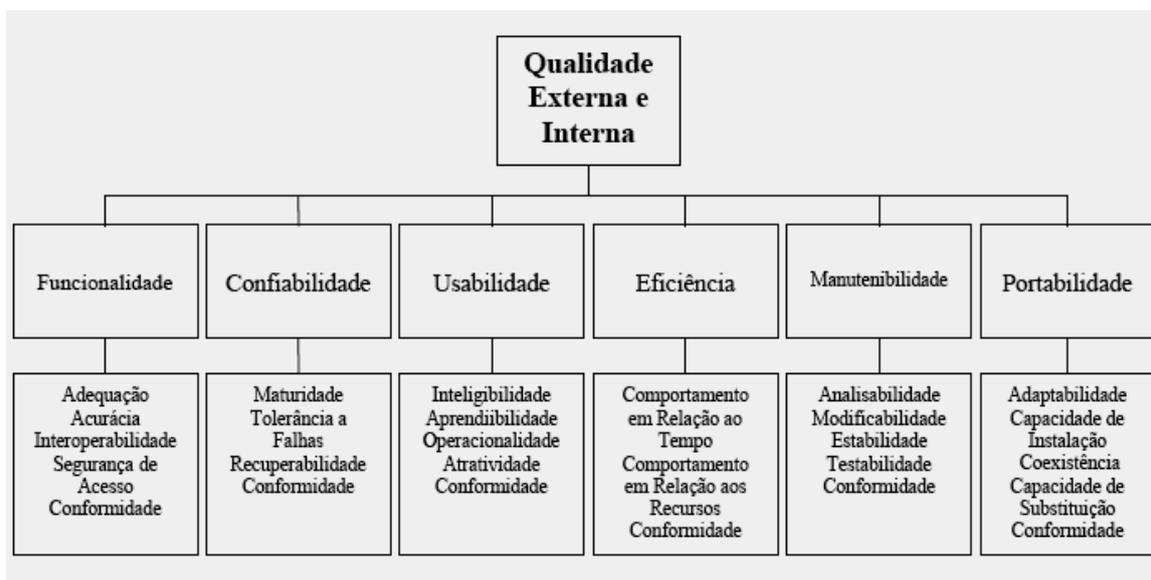
2.3 Qualidade em jogos

Com o uso dos computadores numa variedade de áreas, desenvolver ou selecionar produtos de software de alta qualidade é de primordial importância para o sucesso de negócios e para a segurança humana. Deste modo, especificar e avaliar a qualidade do produto de software são fatores chave para garantir uma qualidade adequada. Isto pode ser alcançado pela definição apropriada das características de qualidade, levando em consideração o uso pretendido do produto de software. É importante que cada característica relevante de qualidade do produto de software seja especificada e avaliada, utilizando quando possível, métricas validadas ou amplamente aceitas (NBR ISO/IEC 9126-1: 2003).

Atualmente como em todos os produtos fabricados, os jogos educativos devem obedecer à NORMA ISO/IEC 9126, em 1991, foi publicada a norma ISO/IEC 9126 contendo características e sub características que definem um produto de qualidade. Após as revisões e sucessivas melhorias foram criadas divisões dessa norma: ISO/IEC 9126-1: Modelo de Qualidade.

A norma ISO/IEC 9126-1 apresenta um conjunto de características que definem um modelo de qualidade que pode ser aplicada a qualquer produto de software. Na visão de PAPANIKOLAOU & MAVROMOUSTAKOS (2007), citado por BARBOSA NETO (2012) diz que qualidade de jogos educativos deve aderir aos parâmetros definidos pela norma ISO 9126 [ISO 2012]. Esses parâmetros, ilustrados na figura 2.2, são definidos como modelo de qualidade para qualidade interna e externa, possuindo definições de seis características básicas que um produto de software deve ter para ser considerado um software de qualidade: funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade.

Figura 2.2- Características e Sub características de Qualidade Externa e Interna



Fonte: MACHADO & SOUZA (2011)

Segue de MACHADO (2011), o detalhamento das características da NBR ISO/IEC 9126-1:2003.

1. **Funcionalidade:** capacidade de fornecer funções que correspondam às necessidades explícitas e implícitas do usuário quando o software é utilizado sob condições especificadas.
 - Adequação: capacidade de fornecer um conjunto apropriado de funções para tarefas específicas e objetivas do usuário.
 - Acurácia: capacidade de fornecer o resultado com o grau de precisão desejado.
 - Interoperabilidade: capacidade de interagir com um ou mais sistemas.
 - Segurança de Acesso: capacidade de proteger dados e informações de pessoas ou sistemas não autorizados.
 - Conformidade: capacidade de aderir a padrões, convenções, leis e prescrições similares relativas a funcionalidade.
2. **Confiabilidade:** capacidade de o software manter seu nível de desempenho quando utilizado em condições estabelecidas.
 - Maturidade: capacidade de evitar defeitos no software.
 - Tolerância a Falhas: capacidade de manter um nível de desempenho estabelecido em caso de defeito no software.
 - Recuperabilidade: capacidade de recuperar dados diretamente afetados no caso de falhas.
 - Conformidade: capacidade de aderir a padrões, convenções, leis e prescrições similares relativas a confiabilidade.
3. **Usabilidade:** capacidade que o produto tem de ser entendido, aprendido, utilizado e ser atraente para o usuário.

- Inteligibilidade: capacidade do produto de fazer o usuário entender se o software é adequado, e como ele pode ser usado para tarefas particulares.
 - Aprendibilidade: capacidade que o produto deve ter de fazer o usuário entendê-lo.
 - Operacionalidade: capacidade que o produto deve ter para que o usuário possa aprendê-lo e controlá-lo.
 - Atratividade: capacidade do produto em ser atraente para o usuário.
 - Conformidade: capacidade de aderir a padrões, convenções, leis e prescrições similares relativas a usabilidade.
4. **Eficiência:** relacionamento entre o nível de desempenho do software e a quantidade de recursos utilizados, sob condições estabelecidas.
- Comportamento em Relação ao Tempo: capacidade de fornecer tempos de resposta e processamento adequados, bem como taxas de transferência.
 - Comportamento em Relação aos Recursos: capacidade de usar quantidade e tipos de recursos adequados.
 - Conformidade: capacidade de aderir a padrões e convenções relativas a eficiência.
5. **Manutenibilidade:** esforço necessário para se fazer modificações específicas no software.
- Analisabilidade: capacidade em diagnosticar deficiências e causas de defeitos.
 - Modificabilidade: capacidade que o produto tem de receber modificações.
 - Estabilidade: capacidade de evitar efeitos inesperados a partir de modificações.
 - Testabilidade: capacidade de validar as modificações efetuadas no produto.
 - Conformidade: capacidade de aderir a padrões e convenções relativas a manutenibilidade.
6. **Portabilidade:** capacidade que o produto tem de ser transferido de um ambiente para outro.
- Adaptabilidade: capacidade de ser adaptado em diferentes ambientes sem intervenção.
 - Capacidade de Instalação: capacidade de ser instalado em um ambiente específico.
 - Coexistência: capacidade que o produto tem de coexistir com outro software independente em um ambiente comum, compartilhando recursos comuns.
 - Capacidade de Substituição: capacidade que o produto de software deve ter de ser usado no lugar de outro produto de software com o mesmo propósito no mesmo ambiente.
 - Conformidade: capacidade de aderir a padrões e convenções relativas a portabilidade.

2.4 Considerações Finais

Nesse capítulo, foi mostrado a importância que os jogos tem na educação, pois além de divertir e descontrair, eles também ensinam e desenvolvem habilidades nos jogadores. Foi mostrado também que, durante o decorrer da história da humanidade, houve momentos em que as pessoas usaram jogos e brincadeiras para algum tipo

específico de ensino ou treinamento, como na China Antiga, para treinamentos militares e como no séc. XVI com o Instituto dos Jesuítas para engrandecer as ações didáticas.

Foi apresentado, nesse capítulo, algumas formas de classificar os jogos e o que cada grupo de jogos contribuem para o desenvolvimento do jogador. Essa informação auxiliará na hora de escolher qual o tipo de jogo será usado para atingir o que se deseja ensinar. Como exemplo, pode ser usado um jogo de simulação para mostrar como a física se comporta em um determinado cenário.

Assim como qualquer produto desenvolvido para o uso público, um jogo também deve seguir um nível de qualidade. Foi apresentado nesse capítulo, alguns requisitos de qualidade que um jogo deve possuir para atingir o sucesso. Um jogo que possui um péssimo desempenho, por exemplo, dificilmente atrairá um público interessado.

No capítulo a seguir, será apresentado as ferramentas que serão utilizadas para esse *guideline*. Será definido HTML, alguns de seus elementos, CSS, suas propriedades e JavaScript,

3 HTML 5.0

3.1 Definição

Conforme o *Linux Dictionary V 0.16*, do autor NGUYEN, HTML é definido como um conjunto de convenções para marcação de partes de um documento para que, quando acessado por um programa analisador, cada parte apareça com um formato distinto. HTML significa *HyperText Markup Language* e é a linguagem de autoria usado para criar documentos na *World Wide Web*.

O HTML foi originalmente desenvolvido por Tim Berners-Lee e ganhou destaque, impulsionado pelo navegador Mosaic¹, desenvolvido por Marc Andreessen, na década de 1990. A partir de então, desenvolvedores e fabricantes de browsers passaram a utilizar o HTML como base, compartilhando as mesmas convenções.

Em 1945, Bush propôs pela primeira vez os princípios do hipertexto, que lançou as bases para Tim Berners-Lee e outros para inventar a *World Wide Web*, HTML, HTTP (*HyperText Transfer Protocol*) e URLs (*Universal Resource Locator*) em 1990.

HTML é usado para definir a estrutura e layout de uma página da Web, como uma página de procura e de quaisquer funções especiais. HTML faz isso usando o que são chamados de *tags* que possuem atributos. Por exemplo <p> significa uma quebra de parágrafo. O programa analisador para acessar documentos é chamado de Web browsers, em português, navegadores da web. Com o HTML é possível incluir capacidades que possibilitam os autores de inserirem hyperlinks, que quando acionados mostram outras páginas do documento HTML.

As versões HTML+, HTML2.0 e HTML3.0, surgiram entre 1993 e 1995, e após diversas mudanças visando enriquecer as possibilidades da linguagem. Em 1997, o grupo de trabalho do W3C, trabalhou na versão 3.2 da linguagem, fazendo com que ela fosse padronizada como prática comum.

De acordo com a W3C, fundada em 1994 por Tim Berners-Lee, cuja sigla é uma abreviação do *World Wide Web Consortium*, um consórcio de companhias internacionais que englobam a Internet e a Web, HTML5 é a uma especificação que define a 5ª revisão do *Hypertext Markup Language* (HTML) ou Linguagem de Marcação de Hipertexto. O HTML5 apresenta nesta revisão uma mudança que configura o endereçamento nas aplicações Web incluindo funções específicas que

¹ <http://www.ncsa.illinois.edu/Projects/mosaic.html>, Acessado em Setembro de 2013

incorporam gráficos, áudio, vídeo e documentos interativos. Também pode ser referenciado como *Web Applications 1.0*.

O HTML5 é o sucessor do HTML 4.01, cuja versão havia sido lançado inicialmente em 1999. Desde então a Internet mudou significativamente e a criação do HTML5 passou a ser uma necessidade. A nova linguagem de marcação foi baseada em padrões pré-estabelecidos, tais como:

- Novas configurações baseadas em HTML, CSS, DOM, e JavaScript;
- A redução da necessidade de conexões externas, ex.: Flash;
- Manuseio de erros precisava ser mais fácil que as versões anteriores;
- O script tinha que ser substituído por mais marcação;
- HTML5 deveria ser um dispositivo independente; e
- O processo de desenvolvimento deveria ser visível para o público.

O HTML5 na versão teste foi primeiramente apresentado em 2008, porém seu lançamento para o público somente aconteceu em 2011. Atualmente os principais browsers (Chrome, Safari, Firefox, Opera, IE) têm suporte em HTML5.

Com o HTML5 é possível criar novas *tags* e modificar a função de outras, com um padrão universal para a criação de seções comuns e específicas como rodapé, cabeçalho, barras laterais, menus, padrão de nomenclatura de ID, Classes ou *tags* e capturar de maneira automática as informações localizadas nos rodapés dos websites.

HTML5 juntamente com o CSS e Javascript são ferramentas que além de oferecer desenvolvimento de experiências incríveis na *web* também permitem criação de aplicativos móveis fantásticos, causando até mesmo a sensação real de um aplicativo nativo.

Com o HTML5 foi possível eliminar a necessidade de plug-ins para aplicações multimídia, antes só possível através de terceiros como Flash², da Adobe, o Silverlight³, da Microsoft ou o recente JavaFX⁴, da Oracle. A maior de todas as vantagens do HTML5 é seu custo zero (*Open-source*) para implementação, não havendo necessidade de pagamento de taxas, licenças ou *royalties*, gerando assim uma economia enorme para as empresas de desenvolvimento de jogos, aplicativos, entre outros.

² <http://www.adobe.com/products/flash.html>, Acessado em Setembro de 2013

³ <http://www.microsoft.com/brasil/Silverlight/>, Acessado em Setembro de 2013

⁴ <http://www.oracle.com/us/technologies/java/fx/overview/index.html>, Acessado em Setembro de 2013

3.2 Representação dos elementos de HTML

Como foi dito anteriormente, HTML, inclusive a quinta versão, utiliza-se de tags para representar seus elementos. As tags são representadas por um nome (que identifica a tag) entre “<” e “>”, como por exemplo, a tag <p>, que representa o elemento de parágrafo. As tags são compostas de uma tag de abertura e uma tag de fechamento e todo elemento que estiver entre essas duas tags fará parte do conteúdo da tag. Aproveitando o exemplo anterior, a tag <p> é a tag de abertura e a tag </p> é representação da tag de fechamento. Existem algumas exceções, que são chamadas de tags vazias, como a tag
, usada para pular para outra linha, que não possui tag de fechamento e conseqüentemente não possuem conteúdo [W3SCHOOLS 1999-2013].

A maioria das tags possui atributos, que provém informações adicionais para a tag. Como exemplo, tem-se a tag <a> que representa um link para uma outra página na Internet. A tag possui um atributo href, que diz qual é o endereço da página na Internet que o link levará, por exemplo, Link para um site de referência para programar em HTML. Nota-se que o valor do atributo href vem entre aspas e a frase entre as tags de abertura e fechamento é o conteúdo da tag [W3SCHOOLS 1999-2013].

3.3 Elementos básicos para uma página em HTML

A tag <!DOCTYPE> define o tipo de documento e o doctype html indica que o documento é de HTML 5. A tag <html> representa a página da web em si. Na tag <head> será colocado todo o código de JavaScript e CSS, ou indicará ao browser onde os encontrar, meta-informações, e outros elementos, normalmente não visíveis em uma página. Já na tag <body> serão colocados todos os elementos que aparecem no corpo da página.

Dentro da tag <head> entram tags como a tag <meta>, que pode, por exemplo, definir a tabela de caracteres que será usada para interpretar o conteúdo do site. A tabela mais comum atualmente é o Unicode, que é padrão para o mundo inteiro, garantindo que o site será bem visualizado em qualquer lugar. A tag <title>, que é outra tag que é colocada dentro da tag <head>, define o título da página, o qual toda página na web possui e o texto colocado no seu conteúdo, será o seu título.

O quadro 3.1 mostra alguns elementos de HTML 5 e o que eles representam. Eles podem ser colocados no conteúdo da tag *<head>*, como o caso do *<style>*, ou da tag *<body>*, como o caso do **.

Quadro 3.1 - Alguns elementos de HTML 5

Tag	Descrição
<code><!-- ... --></code>	Define um comentário no código
<code><a></code>	Define um link
<code><audio></code>	Define um conteúdo sonoro
<code></code>	Define um texto em negrito
<code><body></code>	Define o corpo do documento
<code>
</code>	Define uma quebra de linha/Pula uma linha
<code><button></code>	Define um botão clicável
<code><canvas></code>	Define uma área para desenhar via script
<code><div></code>	Define uma seção no documento
<code><h1></code> a <code><h6></code>	Define cabeçalhos já formatados
<code><html></code>	Define a raiz de um documento HTML
<code><i></code>	Define um texto em itálico
<code></code>	Define uma imagem
<code><link></code>	Define uma relação entre o documento e um recurso externo (como por exemplo um arquivo de CSS)
<code><p></code>	Define um parágrafo
<code><script></code>	Define um script (como exemplo, um trecho de código em JavaScript)
<code><source></code>	Define múltiplos recursos de mídia para elementos de mídia como <code><video></code> e <code><audio></code>
<code><style></code>	Define informações de estilo do documento (exemplo, um trecho de código de CSS)
<code><title></code>	Define um título para o documento
<code><video></code>	Define um vídeo ou um filme

Fonte: Adaptado de W3SCHOOLS (1999-2013)

3.3.1 Tags para Texto

Para acrescentar textos, utiliza-se a *tag* `<p>`, para criar um parágrafo, ou as *tags* pré-formatadas, de `<h1>` até `<h6>`, que representam diferentes formatações para cabeçalhos. Vale ressaltar que é importante a utilização dessas *tags* para permitir a alteração do estilo do texto, ou seja, para permitir a troca de fonte, tamanho, cor, posicionamento, espaçamento e outros atributos. A utilização de estilos será tratada mais adiante nesse capítulo.

3.3.2 Tag para Imagem

Para o acréscimo de imagens, utiliza-se a *tag* ``, dentro de uma outra *tag* para servir de contêiner, sendo a mais comumente usada, a *tag* `<p>` [HARRIS 2013]. A *tag* `` é vazia, ou seja, ela não possui uma *tag* de fechamento e não possui conteúdo, apenas atributos, das quais dois são requeridos para o funcionamento da *tag*, os atributos `src` e `alt`. O atributo `src`, que é uma abreviação de *source* (fonte em inglês), indica a URL onde se encontra a imagem que se deseja mostrar. Vale ressaltar que poucos formatos são universais, a maioria das imagens na web estão no formato `.png`, `.jpg` ou `.gif` [HARRIS 2013]. No caso da imagem não for possível de ser carregada, o atributo `alt` fornece um texto alternativo, definido pelo autor, que será exibido no lugar da imagem. O Quadro 3.2 mostra alguns atributos globais que podem ser usados por qualquer elemento [W3SCHOOLS 1999-2013].

Quadro 3.2 - Alguns atributos globais

Atributo	Descrição
accesskey	Especifica uma tecla de atalho para ativar ou dar foco ao elemento
class	Especifica uma ou mais classes para um elemento
contenteditable	Especifica se um elemento é editável ou não
contextmenu	Especifica um menu de contexto para um elemento, o qual aparecerá quando o usuário clicar com o clique direito do mouse no elemento
draggable	Especifica se um elemento é arrastável ou não
dropzone	Especifica se o dado de um elemento é copiado, movido, ou linkado, quando solto após ser arrastado
id	Especifica uma identificação única para o elemento
lang	Especifica o idioma do conteúdo do elemento
spellcheck	Especifica se um elemento deve ou não ter sua ortografia ou gramática checada
style	Especifica um estilo CSS para o elemento
title	Especifica uma informação extra sobre um elemento
translate	Especifica se o valor de um elemento deve ser traduzido ou não, quando a página for localizada (traduzida)

Fonte: Adaptado de W3SCHOOLS (1999-2013)

3.3.3 Tag para Hiperlinks

Outra *tag* que pode vir a ser usada para a criação de um jogo em HTML, é a *tag* de link `<a>`. Essa *tag* é utilizada para realizar a mudança para uma outra página em HTML. Como dito anteriormente, ela utiliza do atributo `href`, para indicar o endereço da página HTML alvo. A *tag* `<a>` pode ser usada também como um contêiner para a *tag* `` criando assim uma imagem que também é um link para uma outra página.

3.3.4 Atributos Globais

Normalmente as *tags* possuem atributos essenciais para seu funcionamento, como a *tag* `<a>` que não faria sentido sem o atributo `href`, e a *tag* `` não mostraria imagem alguma sem o atributo `src`. Mas todas as *tags* podem se beneficiar de um grupo de atributos que podem ser acrescentados em qualquer *tag*, os chamados atributos

globais. Dos atributos existentes, tem-se os atributos *class* e *id*, que são importantes tanto para a utilização de estilos, quanto para a programação com JavaScript.

O atributo *class* é utilizado para definir uma série de comportamentos ou estilos específicos para um grupo de elementos. Supondo-se que na página aparecerão alguns textos em vermelho, pode-se criar uma *class*=”texto_vermelho” e aplicar em todas as *tags* que se deseja aplicar o efeito. Já o atributo *id*, é utilizado quando se deseja aplicar alguns comportamentos a apenas um único elemento, por exemplo, aplicar um efeito no jogador apenas, pode-se criar então o *id*=”jogador”, porém nesse caso, apenas um elemento poderá ter esse *id*, pois ele é único.

3.4 Utilizando CSS

Utilizando somente HTML, não é possível (em alguns casos até é possível, mas não é trivial) formatar a cor de fundo, posição, fonte dos textos, posição entre outros parâmetros de estética. Para isso utiliza-se o CSS (*Cascading Style Sheets*, em português Folha de Estilos em Cascata) [W3SCHOOLS 1999-2013].

O CSS é composto por propriedades, que possuem diversos valores para definir como o navegador deve posicionar ou estilizar os elementos do HTML. Os valores das propriedades são atribuídas da forma propriedade:valor; podendo ser um ou mais valores separados por um espaço em branco. Para valores com nome com mais de uma palavra, deve-se usar aspas envolvendo o nome, como por exemplo, a fonte “Times New Roman”. Os valores das propriedades, na maioria das vezes, podem ser herdados das propriedades do elemento que está acima da hierarquia dos elementos que estão recebendo tais propriedades, quando é dado o valor *inherit*. Quando não definidas, os elementos de HTML já possuem uma série de propriedades padrão.

Com relação ao código, o CSS pode ser definido dentro de uma *tag*, nesse caso é dito que o CSS está em linha e vem dentro do atributo *style*. Como resultado, o estilo é aplicado apenas no elemento em que está atribuído.

Todo o código em CSS é colocado entre aspas, como valor do atributo *style*, e cada propriedade é sempre finalizada com um caractere de ponto e vírgula “;”. Vale ressaltar que para o CSS em linha, é recomendado a utilização de aspas simples para nomes compostos de mais de uma palavra, pois o uso das aspas duplas pode confundir o *browser*.

Uma desvantagem do uso do código de CSS em linha, devido ao fato de estar embutido no código HTML, dificulta o processo de manutenção do código. O uso do CSS em linha não é recomendado também quando se deseja aplicar a diversos elementos o mesmo estilo, para isso é recomendado o CSS interno, quando o código CSS está dentro do próprio HTML, em uma *tag* `<style>` na *tag* `<head>`, ou o externo, quando o código CSS está em outro arquivo.

Todo código dentro de uma *tag* `<style>` já é interpretado como código CSS ao invés de código HTML. Portanto, para implementar um trecho de CSS interno, só é necessário colocá-lo como conteúdo da *tag*. O que é necessário, porém, é acrescentar na *tag* `<style>` um atributo *type*, com valor “text/css”, para indicar que o código é de CSS. No CSS interno, todos os estilos colocados no conteúdo da *tag* `<style>` será aplicado em todos os elementos da página HTML, conforme especificado.

A última forma de organizar os códigos de CSS é a externa, em que todos os códigos ficam armazenados em um arquivo separado. Essa maneira é muito útil quando se trabalha com várias páginas de HTML, todas com o estilo semelhante e também permite alterar o visual da página sem precisar editar o arquivo de HTML. Para utilizar a forma externa, é necessário criar o arquivo, que contém todo o código em CSS, com extensão .css, e importá-lo na página de HTML utilizando a *tag* `<link>` dentro da `<head>`. Tanto para o CSS interno, quanto para o externo, para cada elemento, as propriedades são colocadas entre chaves, “{ }”, da forma elemento { propriedade : valor; }, onde elemento pode ser o nome de um elemento, uma classe ou um *id*. O Quadro 3.3 lista algumas propriedades de CSS e a suas respectivas descrições.

Quadro 3.3 - Algumas propriedades de CSS

Propriedade	Descrição	Valores
background-color	Define uma cor para o plano de fundo de um elemento	Um valor de cor
background-image	Define uma imagem para o plano de fundo de um elemento	A url de uma imagem, como "url('imagem.gif');"
background-position	Define a posição inicial de uma imagem de plano de fundo	O valor da posição, em pixel ou right/left e top/bottom (nessa ordem, horizontal e vertical), como "left top;" ou "0 30px;"
border	Define todas as propriedades de borda em uma declaração apenas	border-width, border-style e border-color, exemplo "5px solid red;"
border-bottom	Define todas as propriedades de borda para a borda inferior	
border-left	Define todas as propriedades de borda para a borda esquerda	
border-right	Define todas as propriedades de borda para a borda direita	
border-top	Define todas as propriedades de borda para a borda superior	
border-bottom-color, border-left-color, border-right-color, border-top-color	Define a cor da borda inferior, esquerda, direita ou superior respectivamente	Um valor em hexadecimal para rgb, como "#FF0000"; um valor em rgb, como "rgb(255,0,0)"; o nome de uma cor, em inglês, como "red;" A borda também pode receber o valor "transparent;" para uma borda transparente
border-bottom-style, border-left-style, border-right-style, border-top-style	Define o estilo da borda inferior, esquerda, direita ou superior respectivamente	none (padrão) - sem borda, dotted - pontilhado, dashed - tracejado, solid - sólida, double - duplo sólida, groove, ridge, inset e outset - bordas com efeito 3D
border-bottom-width, border-left-width, border-right-width, border-top-width	Define a espessura da borda inferior, esquerda, direita ou superior respectivamente	Valor em pixels, como "5px;" ou um dos três valores pré-definidos, thin - fina, medium - média e thick - espessa
height	Define a altura de um elemento	auto - o navegador calcula a margem, um valor em pixel, centímetro, etc., um valor em % do elemento que o contém ou inherit - para receber o valor do elemento que o contém
max-height	Define a altura máxima de um elemento	
min-height	Define a altura mínima de um elemento	
width	Define a largura de um elemento	
max-width	Define a largura máxima de um elemento	
min-width	Define a largura mínima de um elemento	
font	Define todas as propriedades de fonte em uma declaração	
font-family	Especifica uma família de fontes para um texto	Um ou mais nome de fontes separados por vírgulas, com aspas em caso de nome com mais de uma palavra
font-size	Especifica o tamanho da fonte para um texto	Um valor em pixels ou em em (1 em é o valor padrão do tamanho da fonte nos navegadores, equivalente a 16 px, px não funciona em versões antigas, antes da 9, do Internet Explorer)
font-style	Especifica um estilo de fonte para um texto	normal, italic - itálico, oblique - semelhante ao itálico
margin	Define todas as propriedades de margem em uma declaração	Valores como especificado abaixo na ordem: top right bottom left, ou top e bottom right e left ou um valor para as 4 margens
margin-bottom	Define a margem inferior de um elemento	auto - o navegador calcula a margem, um valor em pixel, centímetro, etc., um valor em % do elemento que o contém ou inherit - para receber o valor do elemento que o contém
margin-left	Define a margem esquerda de um elemento	
margin-right	Define a margem direita de um elemento	
margin-top	Define a margem superior de um elemento	
color	Define a cor de um texto	Um valor de cor
text-align	Define o alinhamento horizontal de um texto	center - centralizado, left ou right, justify - justificado
text-decoration	Especifica a decoração do texto	none - nenhuma (também usada para remover underlines dos links), underline, line-through, underline
text-indent	Especifica a indentação do texto	valor em pixels, como "50px;"
text-transform	Controla e transforma as letras das palavras do texto em maiúscula ou minúscula	uppercase, lowercase, capitalize - transforma em maiúscula a primeira letra de cada palavra do texto

Fonte: Adaptado de W3SCHOOLS (1999-2013)

3.4.1 Definindo Cores

Existe algumas maneiras de especificar cores no CSS. Serão apresentadas nesse trabalho, quatro maneiras, mais usadas: a representação em hexadecimal, a representação em *rgb* e a *rgba*, e as cores com nomes pré-definidos no *browser*.

As formas *RGB* e Hexadecimal são semelhantes, ambas utilizam valores de vermelho, verde e azul (*Red Green Blue*) para especificar a cor. Em Hexadecimal os valores *RGB* vão de 00 até FF, sendo dois algarismos, no início, para vermelho, dois, no meio, para o verde e dois, no fim, para o azul, totalizando seis algarismos. É representado para o navegador na forma #valores, como por exemplo, #FF0000 para a cor vermelha. Já na forma *RGB* o valor varia de 0 a 255 separados por vírgula, da forma *rgb*(vermelho, verde, azul), como por exemplo a cor vermelha com *rgb*(255,0,0).

O *RGBA* (*Red Green Blue Alpha*) possui a mesma representação do *RGB* com o acréscimo da opacidade alfa, variando de 0.0 para completamente transparente e 1.0 para completamente opaco. A representação é *rgba*(vermelho, verde, azul, alfa), como o *rgba*(255,0,0,0.5) para um vermelho meio transparente.

A última forma utiliza nome de cores em inglês, como *red*, *blue*, entre outros.

3.4.2 Posicionando os elementos da página

Até agora, os elementos foram acrescentados na página de forma que o programador tenha pouco controle sobre o posicionamento dos elementos. É possível dizer para o navegador onde exatamente deve estar cada elemento da página.

Para o posicionamento dos elementos são utilizadas, também, propriedades de CSS. Os elementos podem ser posicionados utilizando as propriedades *top*, *bottom*, *left* e *right*. Porém, essas propriedades não funcionarão a menos que seja especificada a propriedade *position*, funcionando diferentemente para cada método de posicionamento [W3SCHOOLS 1999-2013]. Como mostrado no Quadro 3.4, a propriedade *position* possui quatro valores: *static* – estático, *fixed* - fixo, *relative* - relativo e *absolute* - absoluto.

O posicionamento estático é o padrão de todos os elementos em HTML que não possuem a propriedade *position*. Nesse método de posicionamento os elementos seguem o fluxo normal de posicionamento dos elementos e eles não são afetados pelas propriedades *top*, *bottom*, *left* e *right* [W3SCHOOLS 1999-2013].

No posicionamento fixo, os elementos são posicionados relativamente à janela do navegador, através das propriedades *top*, *bottom*, *left* e *right*. Os elementos utilizando essa propriedade ficarão fixos na tela, mesmo ao mover a barra de rolagem e eles são removidos do fluxo normal de posicionamento dos elementos, como se os elementos não existissem na página. Esses elementos podem se sobrepor a outros elementos [W3SCHOOLS 1999-2013].

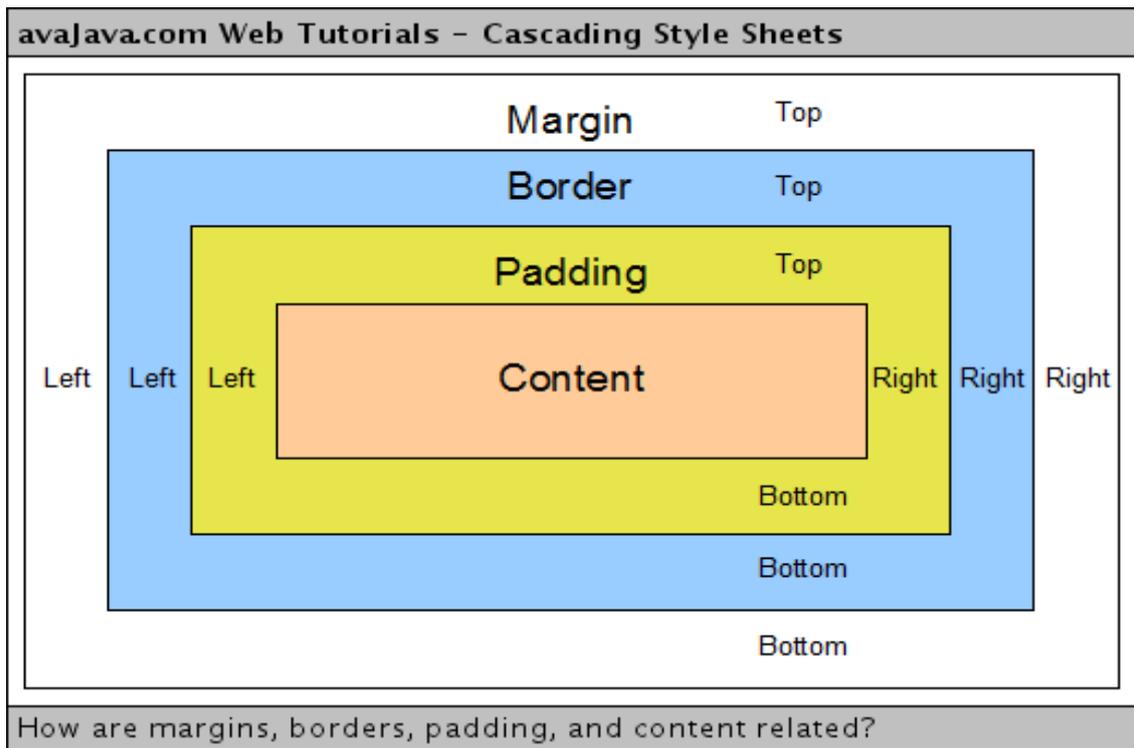
Para o posicionamento relativo, os elementos são posicionados relativamente à sua posição normal. Os elementos contidos em um elemento com posição relativa podem ser movidos e podem se sobrepor a outros elementos, mas o espaço reservado para o elemento continua sendo preservado pelo fluxo normal. Elementos com posição relativa são frequentemente usados como contêineres para elementos com posição absoluta [W3SCHOOLS 1999-2013].

Já no posicionamento absoluto, os elementos são posicionados relativamente ao primeiro elemento com o posicionamento não estático, que os contém. Caso tal elemento não exista, será considerado o elemento `<html>` [W3SCHOOLS 1999-2013]. Por exemplo, uma *tag* `<p>` dentro da `<body>` vai ter a posição relativa ao corpo da página, caso a *tag* `<body>` tenha um posicionamento diferente do estático, e uma *tag* `<a>` dentro dessa mesma *tag* `<p>` terá a sua posição relativa à *tag* `<p>`.

Os elementos que estão fora do fluxo normal de posicionamento, como foi dito anteriormente, podem sobrepor uns aos outros. Nesse caso a propriedade *z-index* controla quem fica na frente de quem, podendo ter valores positivos, nulo ou negativos. O elemento com maior valor do *z-index* ficará na frente dos demais com menores valores que o seu. Caso os elementos não tenham o valor *z-index* especificados, o elemento que foi posicionado por último ficará na frente dos anteriores [W3SCHOOLS 1999-2013].

Além do posicionamento, existem algumas propriedades para controlar a distância entre dois elementos na página. As propriedades *padding* (afastamento) e *margin* (margem) são usadas para esse caso. A diferença entre eles somente está no fato de que o valor do *padding* afasta também a borda do elemento enquanto o *margin* afasta somente os elementos externos, como ilustrado na Figura 3.1.

Figura 3.1 - Comportamento das tags *margin*, *border* e *padding*



Fonte: <http://www.avajava.com/tutorials/lessons/how-are-margins-borders-padding-and-content-related.html>, Acessado em Setembro de 2013

O Quadro 3.4 mostra as propriedades de CSS voltadas a posição e afastamento.

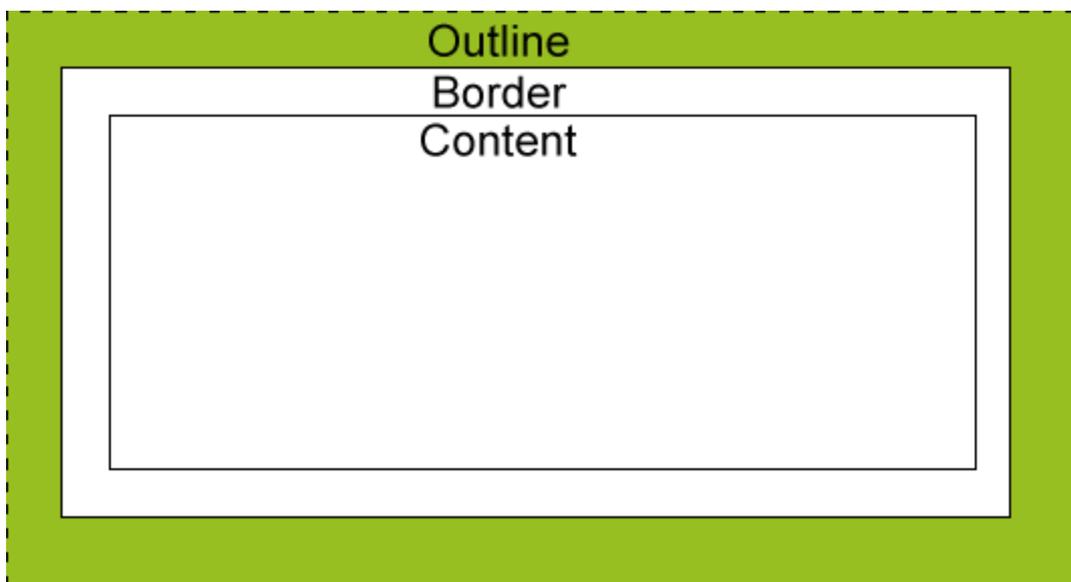
Quadro 3.4 - Propriedades de CSS para posição e afastamento

Propriedade	Descrição	Valores
outline	Define todas as propriedades de contorno em uma declaração	out-line color, outline-style, outline-width exemplo "5px solid red;"
outline-color	Define a cor do contorno	Um valor em hexadecimal para rgb, como "#FF0000";, um valor em rgb, como "rgb(255,0,0)";, o nome de uma cor, em inglês, como "red;" O contorno também pode receber os valores "inherit;" - para manter a mesma cor do elemento que o contém e "invert;" para inverter as cores do elemento que o contém
outline-style	Define o estilo do contorno	none (padrão) - sem borda, dotted - pontilhado, dashed - tracejado, solid - sólida, double - duplo sólida, groove, ridge, inset e outset - bordas com efeito 3D
outline-width	Define a espessura de um contorno	Valor em pixels, como "5px;" ou um dos três valores pré-definidos, thin - fina, medium - média e thick - espessa
padding	Define todas as propriedades de afastamento de um elemento	Valores como especificado abaixo na ordem: top right bottom left, ou top e bottom right e left ou um valor para as 4 margens
padding-bottom	Define o afastamento inferior de um elemento	auto - o navegador calcula a margem, um valor em pixel, centímetro, etc., um valor em % do elemento que o contém ou inherit - para receber o valor do elemento que o contém
padding-left	Define o afastamento esquerdo de um elemento	
padding-right	Define o afastamento direito de um elemento	
padding-top	Define o afastamento superior de um elemento	
bottom	Especifica a posição inferior de um elemento	
left	Especifica a posição esquerda de um elemento	
right	Especifica a posição da direita de um elemento	
top	Especifica a posição do topo de um elemento	
cursor	Especifica o tipo do cursor a ser mostrado, quando o elemento é apontado	URL - o caminho para uma arquivo de imagem exemplo "url(imagem.gif);", auto - decidido pelo navegador (padrão), crosshair, default, e-resize, help, move, n-resize, ne-resize, pointer, progress, s-resize, se-resize, sw-resize, text, w-resize, wait, separados entre vírgulas para informar mais de um possível valor, exemplo: "cursor:url(imagem.gif), auto;"
display	Especifica como um elemento deve ser mostrado na tela	none - elemento não é mostrado e não ocupa espaço, block - o elemento pega todo o comprimento disponível e quebra a linha antes e após, inline - o elemento ocupa somente o comprimento necessário
float	Especifica se um grupo de elementos deve ou não flutuar	left, right, none
clear	Especifica qual lado de um elemento que não permite elementos flutuantes	left, right, both, none
overflow	Especifica o que acontece quando um elemento transborda do grupo	visible - o elemento mantém visível, hidden - o elemento é cortado e só é mostrado a parte dentro do grupo, scroll, o elemento é cortado e a barra de rolagem é mostrada, auto - o elemento é cortado e uma barra de rolagem é mostrada
position	Especifica o tipo do método de posicionamento usado em um elemento	static - estático, relative - relativo, absolute - absolute, fixed - fixo
visibility	Especifica se um elemento é visível ou não	hidden - elemento é invisível mas ainda ocupa espaço, visible - visível (padrão)
z-index	Define a ordem do posicionamento de um elemento no eixo Z	um valor positivo ou negativo

Fonte: Adaptado de W3SCHOOLS (1999-2013)

Os Quadros 3.3 e 3.4, nota-se a existência de propriedades que permitem aplicar uma borda ou um contorno nos elementos. A diferença entre elas é que *border* (borda) envolve o conteúdo do elemento e é afastada pelo *padding*, já o *outline* (contorno) envolve o elemento todo, incluindo a margem e a borda, como ilustrada na Figura 3.2.

Figura 3.2 - Diferença entre *Outline* e *Border*



Fonte: http://www.w3schools.com/css/css_outline.asp, Acessado em Setembro de 2013

O Quadro 3.4 apresentou as propriedades *float* e *clear*. A propriedade *float* permite que os elementos flutuem horizontalmente, podendo ser empurrados para esquerda ou direita, permitindo outros elementos os envolverem. Isso geralmente faz com que o elemento vá o máximo a esquerda ou direita o possível do elemento que o contém. Os elementos que vêm após o elemento flutuante, no código html, vai flutuar ao seu redor, mas os elementos que vêm antes não são afetados. No caso de texto, se um elemento flutua para a direita, o texto a seguir flutuará ao redor dele pela esquerda, como no exemplo do Quadro 3.5, ilustrado pela Figura 3.3. A propriedade *clear* serve para que o elemento, vindo após um elemento com a propriedade *float*, não flutua. É necessário especificar se o *clear* não permitirá a flutuação para a esquerda, para a direita, ou ambas.

3.4.3 Organizando os elementos

A maioria das páginas na Internet separam os conteúdos em diversas colunas, o mesmo pode ser aplicado para um jogo. Como exemplo, um painel contendo o inventário do personagem do jogador ao apertar a tecla I, pode ser uma *div*, que fica visível ao pressionar a tecla I e volta a desaparecer ou apertá-la novamente. Para organizar os elementos presentes no corpo da página de HTML, é utilizada a tag `<div>` para agrupar os elementos em blocos [W3SCHOOLS 1999-2013].

A tag `<div>` não possui outro significado especial além de servir de contêiner para agrupar outros elementos de HTML. Usadas em combinação com CSS, é possível aplicar propriedades de estilo a um grande número de elementos. É comum utilizar `<div>` para definir o layout da página [W3SCHOOLS 1999-2013].

Outra forma de organizar os elementos é em linha, com a utilização da tag ``. Ela é utilizada, por exemplo, para aplicar propriedades de estilo a partes de um texto. Assim como a `<div>`, `` não possui outro significado a não ser um contêiner para outros elementos [W3SCHOOLS 1999-2013]. A diferença entre o `` e o `<div>` é que o `<div>` normalmente começa e termina em uma nova linha, já o ``, normalmente é usado na mesma linha que outros elementos.

3.5 Tag para Áudio

Nas versões anteriores de HTML, a utilização de áudio, na página, não era uma tarefa fácil, sendo uma das razões que deu força à utilização do Flash para jogos via web. Entretanto, atualmente com o HTML 5, tem-se o suporte nativo à utilização de áudio com a tag `<audio>`. Para a utilização de áudio, primeiramente, é necessário obter os arquivos de áudio em formato mp3, *wave* ou ogg [W3SCHOOLS 1999-2013], e depois colocar no corpo do html, tag `<body>`, a tag `<audio>`, como no exemplo abaixo:

Quadro 3.6 - Implementando a tag <audio>

```
1 <body>
2   <audio controls = "controls">
3     <source src = "audio.mp3" type = "audio/mp3" />
4     <source src = "audio.ogg" type = "audio/ogg" />
5     <source src = "audio.wav" type = "audio/wav" />
6     <a href = "audio.mp3"> Tocar audio </a>
7   </audio>
8 </body>
9
```

Fonte: O Autor

O atributo *controls* é um atributo opcional, que adiciona controles de áudio, com botões de tocar, pausar e controle de volume. A não utilização dele faz com que o áudio seja tocado logo quando a página for carregada. Nota-se que foram utilizadas três tags <source>, embora todos os *browsers*, nas versões atuais, dão suporte à tag <audio>, nem todos suportam a mesma extensão de arquivo de áudio. Alguns funcionam com mp3, outros com ogg ou wav. Para se obter o melhor resultado, é recomendado fornecer pelo menos o áudio em mp3 e ogg [HARRIS 2013]. Dessa forma os *browsers* escolherão o primeiro formato que for compatível. No exemplo também foi utilizada uma tag <a>, que aparecerá no caso do usuário estiver usando uma versão de um *browser* que não tenha compatibilidade com a tag <audio>, sendo assim o link será mostrado no lugar.

3.6 Programando com JavaScript

Até agora foram vistos elementos de HTML ou HTML 5 que podem ser utilizados para o desenvolvimento de um jogo simples, com imagens estáticas, textos, links, áudio, que tocam apenas no início da página ou quando o usuário pressionar o botão de tocar. E também foi visto como o browser mostra, agrupa ou executa os elementos de HTML através de CSS. Porém, para que o jogo modifique os elementos presentes na tela, com animação, movimentação de cenário, áudio, respondendo as ações do jogador, será necessário uma linguagem de programação além do que foi visto até agora, o JavaScript.

3.6.1 Definição

Segundo VALENTINE & REID (2013, p. 2-3), JavaScript é uma linguagem de programação que foi lançado na primeira vez em 1995. O nome JavaScript não representa qualquer relação com a linguagem de programação Java. De um nível elevado, JavaScript tem várias características notáveis:

- É uma linguagem de script: programas JavaScript são "scripts" lidos e executados por um interpretador (ou motor). Ele é distinguido de linguagens compiladas, onde os programas são lidos por um compilador e traduzido para dentro de um arquivo executável. Os motores frequentemente JavaScript próprios são escritos em uma linguagem compilada. Programas escritos em linguagens de script são altamente portáteis, que podem rodar em qualquer ambiente onde um interpretador foi construído para aquela linguagem;
- É um C-like: a sintaxe básica de JavaScript e a estrutura são emprestadas pesadamente de C;
- É uma linguagem orientada a objetos: Porém JavaScript difere da maioria das linguagens orientadas a objeto no seu modelo originário e é baseado em protótipo ao em vez de classe;
- Possui funções de primeira classe: funções JavaScript também são objetos de pleno direito e têm seus próprios métodos e propriedades, e podem ser passados para outras funções como parâmetros ou retornados de outras funções e atribuídas a variáveis;
- Ele é dinâmico: O termo " linguagem de programação dinâmica " é amplo e abrange uma série de características. Dentre as performances mais dinâmicas do JavaScript estão, a sua implementação de digitação variável, seu método eval () e outros aspectos funcionais;
- Variáveis de JavaScript não são verificadas no momento da interpretação (que fazem do JavaScript uma linguagem de tipagem dinâmica), e como as operações ocorrem entre operandos de tipos mistos vai depender de regras específicas dentro de JavaScript (fazendo JavaScript uma linguagem fracamente tipada); e

- É uma implementação de um padrão: Como descrito na seção anterior, o JavaScript é na verdade, uma aplicação da norma ECMA -262, assim como a linguagem de programação C é regulada pelos padrões ISO.

Estas principais características se combinam para fazer de JavaScript algo único. Eles também ajudam a tornar o básico de JavaScript bastante fácil de saber quando se tem uma familiaridade com linguagens C *-like*, pois não haverá muito problema com sintaxe ou a estrutura de JavaScript.

JavaScript também é fortemente influenciada pela Scheme⁵, outra linguagem de programação funcional, que é um dialeto do Lisp⁶. JavaScript obtém muitos de seus princípios de design do Scheme, incluindo o seu escopo.

JavaScript é atualmente uma implementação de uma norma padrão, embora no início não era dessa forma. Em setembro de 1995, a Netscape⁷ lançou a versão 2.0 de seu *browser* Navigator, que teve um novo recurso: uma linguagem de script orientada a objetos que poderiam acessar e manipular os elementos da página. Sendo criado por Brendan Eich, da Netscape, e originalmente denominado de "Mocha"⁸, a nova linguagem de script foi lançado inicialmente como "LiveScript." Pouco tempo depois foi rebatizada de "JavaScript", seguindo a linguagem de programação Java da Sun.

Em 1996, a Netscape submeteu JavaScript à ECMA (Associação europeia dos fabricantes de computadores), para ser considerado um padrão, conforme descrito no site: <http://www.ecma-international.org/memento/history.htm>.

Em Junho de 1997 foi adotado o padrão ECMA-262 definindo adequadamente a linguagem script ECMAScript, sendo o JavaScript considerado um "dialeto" do ECMAScript. Outro dialeto notável de ECMAScript é a versão 3 ou posterior do ActionScript⁹. Tecnicamente, o Internet Explorer¹⁰ não executa JavaScript (devido a questões de direitos autorais), mas em vez disso implementa próprio dialeto da Microsoft ECMAScript chamado "JScript". A última versão, ECMAScript 5.1, foi publicada em junho de 2011.

⁵ <http://groups.csail.mit.edu/mac/projects/scheme/>, Acessado em Setembro de 2013

⁶ Lisp-BR. Disponível em <http://lisp-br.org/>, Acessado em Setembro de 2013

⁷ <http://netscape.aol.com/>, <http://isp.netscape.com/>, Acessados em Setembro de 2013

⁸ <http://docs.webplatform.org/wiki/concepts/programming/javascript/history>, Acessado em Setembro de 2013

⁹ <http://www.adobe.com/devnet/actionsript.html>, Acessado em Setembro de 2013

¹⁰ <http://windows.microsoft.com/pt-br/internet-explorer/download-ie>, Acessado em Setembro de 2013

JavaScript pode ser implementado de várias maneiras diferentes. O sistema de documentos da Adobe Acrobat¹¹, é um exemplo de implementação, sendo uma versão do JavaScript que permite aos usuários utilizar scripts simples em documentos Acrobat. Os motores do JavaScript também foram implementadas com recursos independentes no Windows¹², UNIX¹³ e Linux¹⁴ por algum tempo. Pouco depois, o JavaScript foi introduzido pela primeira vez em 1995, Netscape incluiu a implementação do lado do servidor em sua Enterprise Server. A mais notável implementação de JavaScript do lado do servidor é no sistema de software Node.js¹⁵. As implementações mais comuns de JavaScript estão em navegadores web. O motor browser JavaScript de web normalmente implementa a maioria das funcionalidades especificadas no padrão ECMA-262. Além disso, browsers com frequência amplia JavaScript com outros recursos não especificados pelo padrão ECMA. A mais notável dessas extensões é o DOM¹⁶ (documento objeto modelo), que é um padrão separado mantido pelo consórcio W3C.

3.6.2 Utilizando JavaScript

JavaScript é inserida dentro das páginas de HTML, tanto contida na `<head>`, quanto na `<body>`, podendo ser executada por todos os navegadores modernos [W3SCHOOLS 1999-2013]. Para acrescentar os códigos em JavaScript é utilizada a *tag* `<script>`, tanto para o código acrescentado no próprio arquivo HTML, quanto para a utilização de um arquivo externo. Supondo a existência de um arquivo `script.js`, dentro de um diretório `js`, o qual se encontra no mesmo diretório do arquivo `html`, observe os exemplos abaixo de como inserir o código de JavaScript no arquivo `html`.

¹¹ <https://www.adobe.com/br/products/acrobat.html>, Acessado em Setembro de 2013

¹² <http://windows.microsoft.com/pt-BR/windows/home/>, Acessado em Setembro de 2013

¹³ http://www.unix.org/what_is_unix.html, Acessado em Setembro de 2013

¹⁴ <http://www.linuxnarede.com.br/artigos/fullnews.php?id=1>, Acessado em Setembro de 2013

¹⁵ <http://nodejs.org/>, Acessado em Setembro de 2013

¹⁶ <http://www.w3schools.com/html/dom/default.asp>, Acessado em Setembro de 2013

Quadro 3.7 - Inserindo JavaScript no HTML

```
1 <head>
2   <script type = "text/javascript">
3     function funcao() {
4       document.write("Olá Mundo!")
5     }
6   </script>
7
8   <script src = "js/script.js" type = "text/javascript"> </script>
9 </head>
10
```

Fonte: O Autor

Na primeira *tag* `<script>`, o código foi inserido como conteúdo da *tag*. Nele foi criada uma função que mostrará na tela a mensagem “Olá Mundo!”, no momento em que a página for carregada. Já a segunda *tag* importará todo o script contido no arquivo `script.js`. Assim como o CSS externo ao html, separando o código em JavaScript do HTML, permite o reuso do código em várias páginas.

3.6.3 Criando formulários

Uma forma fundamental de interação com o jogador é através de formulários. Neles pode-se, por exemplo, pedir ao usuário que insira o nome do seu personagem, para um jogo de RPG, ou para preencher as letras em um jogo de forca, entre outras utilizações. Os elementos de entrada podem ser desde caixas de textos a elementos mais avançados, como *radio buttons* ou *combo boxes*. A criação dos elementos é dada utilizando HTML, mas para realmente ler as entradas do usuário é necessário JavaScript [HARRIS 2013].

O Quadro 3.8 mostra os elementos usados para a criação de formulários. O elemento `<input>` possui diversos atributos, listados no Quadro 3.9, para controlar como o elemento vai ser exibido, o que ele deve receber e outros parâmetros.

Quadro 3.8 - Elementos para formulários

Tag	Descrição
<form>	Define um elemento de formulário para as entradas do usuário
<input>	Define um elemento que controla uma entrada do usuário
<textarea>	Define um elemento para controle de entrada com múltiplas linhas
<label>	Define um rótulo para um elemento <input>
<fieldset>	Agrupa elementos relacionados em um <form>
<legend>	Define uma legenda para um elemento <fieldset>
<select>	Define uma lista expandível
<optgroup>	Define um grupo de opções relacionadas em uma lista expandível
<option>	Define uma opção em uma lista expandível
<button>	Define um botão clicável
<datalist>	Especifica uma lista de opções pré-definidas para controles de entrada
<keygen>	Define um campo de gerador de chaves em pares
<output>	Define o resultado de um cálculo

Fonte: Adaptado de W3SCHOOLS (1999-2013)

Quadro 3.9 - Alguns atributos para a tag <input>

Atributo	Descrição	Valores
accept	Especifica os tipos de arquivos que o servidor aceita (somente para o type = "file")	audio/*, video/*, image/*, MIME_type
alt	Especifica um texto alternativo para imagens (somente para o type = "image")	Um texto
autocomplete	Especifica se o <input> deve completar a entrada do usuário	on, off
autofocus	Especifica se o <input> recebe foco automaticamente quando a página carrega	autofocus
checked	Especifica se o elemento deve vir pré-selecionado ao carregar a página (para type="checkbox" ou type="radio")	checked
form	Especifica on ou mais <form> que o elemento <input> pertence	o id do <form>
height	Especifica a altura do <input> (para type="image")	altura em pixels
list	Referencia uma <datalist> que contém as opções pré-definidas para o <input>	o id da <datalist>
max	Especifica o valor máximo do elemento <input>	um número ou data
maxlength	Especifica o número máximo de caracteres permitidos pelo elemento <input>	valor numérico
min	Especifica o valor mínimo do elemento <input>	um número ou data
multiple	Especifica que o usuário pode entrar mais de um valor em um elemento <input>	multiple
name	Especifica o nome de um <input>	Texto
placeholder	Especifica uma breve dica que descreve o valor esperado de um <input>	Texto
readonly	Especifica se o campo de entrada é de somente leitura	readonly
required	Especifica se o campo de entrada deve ser preenchido obrigatoriamente antes do formulário ser submetido	required
size	Especifica o comprimento, em caracteres, de um <input>	número
src	Especifica a URL da imagem que será usada como um botão de submeter (somente para type="image")	URL
type	Especifica o tipo de <input> que o elemento mostrará	button, checkbox, color, date, datetime, datetime-local, email, file, hidden, image, month, number, password, radio, range, reset, search, submit, tel, text, time, url, week
value	Especifica o valor de um elemento <input>	Texto
width	Especifica o comprimento de um elemento <input> (apenas para type="image")	Tamanho em pixels

Fonte: Adaptado de W3SCHOOLS (1999-2013)

3.6.4 Criando eventos

Com o que foi visto até agora, ao apertar um botão de um formulário, não surtirá efeito algum. Para que um *script* seja executado, as *tags* de HTML podem implementar uma série de atributos que são usados para engatilhar eventos, os atributos globais de evento. O Quadro 3.10 mostra alguns desses atributos e a descrição de quando os seus eventos serão executados.

Quadro 3.10 - Quadro de atributos globais de evento

Atributo	Descrição
onerror	Executado quando um erro ocorre
onhaschange	Executado quando o documento mudou
onload	Executado assim que a página carregar
onoffline	Executado quando o documento ficar offline
ononline	Executado quando o documento ficar online
onfocus	Executado quando o elemento for focado
onblur	Executado quando o elemento perder o foco
oninvalid	Executado quando um elemento é inválido
onselect	Executado quando algum texto for selecionado no elemento
onsubmit	Executado quando o formulário é submetido
onkeydown	Executado quando o usuário está pressionando uma tecla
onkeypress	Executado quando o usuário pressionou uma tecla
onkeyup	Executado quando o usuário soltou uma tecla
onclick	Executado quando o usuário clica no elemento
ondblclick	Executado quando o usuário clica duas vezes no elemento
ondrag	Executado quando o elemento for arrastado
ondragend	Executado depois que o elemento foi arrastado
ondragenter	Executado quando o elemento foi arrastado para uma determinada área alvo
ondragleave	Executado quando um elemento deixa uma determinada área alvo
ondragover	Executado quando um elemento é arrastado sobre uma determinada área alvo
ondragstart	Executado no início da operação de arrastar um objeto
ondrop	Executado quando o elemento arrastado está sendo solto
onmousedown	Executado quando um botão do mouse é pressionado no elemento
onmousemove	Executado quando o ponteiro do mouse passa pelo elemento
onmouseout	Executado quando o ponteiro do mouse deixa o elemento
onmouseover	Executado quando o ponteiro do mouse está sobre o elemento
onmouseup	Executado quando o botão do mouse é solto sobre o elemento
onmousewheel	Executado quando a rodana do mouse é rotacionada
onscroll	Executado quando a barra de rolagem de um elemento está sendo usada

Fonte: Adaptado de W3SCHOOLS (1999-2013)

Uma forma de interação com usuário muito usadas em jogos, é a utilização do teclado. Para capturar uma tecla pressionada pelo usuário, pode-se colocar um atributo de evento, *onkeypress* ou *onkeydown* na tag *<body>* e então chamar uma função que irá executar as alterações desejadas. A função receberá automaticamente o evento, através de um objeto *Event*, normalmente chamado de “e” [HARRIS 2013]. Usando então *e.keyCode*, ou *e.which*, obtém-se o código da tecla pressionada e usando-se a função *fromCharCode(código)*, obtém-se o caractere que foi pressionado.

Outra forma de interação bem utilizada em jogos no computador, é a utilização dos eventos do *mouse*. Isso se deve ao fato de ser um dos meios de interação mais intuitivo que o computador possui. HTML conta com vários eventos com relação ao mouse, como clique e duplo clique, arrastar um elemento para algum canto específico, ou simplesmente arrastar o ponteiro do *mouse* em cima de um elemento.

3.6.5 Usando números aleatórios

Números aleatórios são uma parte chave da programação de um jogo, que tende a imitar a complexidade e a imprevisibilidade do universo [HARRIS 2013]. Números aleatórios são importantes, também, para o valor de *replay*, permitindo que, cada vez que o jogo é jogado, alguns elementos estejam diferentes das vezes anteriores.

A maioria das linguagens de programação possui algum gerador de números aleatórios. No caso de JavaScript, existe uma função especial para a geração de um número semi-aleatório, presente na biblioteca *Math* [HARRIS 2013]. A função *Math.random()* gera um número de 0 a 1, incluindo o 0, mas não incluindo o 1. Para se obter valores de uma faixa desejada, como por exemplo, de 1 a 10, basta multiplicar o valor da função *random* pelo valor, nesse caso 10 e aplicar a função *Math.ceil()*, que arredonda os valores para cima, para de 1 a 10 e *Math.floor()*, que arredonda os valores para baixo, para 0 a 9.

3.7 Elemento *Canvas*

Segundo ROWELL (2011, p. 3), a API do elemento *Canvas*, do HTML 5, foi desenvolvido usando JavaScript. Ela permite que desenvolvedores web criem visualizações e animações direto no browser sem o uso da ferramenta Flash. Portanto, o *Canvas* está rapidamente se tornando padrão para gráficos on-line e para interatividade.

Muitos desenvolvedores ainda não conseguem exercer todos os recursos que esta poderosa tecnologia tem para oferecer.

Canvas foi originalmente criado pela Apple em 2004 para implementar *widgets* para o *Dashboard* e para gráficos de energia no navegador Safari. Depois foi adotado pelo Firefox, Opera e Google Chrome. Atualmente, *canvas* é uma parte da nova especificação HTML5 para a próxima geração de tecnologias web.

Segundo LAWSON & SHARP (2011, p. 115-140), o elemento *Canvas* fornece uma API para desenho 2D – linhas, preenchimentos, imagens, texto, formas, entre outros. A API tem sido utilizada amplamente nas aplicações, incluindo planos de fundo para sites, elementos de navegação, ferramentas gráficas, aplicações completas, jogos e emuladores.

O *Canvas* e o SVG são duas API de desenho muito boas. O SVG é uma API de modo retido, e a API *canvas2D* é uma API de modo imediato.

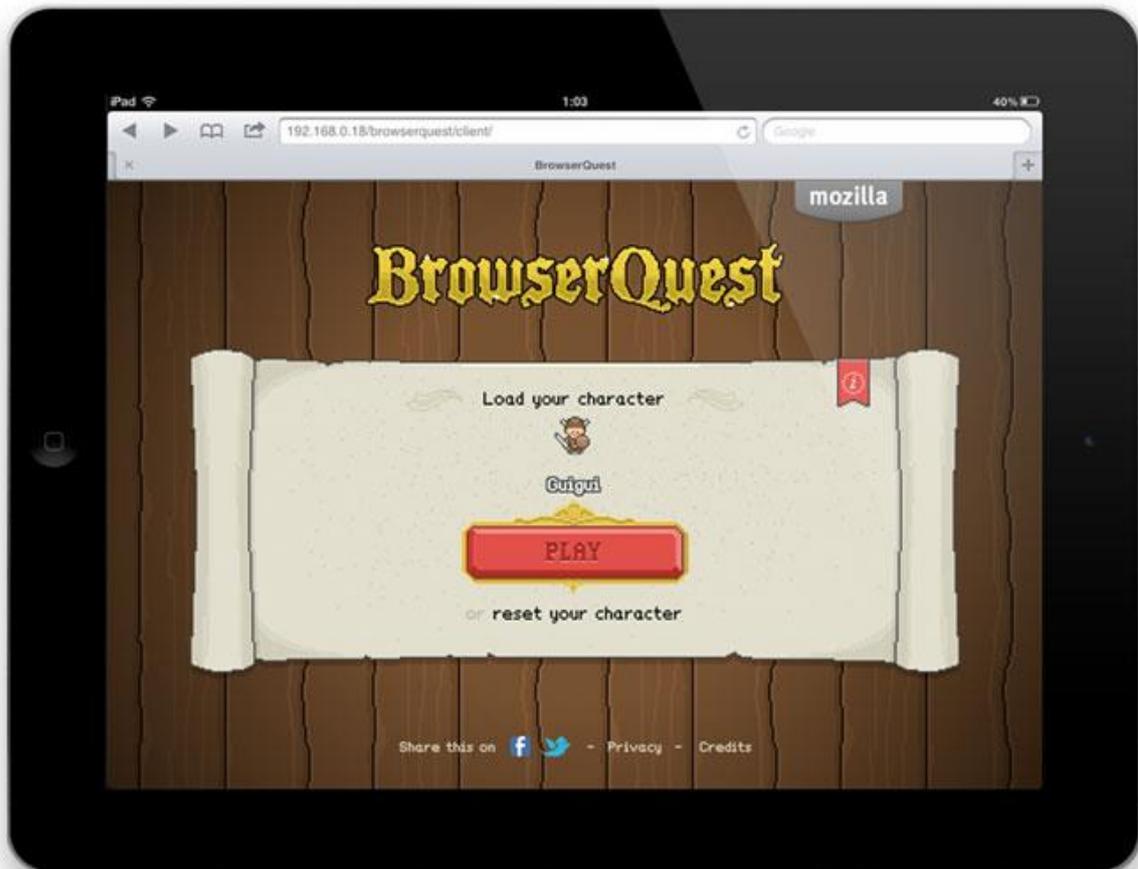
O SVG estrutura uma árvore que representa o estado atual de todos os objetos desenhados na tela, que é o caso de uma API de modo retido. Neste caso é possível conectar objetos específicos na árvore e escutar eventos de clique e toque, incluindo elementos que detectam colisão para jogos. *Canvas* é uma imagem baseada em bitmap, portanto não é redimensionável, com ajustes somente através de zoom.

Canvas é recomendado para diversas animações e aplicativos desenvolvidos no JavaScript. Comparado ao SVG, é uma API de nível inferior ou seja, é melhor quando não há interação de mouse, devido a não haver uma árvore no estado de *Canvas*. Para interação com teclado tem mostrado melhor utilidade, da mesma forma que jogos de 8-bit recentemente e é muito recomendado para trabalhar com pixels.

4 Browserquest

PAUL - ArsTechnica (2012) - descreve o BrowserQuest, ilustrado na figura 4.1, como um jogo construído com JavaScript e HTML5, uma demonstração convincente de como os padrões existentes podem ser usados na criação de jogos de browser.

Figura 4.1 - Tela Inicial do Browserquest em um tablet



Fonte: <http://www.littleworkshop.fr/browserquest.html>, Acessado em Setembro de 2013

BrowserQuest foi desenvolvido para os navegadores Google Chrome¹⁷, Mozilla Firefox¹⁸, Safari¹⁹ (Apple) e Opera²⁰, com base na nova tecnologia WebSocket²¹. Pode ser jogado por milhares de usuários ao mesmo tempo, espalhados em vários mundos do jogo. Além do computador, pode ser jogado dispositivos móveis, ilustrado na figura 4.2.

¹⁷ <http://www.google.com/intl/pt-BR/chrome/>, Acessado em Setembro de 2013

¹⁸ <http://www.mozilla.org/pt-BR/firefox/new/>, Acessado em Setembro de 2013

¹⁹ <https://www.apple.com/br/safari/>, Acessado em Setembro de 2013

²⁰ <http://www.opera.com/pt-br>, Acessado em Setembro de 2013

²¹ <http://www.websocket.org/aboutwebsocket.html>, Acessado em Setembro de 2013

Cada jogador ainda pode conversar com outros usuários e também ver quantas pessoas estão atualmente online no seu mundo.

Figura 4.2 - Tela do Browserquest em um iPad e em um iPhone



Fonte: <http://www.littleworkshop.fr/browserquest.html>, Acessado em Setembro de 2013

Segundo ALLERGIC (2013), citado por CHEN (2013), BrowserQuest é um jogo de aventura MMORPG (*Massively Multiplayer Online Role Playing Game*, em português, jogo online de multijogador em massa de interpretação) construído por Mozilla Foundation²² e Little Workshop²³. Ele foi originalmente projetado para ser um demo de como o Canvas de HTML 5 e o WebSockets podem ser usados na implementação de um jogo. Ele abriga vários servidores com balanceamento de carga e com múltiplo ambiente em cada um. Em conexão, o jogador pode interagir com outros jogadores no ambiente onde estiverem conectados. Outra notável contribuição do BrowserQuest é a compatibilidade com dispositivos móveis (iOS²⁴, Android²⁵).

²² <http://www.mozilla.org/en-US/foundation/>, Acessado em Setembro de 2013

²³ <http://www.littleworkshop.fr/>, Acessado em Setembro de 2013

²⁴ <http://www.apple.com/br/ios/>, Acessado em Setembro de 2013

²⁵ <http://www.android.com/>, Acessado em Setembro de 2013

BrowserQuest é uma mistura de múltiplas tecnologias. O mecanismo de processamento manipula múltiplos elementos Canvas colocados um sobre o outro, como camadas gráficas. Uma API de áudio maneja os sons e as músicas do jogo, enquanto o localStorage é utilizado para salvar os personagens do jogador sobre o lado do cliente. A utilização do WebSockets capacita os browsers a abrirem uma conexão persistente bidirecional com um servidor. No BrowserQuest, isto é o que permite pessoas a jogarem e a se comunicarem comitadamente em tempo real sem a necessidade de qualquer outra conexão, e com baixa latência, ou seja praticamente em tempo real.

4.1 Considerações Finais

Esse capítulo apresentou um jogo que mostra que o HTML 5 tem potencial para o desenvolvimentos de jogos para web.

No próximo capítulo, será desenvolvido o *guideline* para a construção de jogos educativos, desde o processo de se obter a ideia até os testes. Serão exemplificados os conceitos vistos no capítulo anterior.

5 Guideline

A criação de um jogo inicia quando o *designer* tem uma ideia, rapidamente começando o processo criativo. Nesse processo o *designer* deve estar bem amparado por uma equipe, pois a criação de um jogo equivale a uma produção cinematográfica. Cada um desempenhando seus papéis e ao mesmo tempo estando conectados e unidos entre si, de acordo com as etapas, visando o mesmo ideal, que é o jogo.

A criação é atribuída à característica do *designer* em ver o mundo diferente da maioria das pessoas. A realidade em sua volta, para ele, é um conjunto de sistemas que interagem entre si, conseguindo realizar diversas tarefas. Assim quando ele olha o mundo, vê desafios, estruturas, jogos, em tudo no que a estrutura do comportamento da sociedade propicia diversos jogos.

Segundo Marcelo & Pescuite (2009), alguns exemplos são:

- A Bolsa de Valores é uma excelente fonte de inspiração, o mecanismo de subida e descida de ações pode ser adaptado a um jogo; e
- O conflito político entre dois países também pode ser transformado em um jogo de guerra.

5.1 Profissionais de desenvolvimento de games

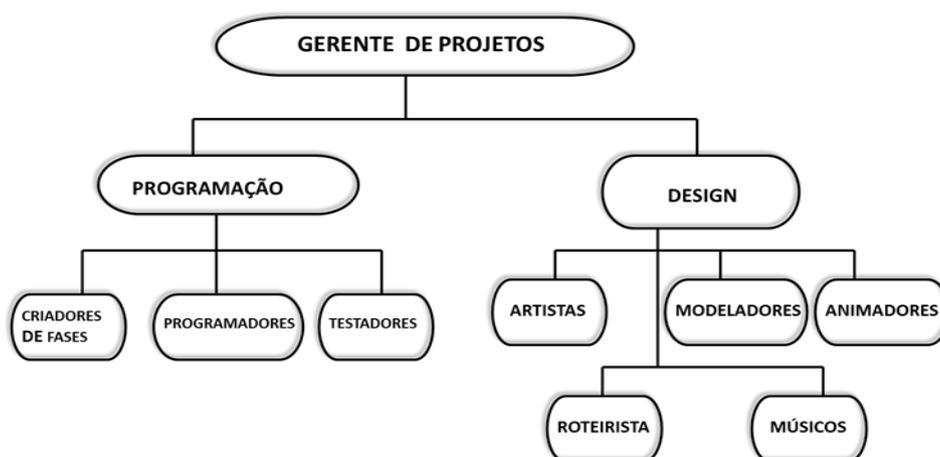
Como sugere G1 (2006), a criação de um jogo envolve um conjunto de profissionais, ilustrado nas figuras 5.1 e 5.2, cada um responsável por uma parte dele. Abaixo segue um exemplo de divisão das etapas do desenvolvimento e seus respectivos profissionais, segundo FABIANO (2010):

- **Gerencia de Projetos** – O gerente é responsável em fazer reuniões de acompanhamento com a equipe de desenvolvimento, as quais são importantes para manter os membros da equipe atualizados sobre a situação dos projetos e o comprometimento do mesmo. Também realiza correções quando houverem;
- **Equipe de Som** – pessoal responsável por criar/editar todos os sons do jogo, compor trilhas sonoras, vozes e efeitos especiais;
- **Roteiro** – É desenvolvido pelo *Game Designer*, o qual vai imaginar e escrever a história do jogo, fazendo com que fique interessante, atrativa e coerente.

- **Desenho de Conceito** – É o Desenhista Conceitual/Artista Gráfico, responsável por desenvolver a criação, dar vida às ideias do roteirista, fazendo: esboços de personagens e ambientes (os desenhos podem ser feitos a mão), pintura e colorização em programas como o Photoshop²⁶ com *tablets*, na forma 2D, com o intuito de expor como serão tais elementos;
- **Modelagem/Animação** – É o profissional de animação e modelagem 3D. Cabe a ele passar os desenhos 2D para 3D, desde os personagens e cenários à animação dos jogos. Fica também com a parte de criar noções de espaço, profundidade, iluminação e dimensões do ambiente do jogo;
- **Programação** – É o Programador o profissional responsável pelo desenvolvimento do jogo. Vai unir o trabalho de todos os profissionais, desde controle de câmera e posicionamento de objetos até o controle de entrada do meio externo, da física do jogo, além de outros mais; e
- **Pesquisadores e Professores** – Para aspectos pedagógicos e lúdicos é importante o acompanhamento de professores ou pesquisadores da área que auxiliam com os conhecimentos e experiências voltados para o conteúdo educativo que se deseja tratar.

Figura 5.1 - Profissionais para desenvolvimento de games em geral

PROFISSIONAIS DOS GAMES



Fonte: FABIANO (2010)

²⁶ <http://www.adobe.com/br/products/photoshop.html>, Acessado em Setembro de 2013

Figura 5.2 - Equipe Interdisciplinar para jogos educativos



Fonte: O Autor

5.2 O Processo de Design de um Jogo

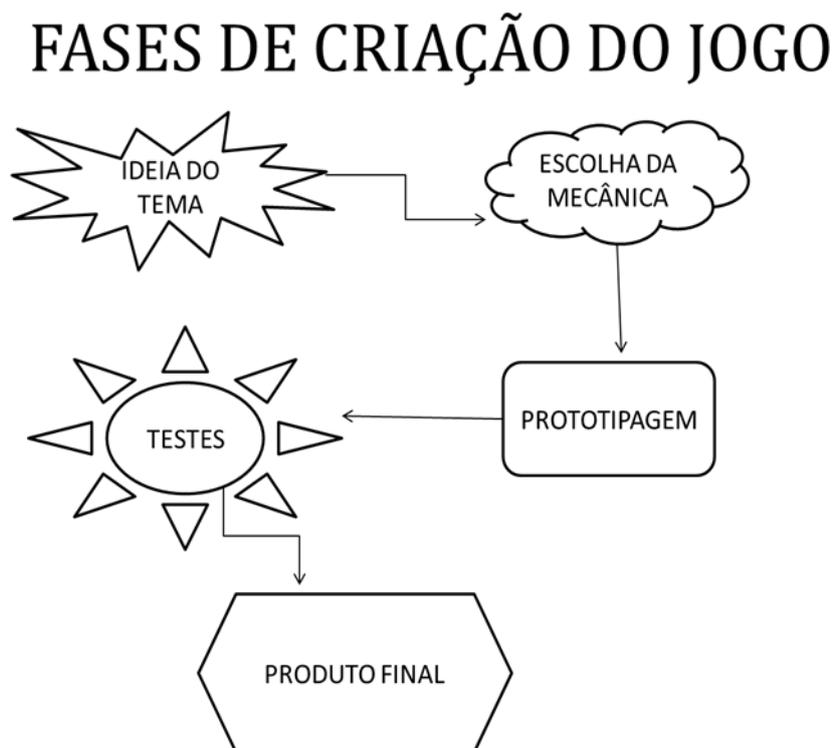
Para criar um novo sistema de jogo, o primeiro elemento a ser pensado é o jogador. Isto determinará o sucesso do jogo, pois deve-se criar o jogo pensando no jogador.

O processo pode ser dividido nas seguintes fases de criação do Jogo, ilustrado, como exemplo, na figura 5.3:

1. **Ideia e Conceito do Jogo** – inicia o projeto do jogo. Pode ser em uma folha de papel (real ou eletrônica), colocando as ideias para ser entendidas, numa construção de um sistema primitivo que será a semente do jogo em si. O ponto de partida de qualquer jogo é a pesquisa, seja na Internet, em biblioteca, para saber se tem algo similar ao tema previsto. A pesquisa serve principalmente para se obter informações sobre o tema e descobrir outros pontos que possam ser explorados;
2. **Conceito do Jogo** – Nessa fase, se obtém o primeiro documento que o *designer* deve elaborar. Ele contém uma introdução do jogo, com informações sobre o título, gênero, configurações, plataforma e demais itens que o *designer* achar importante [PEREIRA, 2006];
3. **Mecânica** – procedimentos e regras. A mecânica mostra os objetivos do jogo, dos mais básicos aos que incluem a finalização do mesmo. É como os jogadores devem fazer para alcançar os objetivos primários para conquistar o objetivo central do jogo, isto é, a criação do modelo que se adéqua às necessidades da ideia concebida. Existem diversos tipos de mecânicas como leilões, desenvolvimento econômico, dominação de áreas, plataformas entre outros. Precisa ser uma mecânica para o sistema, se for ideal, podendo ser substituída durante o desenvolvimento, caso se ache necessário. A jogabilidade pode ser simples ou com muitos detalhes. Nesse ponto deve se ter cuidado para não se perder pela complexidade;

4. Desenvolvimento de um protótipo físico, mesmo que seja de computador. São modelos construídos para simular a aparência e a funcionalidade do produto em desenvolvimento, uma representação da interface. Por meio de um protótipo, os futuros usuários do software e desenvolvedores podem interagir, avaliar, alterar e provar as características mais marcantes da interface e da funcionalidade da aplicação; e
5. Segue-se com os testes até se encontrar uma versão que possa ser considerada final. Comprovada a necessidade, depois dos testes, podem ser feitas as modificações, ou se apresentar um desempenho razoável, pode ser considerado pronto para entrar em fase de produção.

Figura 5.3 - Fases de Criação do Jogo



Fonte: MARCELO & PESCUITE (2009)

5.3 Design de Jogos Educativos – Design Instrucional

É o processo de planejar, desenvolver e aplicar métodos, técnicas e atividades de ensino a fim de facilitar a aprendizagem. Nesse novo conceito de aprendizagem é importante responder o um questionamento como: Para quem o projeto será

desenvolvido? Para que o projeto será desenvolvido? E como o projeto será desenvolvido? [FILATRO & PICONEZ 2004; SILVA & CASTRO 2009], citado por Barbosa Neto (2012).

Definição de design instrucional segundo Filatro (2004) citado por Barbosa Neto (2012):

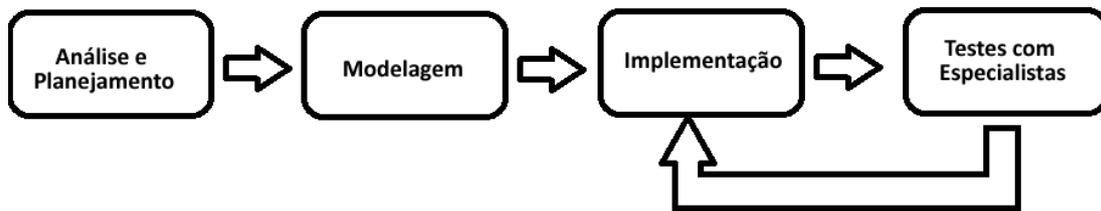
[...] ação intencional e sistemática de ensino, que envolve o planejamento, o desenvolvimento e a utilização de métodos, técnicas, atividades, materiais, eventos e produtos educacionais em situações didáticas específicas, a fim de facilitar a aprendizagem humana a partir dos princípios de aprendizagem e instrução conhecidos. (FILATRO, A. Design Instrucional Contextualizado. Rio de Janeiro, RJ, 2004, p.64-65)

O profissional *designer* instrucional tem a função de elaborar estratégias que consolidem uma relação benéfica entre a tecnologia e a educação, com uma aprendizagem colaborativa e autônoma. Para Filatro e Piconez (2004), citado por Barbosa Neto (2012), o designer instrucional se dedica a planejar, preparar, projetar, produzir e publicar textos, imagens, gráficos, sons e movimentos, simulações, entre outros mecanismos de efetiva contextualização.

Cada jogo possui uma estrutura única e por mais que se assimilem possuem características de jogabilidade específicas. Além disso, para os possíveis impactos sobre a aprendizagem, a representatividade de uma situação real pode influenciar e ser envolvente para os jogadores, pois o realismo é uma das características dos jogos que captam a atenção do jogador. É muito importante ter o cuidado de combinar jogo e educação sem perder o foco do divertimento, para que o aluno tenha um interesse maior em aprender tornando gratificante para ele. É por isso que quando pensar em criar um jogo, tem-se que sempre pensar em seu público alvo.

O processo de concepção de jogos educativos se diferencia em algumas etapas conforme descrito nas seguintes fases: Análise e Planejamento, Modelagem, Implementação e Testes com Especialistas, ilustrado na figura 5.4.

Figura 5.4 - Etapas de desenvolvimento



Fonte: Barbosa Neto (2012)

5.3.1 Análise e Planejamento

Primeiramente é preciso considerar o produto a ser desenvolvido, definir o tema e o objetivo da aplicação. Depois, considerar as aplicações similares e os recursos disponíveis e fazer a coleta dos dados e a sua análise. É necessário conhecer o público alvo, no caso como esse produto será usado, quando, onde e para que? E o que é esperado com o uso da sua aplicação?

Definindo o público alvo, aqui no exemplo o jogo vai ser aplicado no processo de aprendizagem. Deve-se coletar informações sobre o assunto, e se já existirem no mercado, pensar em um diferencial a ser oferecido. E todas as ações para o desenvolvimento do jogo serão documentadas com um escopo, depois se pode utilizar qualquer metodologia de gerenciamento de projeto de *software*. Deve-se priorizar os requisitos e definir o cronograma.

5.3.2 Modelagem

É a técnica de construção de modelos onde o objetivo é: clarificar as ideias para facilitar a compreensão para aprovação do sistema antes da sua construção. Nela deve-se citar todos os personagens, cenários e animação dos jogos e também com a parte de criar noções de espaço, profundidade, iluminação e dimensões do ambiente do *jogo*.

Segundo FALKEMBACH (2005), citado por BARBOSA (2012) a modelagem para o desenvolvimento de aplicações educacionais divide-se na criação de 03(três) tipos de modelos diferentes, ilustrado na figura 5.5:

1. Modelo Conceitual - se refere ao domínio de como o conteúdo pedagógico será disponibilizado ao aluno, todo o detalhamento, as estrutura de acesso a uma

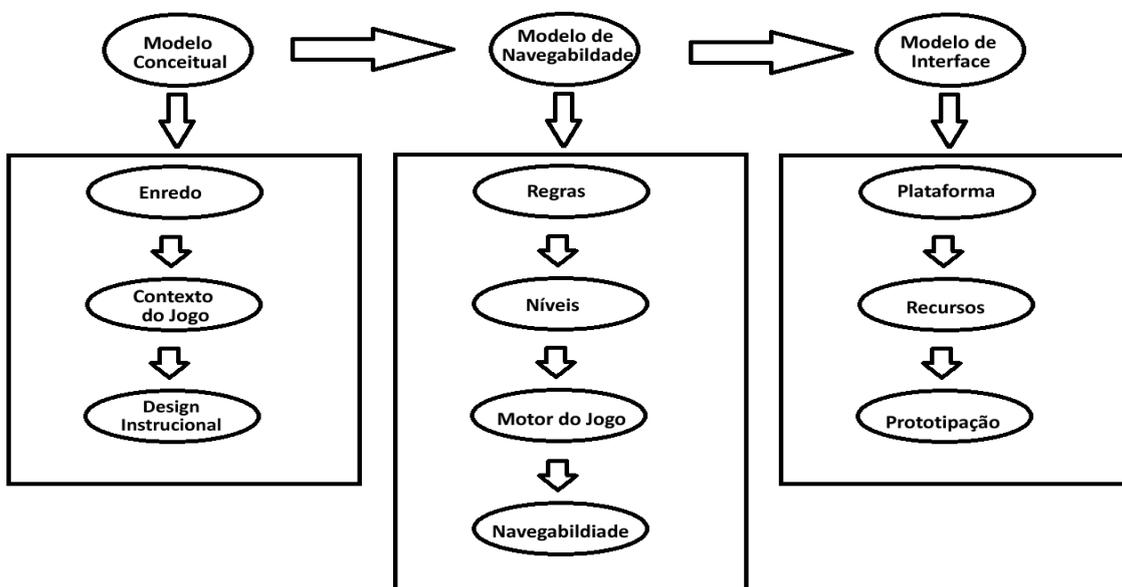
interface e é a organização das informações e das mídias. O primeiro passo é o enredo do jogo. Nela pensa-se em todo contexto do jogo, define-se os personagens de acordo com sua atividade, a sequência histórica, que é a narrativa do jogo, entre outros;

2. Modelo de Navegação – Ao ser concluído o modelo conceitual, deve ser construído o modelo de navegabilidade, definindo-se, a princípio as regras, o comportamento, quantos níveis terá o jogo, o motor a ser usado e sua jogabilidade, e as estruturas de acesso, ou seja, como serão os elos. A navegação deve ser intuitiva para evitar a desorientação do usuário e diminuir a sobrecarga cognitiva. O modelo define o uso de menus, índices, roteiros guiados, entre outros;

3. Modelo de Interface – deve ser compatível com o modelo conceitual e de navegação, ou seja, o *design* de interfaces precisa estar em harmonia com o conteúdo, deve haver um equilíbrio entre a organização das informações e a apresentação estética.

Primeiro se define a plataforma na qual vai ser construído e os recursos que serão usados pelo jogo. A construção depende da infraestrutura que será utilizada pelo jogo.

Figura 5.5 - Modelagem de um jogo educacional



Fonte: Barbosa Neto (2012)

5.4 O Processo de Implementação de um Jogo

É o processo de criar as mídias do projeto, incluindo os sons, as imagens, animações e vídeos utilizando *software* específico. Para que não haja erro conceitual

nem gramatical é preciso verificar várias vezes os textos. Deve-se colocar os créditos dos direitos autorais na fonte, mesmo para as mídias disponíveis na rede.

Para que o aluno não fique desorientado, o programador utiliza um Sistema de Autoria, para integrar todas as mídias em uma estrutura interativa permitindo uma navegação lógica e intuitiva. Essa é fase de transferir todos os dados para o computador. Depois de ter feito a implementação, é preciso testar para que se for necessário, aplicar as devidas correções [FALKEMBACH, 2005].

5.4.1 Programação do jogo

Nesta fase o programador responsável pelo desenvolvimento do jogo, vai unir todos os detalhes de todos os profissionais, vai seguir o passo a passo para programar o jogo e utilizar as ferramentas: HTML5 e JAVASCRIPT.

Para programar em HTML, Javascript e CSS, será necessário apenas um editor de texto, Notepad²⁷ ou a versão aprimorada e gratuita Notepad++²⁸, que possui, entre outras ferramentas, suporte a várias linguagens de programação, inclusive as usadas nesse *guideline*, para o Windows, TextEdit²⁹ para o Mac³⁰ e Gedit³¹ no Linux [HARRIS 2013]. Existem outras ferramentas para facilitar a programação, como o Komodo Edit³², que funciona na maioria dos sistemas operacionais e também é gratuito [HARRIS 2013]. Outras ferramentas para editar HTML, são o Adobe Dreamweaver³³, o Microsoft Expression Web³⁴ e o CoffeeCup HTML Editor³⁵ [W3SCHOOLS 1999-2013].

O código do Quadro 5.1 representa o básico para o funcionamento de uma página em HTML 5, sendo assim o código essencial para todo documento em HTML e será usado e abstraído no restante desse *guideline*. O resultado está ilustrado na Figura 5.6. Vale ressaltar que, embora não seja obrigatório para HTML, a indentação facilita a visualização da hierarquia entre os elementos.

²⁷ <http://www.computerhope.com/jargon/n/notepad.htm>, Acessado em Setembro de 2013

²⁸ <http://notepad-plus-plus.org/>, Acessado em Setembro de 2013

²⁹ <http://support.apple.com/kb/HT2523>, Acessado em Setembro de 2013

³⁰ <http://www.apple.com/pt/mac/>, Acessado em Setembro de 2013

³¹ <https://projects.gnome.org/gedit/>, Acessado em Setembro de 2013

³² <http://www.activestate.com/komodo-edit>, Acessado em Setembro de 2013

³³ <http://www.adobe.com/br/products/dreamweaver.html>, Acessado em Setembro de 2013

³⁴ <http://www.microsoft.com/expression/eng/>, Acessado em Setembro de 2013

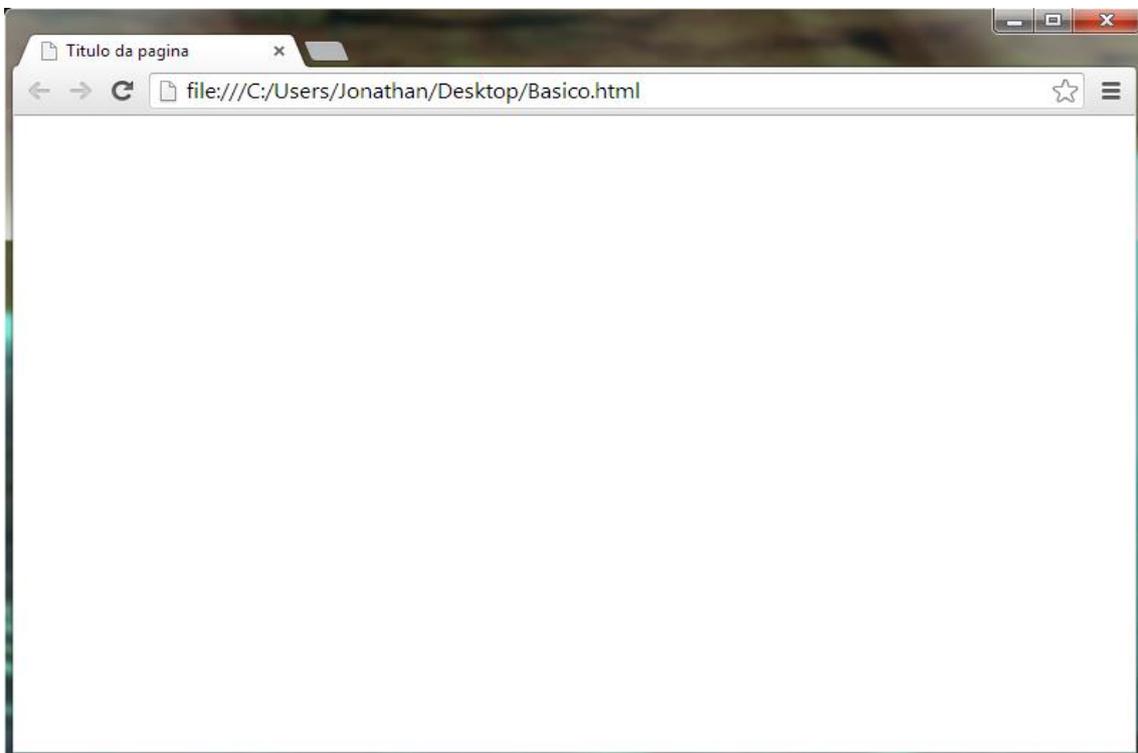
³⁵ <http://www.coffeecup.com/html-editor/>, Acessado em Setembro de 2013

Quadro 5.1 - O código básico para HTML 5

```
1 <!DOCTYPE html>
2 <html lang="pt-br">
3   <head>
4     <meta charset="UTF-8">
5     <title>Titulo da pagina</title>
6   </head>
7   <body>
8   </body>
9 </html>
```

Fonte: O Autor

Figura 5.6 - Básico para o HTML 5



Fonte: Autor

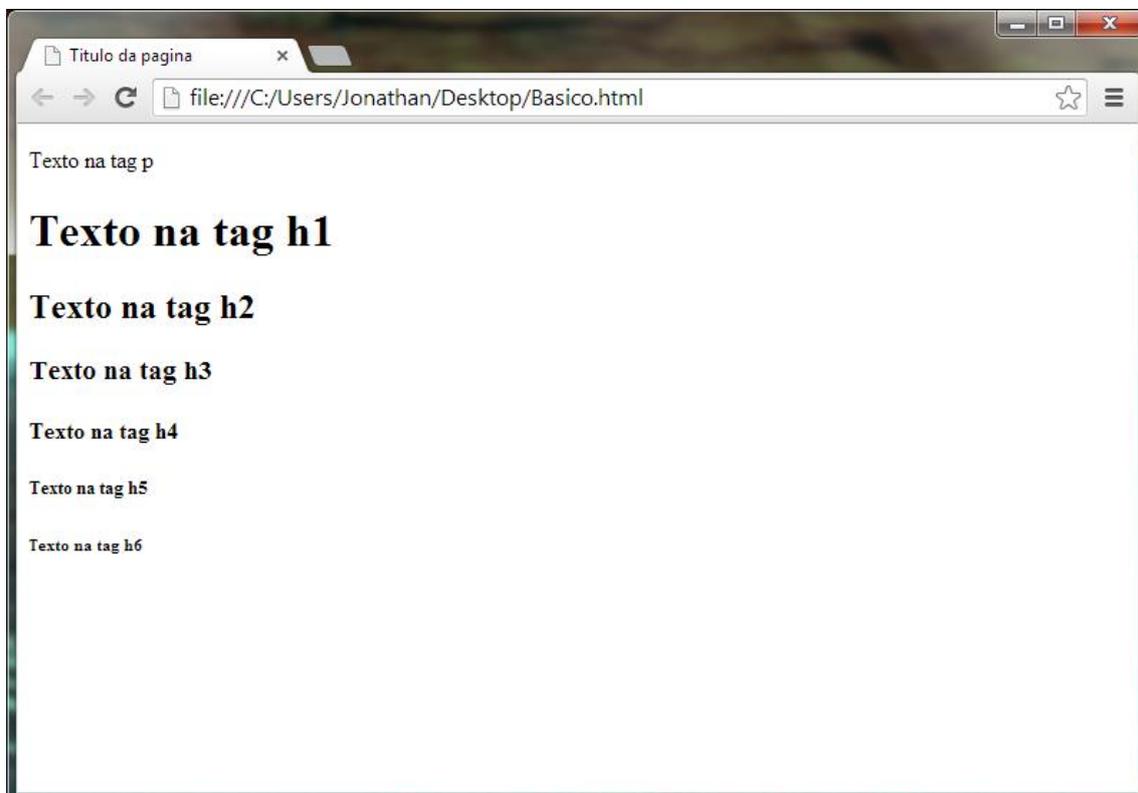
Para acrescentar texto, como foi dito anteriormente, é importante a utilização de uma tag de texto para permitir a formatação do mesmo. O Quadro 5.2 mostra um pequeno exemplo para ilustrar as diferenças entre as tags de texto. O resultado está ilustrado na Figura 5.7.

Quadro 5.2 - Código de exemplo para o uso de texto

```
1 <body>
2   <p>Texto na tag p</p>
3   <h1>Texto na tag h1</h1>
4   <h2>Texto na tag h2</h2>
5   <h3>Texto na tag h3</h3>
6   <h4>Texto na tag h4</h4>
7   <h5>Texto na tag h5</h5>
8   <h6>Texto na tag h6</h6>
9 </body>
```

Fonte: O Autor

Figura 5.7 - Diferenças entre as *tags* de texto



Fonte: O Autor

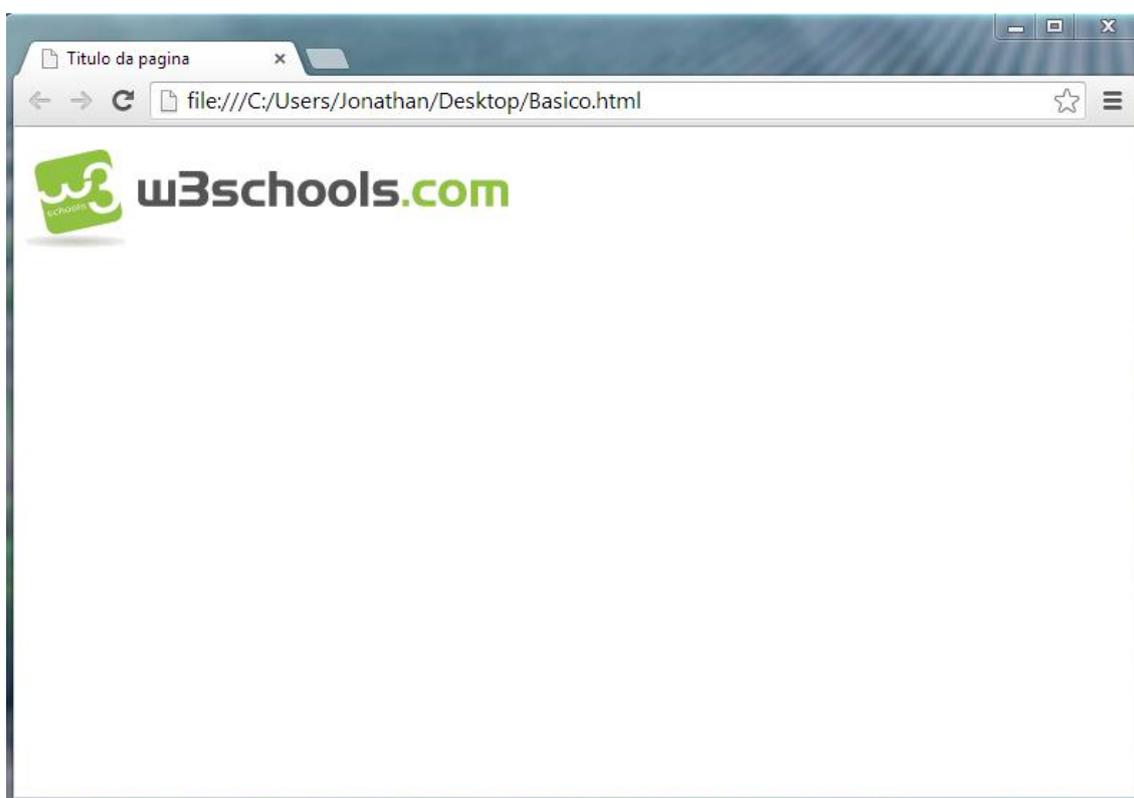
Para acrescentar uma imagem no jogo, como foi visto anteriormente, é usado a *tag* ``. O código do Quadro 5.3, ilustrado na figura 5.8, exemplifica a sua utilização. A imagem utilizada foi retirada do site da W3SCHOOLS e, para o resultado desse exemplo, está salva no mesmo diretório do arquivo html, portanto para o atributo *src*, nesse caso, só é necessário fornecer o nome do arquivo.

Quadro 5.3 - Utilização da tag de imagem

```
1 <body>
2   <p>
3     
4   </p>
5 </body>
6
```

Fonte: O Autor

Figura 5.8 – Resultado do código do Quadro 5.3



Fonte: Autor, sendo a imagem retirada do site da W3SCHOOLS

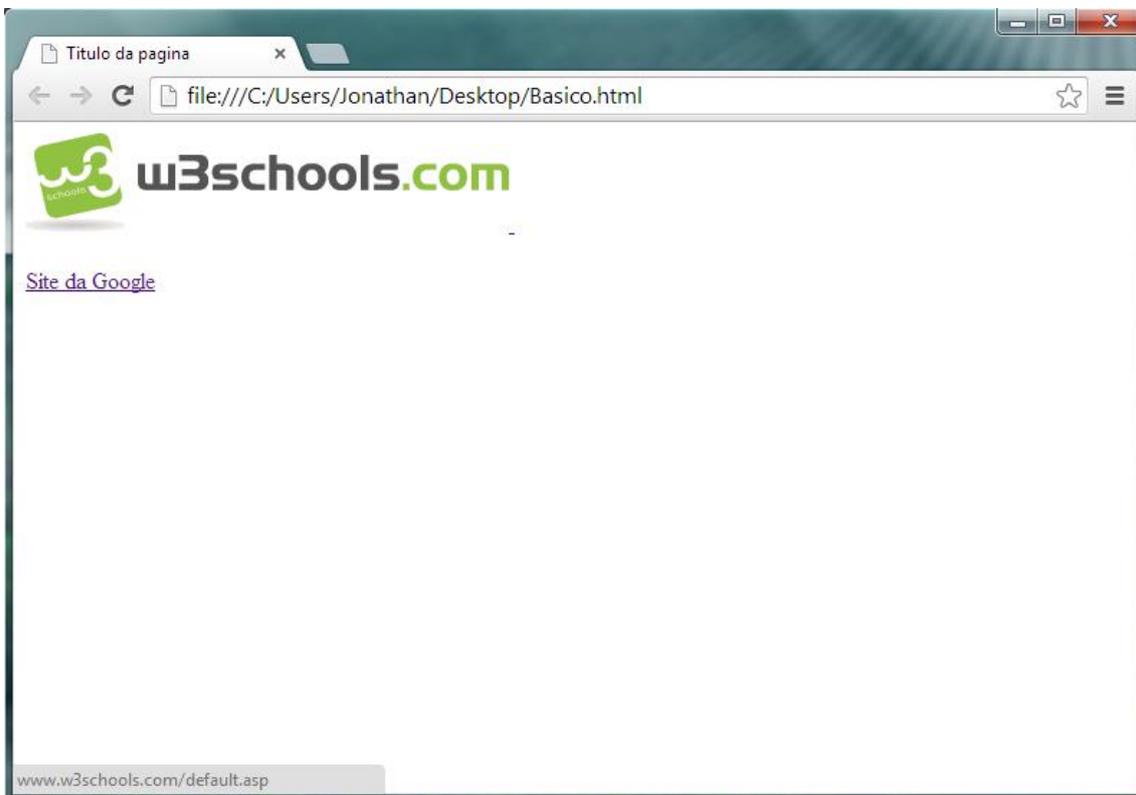
Pode-se utilizar links para realizar a mudança de telas em um jogo. O exemplo no Quadro 5.4, ilustrado na Figura 5.9, mostra uma imagem contida em uma *tag* de link, e também a utilização de um texto contido em uma segunda *tag* *<a>*. Foi utilizado também a *tag* de quebra de linha *
* para separar os elementos na página.

Quadro 5.4 - Exemplo de utilização da *tag* de link

```
1 <body>
2     <a href="http://www.w3schools.com/default.asp">
3         
4     </a>
5     <br><br>
6     <a href="https://www.google.com.br/">Site da Google</a>
7 </body>
8
```

Fonte: O Autor

Figura 5.9 – Resultado do exemplo do Quadro 5.4



Fonte: O Autor

Como visto anteriormente, é utilizado o CSS para trabalhar na estética e no posicionamento dos elementos de HTML. Também foi visto diferentes formas de inserir o código do CSS no documento HTML. O exemplo do Quadro 5.5, ilustrado na Figura 5.10, mostra como é inserido o CSS em linha para alterar o estilo de um elemento apenas.

Quadro 5.5 - Utilizando CSS em linha

```
1 <body>
2   <p style="font-family:'Times New Roman'; color:red; font-size:20px;">
3     Um texto que sera formatado.
4   </p>
5   <p>Um texto sem formatacao.</p>
6 </body>
7
```

Fonte: O Autor

Figura 5.10 – Exemplo do exemplo do Quadro 5.5



Fonte: O Autor

Nesse exemplo, tem-se uma *tag* `<p>` em que o conteúdo dela, no caso do texto “Um texto que sera formatado”, estará utilizando a fonte Times New Roman, com o tamanho de 20 pixels na cor vermelha.

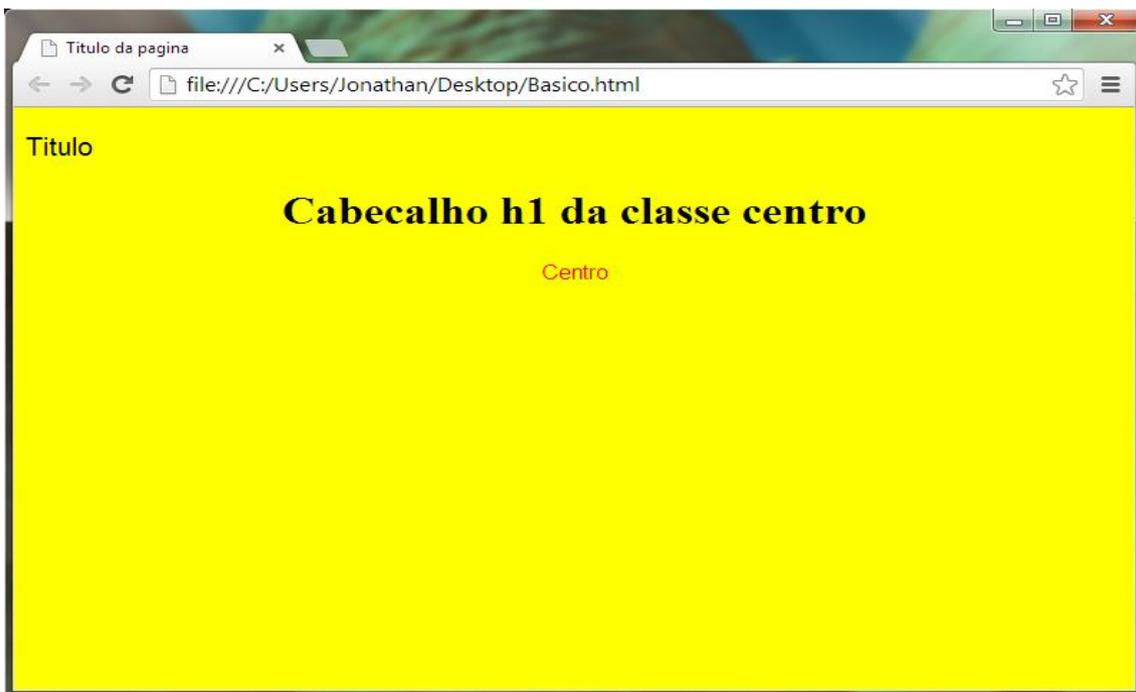
O exemplo do Quadro 5.6, ilustrado pela figura 5.11, mostra a utilização do CSS interno, como explicado anteriormente, essa forma de utilização do CSS permite alterar vários elementos de uma mesma página de HTML.

Quadro 5.6 - Utilização de CSS interno

```
1 <head>
2   <style type="text/css">
3     body {
4       background-color: yellow;
5     }
6     p {
7       font-family:Arial;
8     }
9     #titulo {
10      font-size:20px;
11    }
12    .centro {
13      text-align:center;
14    }
15    p.centro {
16      color:red;
17    }
18  </style>
19 </head>
20 <body>
21   <p id="titulo">Titulo</p>
22   <h1 class="centro">Cabecalho h1 da classe centro</h1>
23   <p class="centro">Centro</p>
24 </body>
25
```

Fonte: O Autor

Figura 5.11 – Resultado do exemplo do Quadro 5.6



Fonte: O Autor

No exemplo, o estilo aplicado na tag <body> tem efeito em todo o conteúdo do corpo da página, nesse caso, a página inteira possui uma cor de fundo amarela. Todos os elementos que possuírem a tag <p> terão seus respectivos textos usando a fonte Arial. O elemento #titulo, representa um elemento com uma id, única, com valor titulo, portanto, apenas o elemento que possuir o atributo id="titulo" receberá a propriedade. Já o .centro, representa uma classe com valor "centro", e as propriedades serão aplicadas a todos os elementos que possuírem o atributo class="centro", mesmo se forem elementos distintos, ou seja, uma tag <p class="centro"> e uma tag <h1 class="centro"> receberão os efeitos das propriedades descritas para .centro, o que não acontece com p.centro. Segundo o exemplo, apenas a tag <p class="centro"> terá a cor em vermelho, a tag <h1 class="centro"> continuará com a cor padrão.

Supondo um arquivo estilo.css, com o código exemplificado no Quadro 5.7.

Quadro 5.7 - Exemplo de CSS Externo

```
1 body {
2     background-color: black;
3 }
4 p {
5     font-family:Tahoma;
6     color: white;
7 }
8
```

Fonte: O Autor

O exemplo do Quadro 5.8 mostra como é importado o arquivo estilo.css.

Quadro 5.8 - Importando arquivo CSS

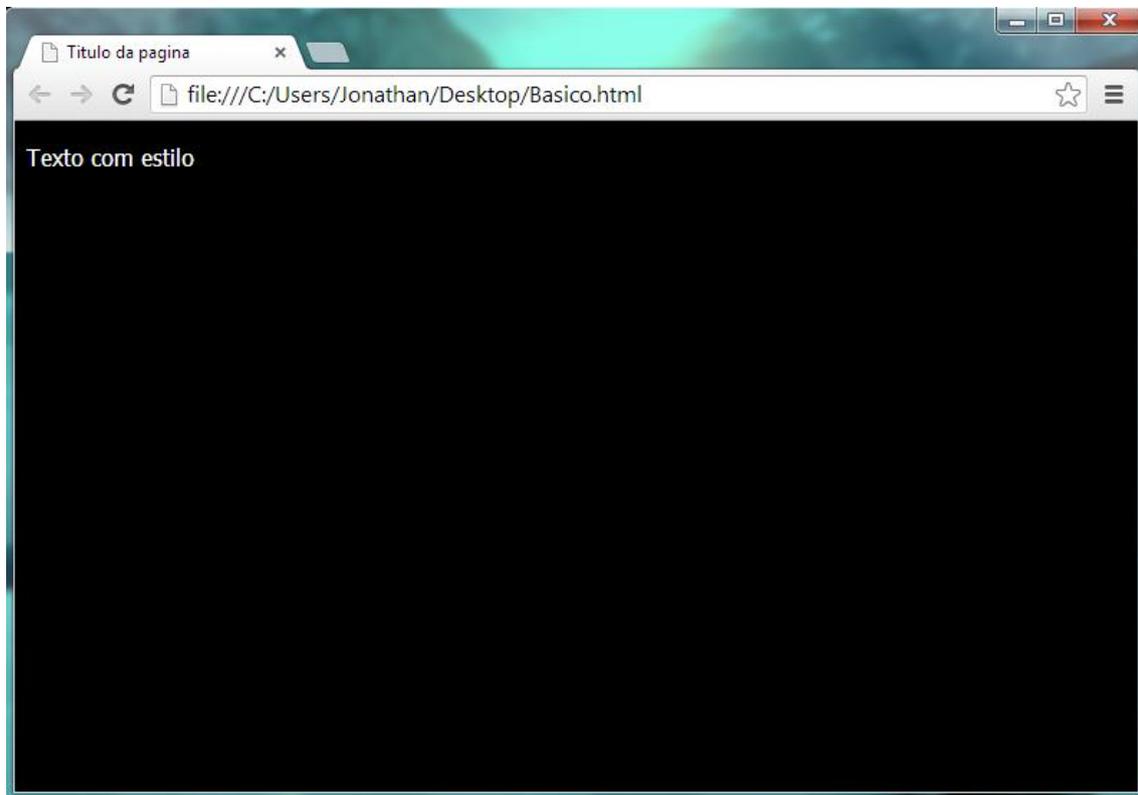
```
1 <head>
2     <link rel = "stylesheet" type = "text/css" href = "estilo.css" />
3 </head>
4 <body>
5     <p>Texto com estilo</p>
6 </body>
7
```

Fonte: O Autor

Todas as páginas em HTML que importarem o arquivo estilo.css como no exemplo acima, terá a cor de fundo do <body> em preto e todos as tags <p> terão fonte em Tahoma, na cor branca, como ilustrado na Figura 5.12. Nota-se que nesse caso o

atributo href define onde se encontra o arquivo, que no exemplo, o arquivo estilo.css está no mesmo diretório que o arquivo html.

Figura 5.12 - Utilizando CSS externo



Fonte: O Autor

O Quadro 5.9 apresenta um exemplo de criação de um formulário bem simples para pedir *login* e senha do usuário, ilustrado pela figura 4.16. Formulários podem fazer parte de um jogo em forma de tela de *login*, área de bate papo, criação de personagens ou mesmo para uma tela para cadastro de usuário, entre outras utilidades.

Quadro 5.9 - Formulário de login e senha

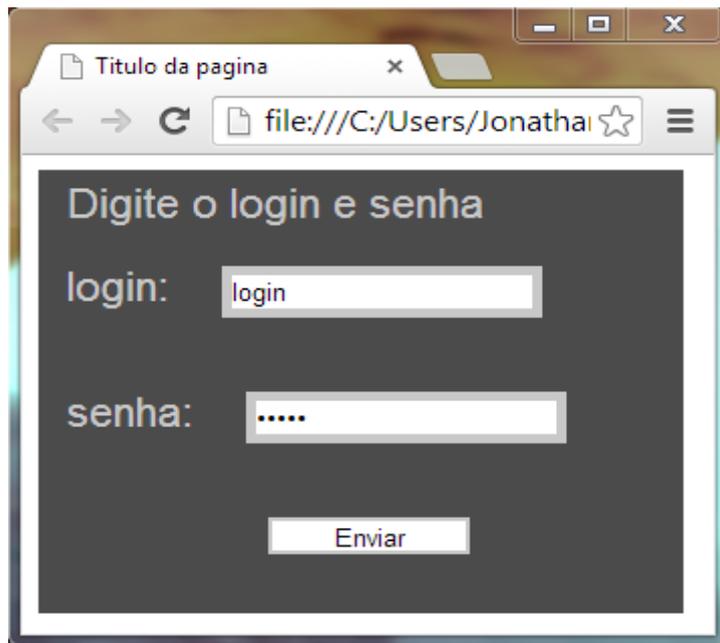
```

1 <head>
2   <style type="text/css">
3     .menu {
4       width:320px;
5       height:240px;
6       background-color:rgba(0,0,0, 0.7);
7     }
8     .menu label {
9       font-family: Arial, "Times New Roman", Tahoma;
10      font-size: 22px;
11      color: rgb(200,200,200);
12    }
13    .campo_de_texto {
14      border: 5px solid rgb(200,200,200);
15      background-color:#FFFFFF;
16      margin:20px;
17      clear:both;
18    }
19    .menu button {
20      color: #000000;
21      background-color: #FFFFFF;
22      border: 2px solid rgb(200,200,200);
23      width: 100px;
24      height: 20px;
25    }
26    .menu button {
27      position:relative;
28      top: 20px;
29      left: 100px;
30      cursor:pointer;
31    }
32    .menu fieldset {
33      border-style:none;
34    }
35  </style>
36 </head>
37
38 <body>
39 <div class="menu">
40 <form>
41   <fieldset>
42     <label for="campos">Digite o login e senha</label>
43     <div id="campos">
44       <label for="campo_login">login: </label>
45       <input type="text" id="campo_login" class="campo_de_texto">
46       <label for="campo_senha">senha: </label>
47       <input type="password" id="campo_senha" class="campo_de_texto">
48     </div>
49     <button type = "button">Enviar</button>
50   </fieldset>
51 </form>
52 </div>
53 </body>
54

```

Fonte: O Autor

Figura 5.13 – Resultado do exemplo do Quadro 5.9



Fonte: O Autor

O exemplo do Quadro 5.10, acrescenta no formulário do exemplo anterior, uma função criada para receber o conteúdo dos campos de texto do formulário ao apertar o botão, para também guardar cada conteúdo em uma variável, exibir uma mensagem de saudação ao usuário e esconder a tela de *login*.

Quadro 5.10 - Adicionando JavaScript ao formulário

```
1 <head>
2   <script type = "text/javascript">
3     function login(){
4       var var_campo_login = document.getElementById("campo_login");
5       var valor_login = var_campo_login.value;
6       var var_campo_senha = document.getElementById("campo_senha");
7       var valor_senha = var_campo_senha.value;
8       var var_tela_login = document.getElementById("tela_login");
9       var_tela_login.style.display="none";
10      alert("Seja bem vindo, " + valor_login + "!");
11    }
12  </script>
13 </head>
14
```

Fonte: O Autor

Na *<body>* deve ser acrescentado na *tag <div>* mais externa, um *id* para que se possa alterar o seu estilo através da função criada:

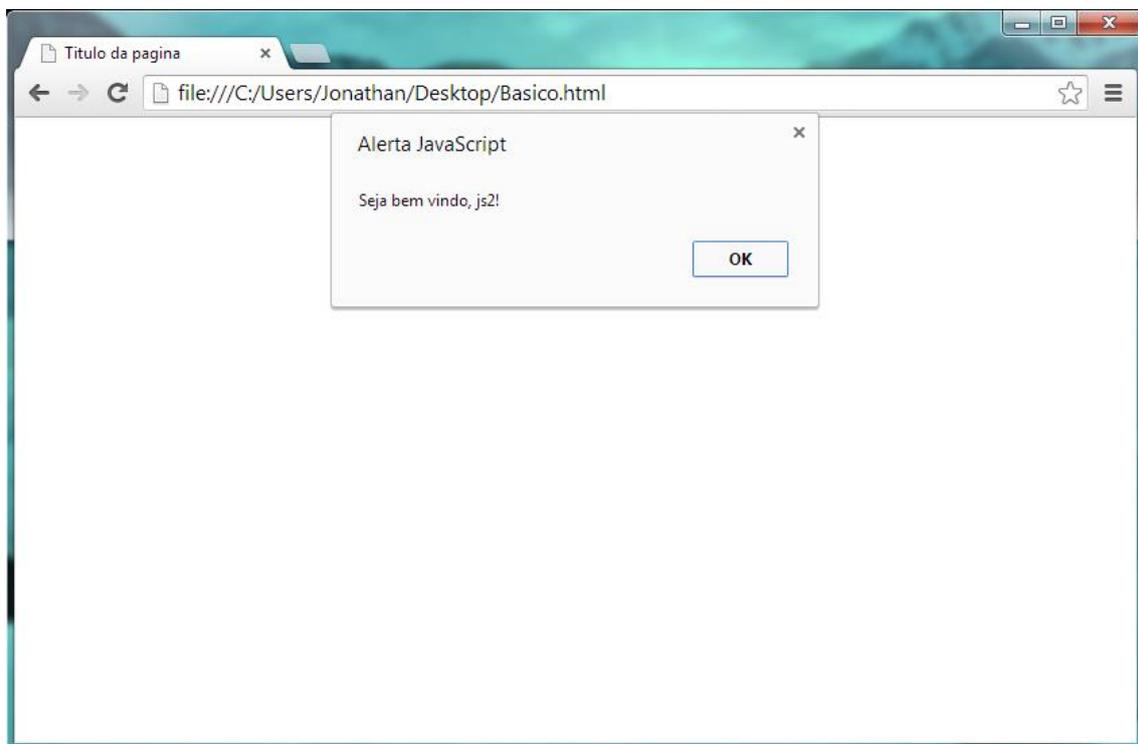
```
<div class="menu" id="tela_login">
```

E foi acrescentado ao botão o atributo de evento *onclick*:

```
<button type = "button" onclick = "login()">Enviar</button>
```

Para colocar um elemento HTML em uma variável, para poder alterá-la depois, é utilizada a função do documento `getElementById`, passando o *id* do elemento, como mostrado no exemplo. Então, são acessados os seus atributos, como *value*, para se obter o conteúdo que foi digitado, e *style*, para alterar o valor de alguma propriedade de CSS. A Figura 5.14 mostra o resultado, após realizadas as modificações acima, ao preencher o *login* com a palavra *js2*, digitar um valor qualquer para senha e pressionar o botão.

Figura 5.14 - Resultado ao clicar o botão



Fonte: Autor

O exemplo contido nos quadros 5.11 e 5.12 gera uma página dividida em cinco campos: o cabeçalho, a área da esquerda, com o evento de clique, a área do meio que gera elementos com o evento de arraste, e a área da direita com o evento de *drop*. A área de baixo apenas atualiza o somatório dos números, gerados pela área do centro, que são arrastados para a área da direita.

Quadro 5.11 - Exemplo de uso de evento de mouse parte 1

```
1 <head>
2 <style type="text/css">
3   *{
4     padding:0;
5     margin:0;
6     cursor:default;
7   }
8   body {
9     width: 640px;
10    height:480px;
11  }
12  span {
13    font-size: 30px;
14  }
15  #cabecalho{
16    width:100%;
17    height: 40px;
18    text-align: center;
19  }
20  #caixaEsquerda, #caixaCentro, #caixaDireita{
21    height: 400px;
22    float:left;
23  }
24  #caixaEsquerda, #caixaEsquerda *{
25    cursor:pointer;
26  }
27  #caixaEsquerda, #caixaDireita{
28    width:25%;
29    background-color:gold;
30  }
31  #caixaCentro{
32    width:50%;
33    overflow:hidden;
34  }
35  #caixaCentro span{
36    background-color:#CCC;
37    margin: 50px 20px;
38    cursor:move;
39    float:left;
40  }
41  #rodape{
42    width:100%;
43    height: 40px;
44    text-align:center;
45    clear:both;
46  }
47 </style>
```

Fonte: O Autor

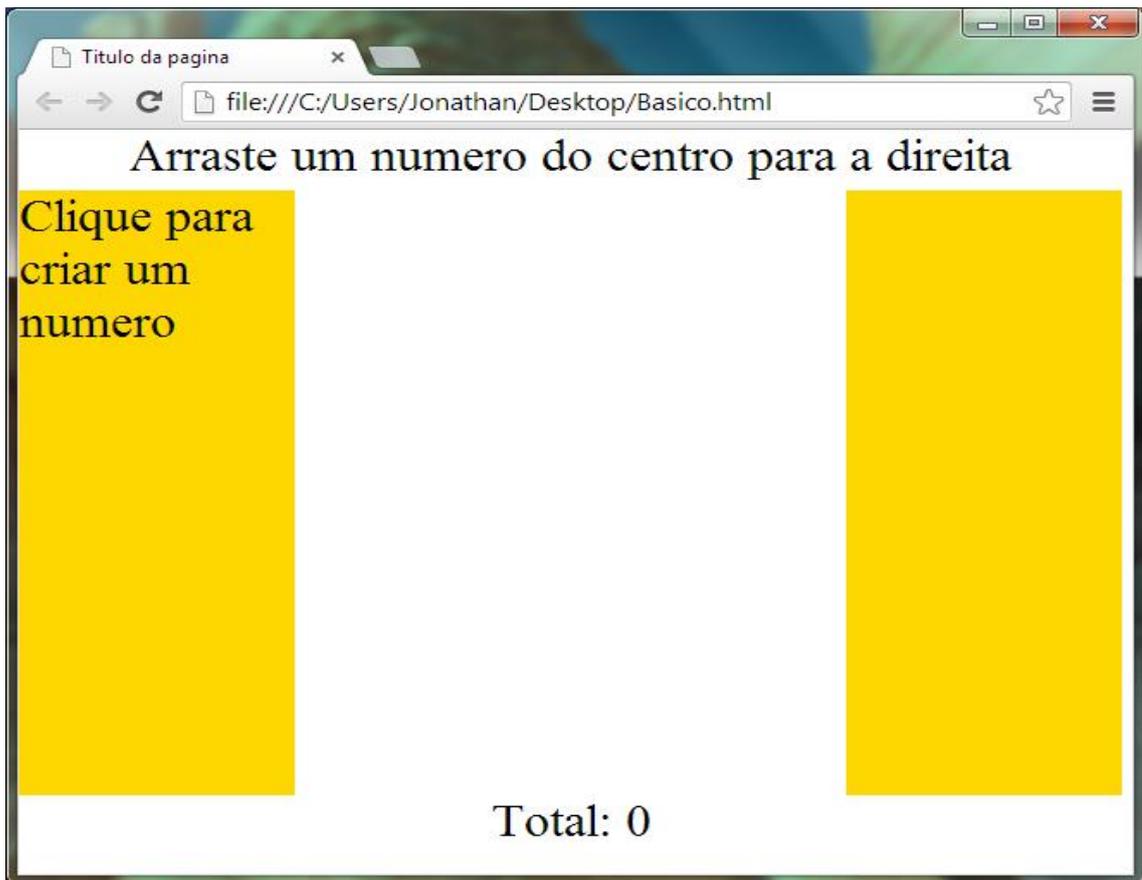
Quadro 5.12 - Exemplo de uso de evento de mouse parte 2

```
48 <script type="text/javascript">
49     function criarNumero(){
50         var vCaixaCentro = document.getElementById("caixaCentro");
51         var vNumero = Math.ceil(Math.random() * 100);
52         var vSpan = document.createElement("span");
53         vSpan.innerHTML = vNumero;
54         vSpan.setAttribute("draggable", "true");
55         vSpan.setAttribute("ondragstart", "drag(event)");
56         vSpan.setAttribute("ondragend", "dragend(event)");
57         vCaixaCentro.appendChild(vSpan);
58     }
59     function permitirDrop(ev){
60         ev.preventDefault();
61     }
62     function drag(ev){
63         ev.dataTransfer.setData("number", ev.target.innerHTML);
64     }
65     function dragend(ev){
66         var vSpan = ev.target;
67         var vCaixaCentro = document.getElementById("caixaCentro");
68         vCaixaCentro.removeChild(vSpan);
69     }
70     function drop(ev){
71         ev.preventDefault();
72         var valor = Number(ev.dataTransfer.getData("number"));
73         var vCampoTotal = document.getElementById("totalSoma");
74         var vTotal = Number(vCampoTotal.innerHTML);
75         vTotal = vTotal + valor;
76         vCampoTotal.innerHTML = vTotal;
77     }
78 </script>
79 </head>
80 <body>
81     <div id="cabecalho">
82         <span>Arraste um numero do centro para a direita</span>
83     </div>
84     <div id="caixaEsquerda" onclick="criarNumero()">
85         <span>Clique para criar um numero</span>
86     </div>
87     <div id="caixaCentro">
88     </div>
89     <div id="caixaDireita" ondrop="drop(event)" ondragover="permitirDrop(event)">
90     </div>
91     <div id="rodape">
92         <span>Total: <span id="totalSoma">0</span> </span> </div>
93 </body>
94 </body>
95
```

Fonte: O Autor

A figura 5.15 mostra a tela inicial do programa.

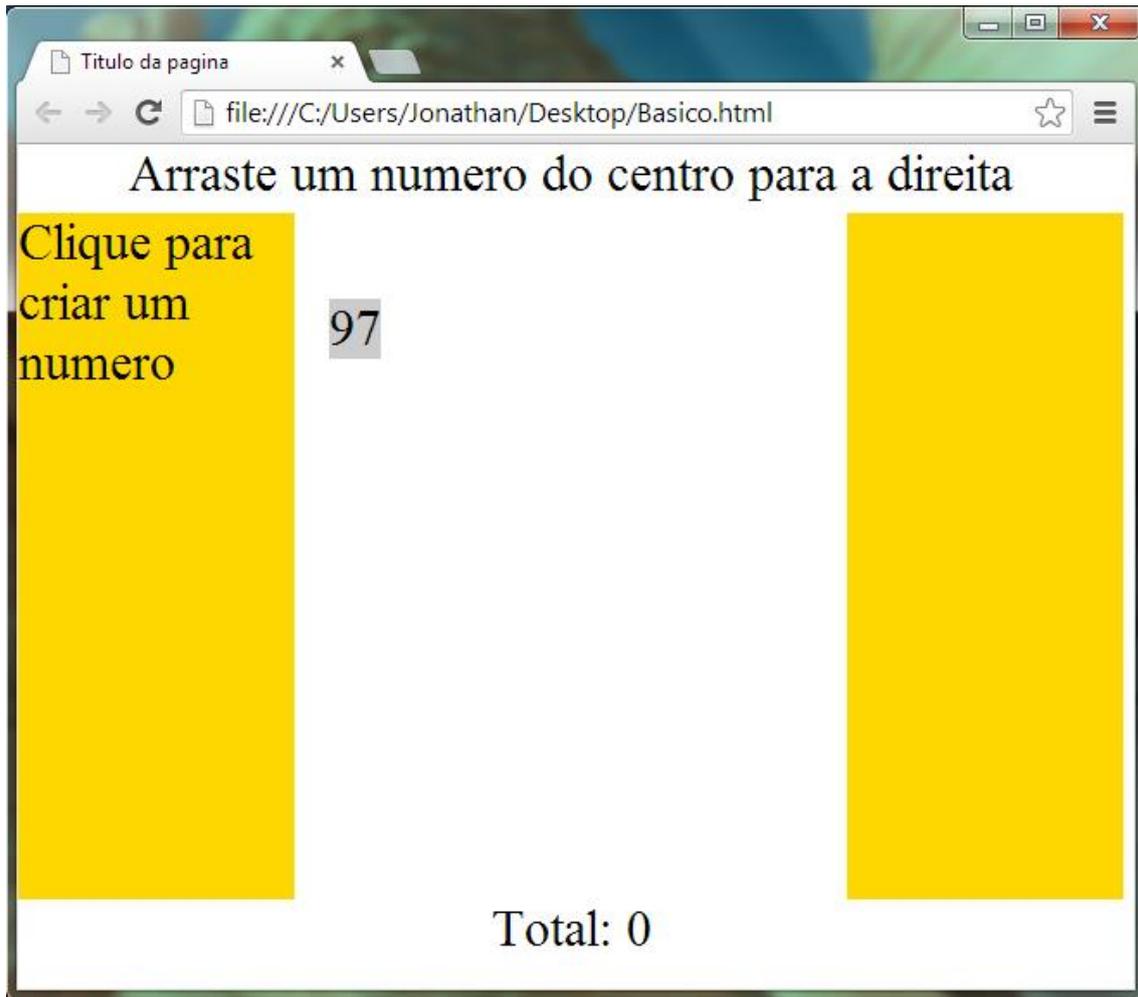
Figura 5.15 - Exemplo *Drag* - Tela Inicial



Fonte: O Autor

Ao clicar na área da esquerda, um número é gerado aleatoriamente, entre 1 e 100, como ilustrado na figura 5.16.

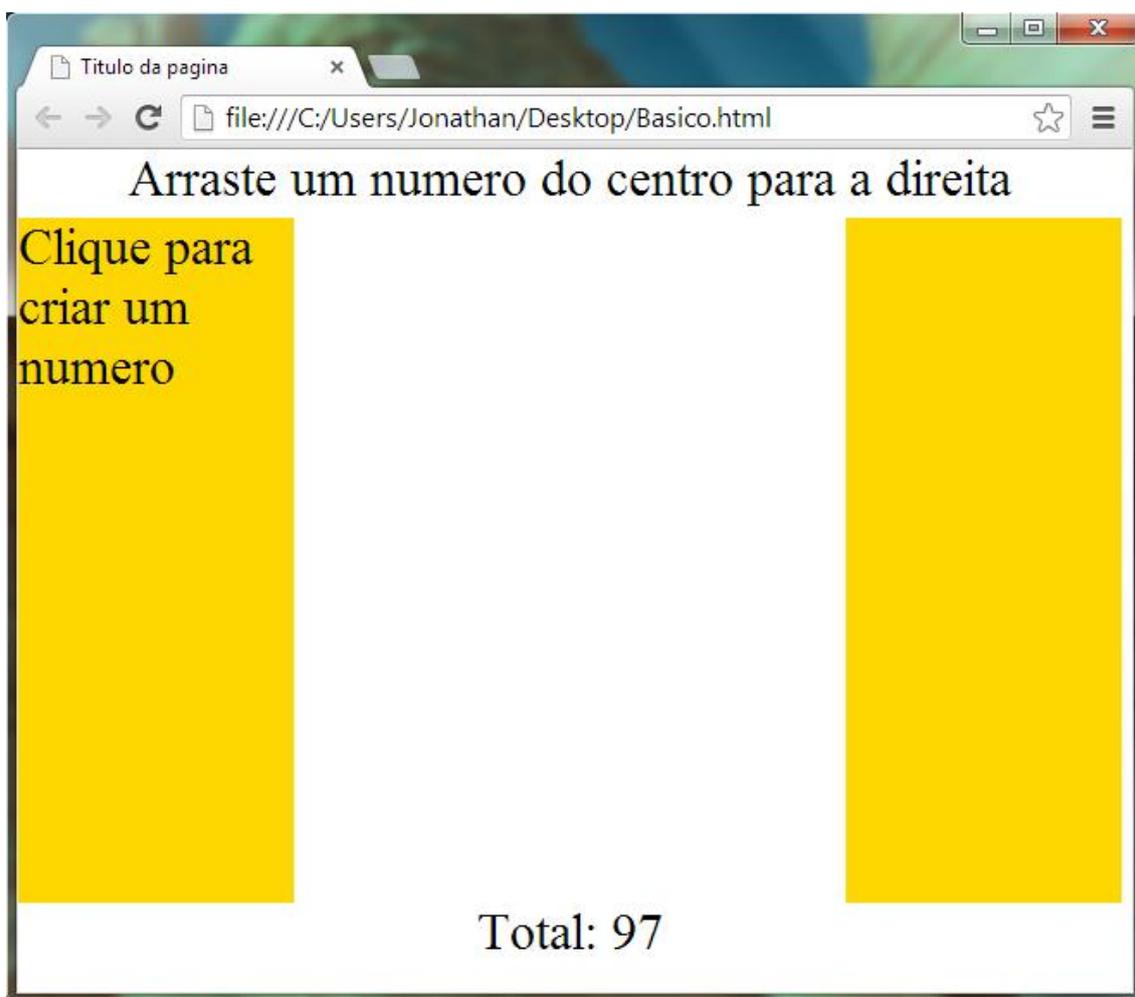
Figura 5.16 - Exemplo *Drag* - Número Gerado



Fonte: O Autor

Ao arrastar o número, o número desaparece e o valor total é atualizado, resultando a tela da figura 5.17.

Figura 5.17 - Exemplo *Drag* - Soltando o Número



Fonte: O Autor

A *tag* `<canvas>` foi apresentada anteriormente nesse projeto. *Canvas* é uma *tag* de HTML 5 que se pode incorporar dentro de um documento HTML com a finalidade de desenhar gráficos com JavaScript. Uma vez que a *Canvas* é um bitmap, cada pixel desenhado nele substitui os pixels já presentes nele na mesma posição.

O Quadro 5.13 mostra o *template* básico para todas as receitas em 2D do HTML5 *Canvas*.

Quadro 5.13 - Básico para a utilização do *Canvas*

```
1 <head>
2   <script>
3     window.onload = function() {
4       var canvas = document.getElementById("myCanvas");
5       var context = canvas.getContext("2d");
6       // colocar o código para desenhar
7     };
8   </script>
9 </head>
10 <body>
11   <canvas id="myCanvas" width="578" height="200">
12   </canvas>
13 </body>
14
```

Fonte: O Autor

Observa-se que o elemento Canvas está inserido dentro do corpo do documento HTML, e está definido como uma *id*, um *width* (comprimento), e um *height* (altura). JavaScript utiliza a *id* para se referenciar à *tag* canvas. Os atributos *width* e *height* são usados para definir o tamanho da área do desenho. Uma vez que o canvas é acessado através da função *getElementById*, então é possível capturar o contexto 2D em uma variável para desenhar nela posteriormente:

```
var context = canvas.getContext("2d");
```

O exemplo contido nos Quadros 5.14, 5.15 e 5.16, apresentado por PORFÍRIO (2012), mostra a utilização da *<canvas>* para desenhar na tela, e como utilizar o teclado para mover um objeto, no caso uma nave.

Quadro 5.14 - Exemplo de utilização de Canvas parte 1

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4  <script>
5      // Global variables
6      var shipX = 0; // X position of ship
7      var shipY = 0; // Y position of ship
8      var canvas; // canvas
9      var ctx; // context
10     var back = new Image(); // storage for new background piece
11     var oldBack = new Image(); // storage for old background piece
12     var ship = new Image(); // ship
13     var shipX = 0; // current ship position X
14     var shipY = 0; // current ship position Y
15     var oldShipX = 0; // old ship position X
16     var oldShipY = 0; // old ship position Y
17     // This function is called on page load.
18     function canvasSpaceGame() {
19         // Get the canvas element.
20         canvas = document.getElementById("myCanvas");
21         // Make sure you got it.
22         if (canvas.getContext)
23             // If you have it, create a canvas user interface element.
24             {
25                 // Specify 2d canvas type.
26                 ctx = canvas.getContext("2d");
27                 // Paint it black.
28                 ctx.fillStyle = "black";
29                 ctx.rect(0, 0, 300, 300);
30                 ctx.fill();
31                 // Save the initial background.
32                 back = ctx.getImageData(0, 0, 30, 30);
33                 // Paint the starfield.
34                 stars();
35                 // Draw space ship.
36                 makeShip();
37             }
38         // Play the game until the until the game is over.
39         gameLoop = setInterval(doGameLoop, 16);
40         // Add keyboard listener.
41         window.addEventListener('keydown', whatKey, true);
42     }
43     // Paint a random starfield.
44     function stars() {
45         // Draw 50 stars.
46         for (i = 0; i <= 50; i++) {
47             // Get random positions for stars.
48             var x = Math.floor(Math.random() * 299);
49             var y = Math.floor(Math.random() * 299);
50             // Make the stars white
51             ctx.fillStyle = "white";
52             // Give the ship some room by painting black stars.
53             if (x < 30 || y < 30) ctx.fillStyle = "black";
54             // Draw an individual star.
55             ctx.beginPath();
56             ctx.arc(x, y, 3, 0, Math.PI * 2, true);
57             ctx.closePath();
58             ctx.fill();
59             // Save black background.
60             oldBack = ctx.getImageData(0, 0, 30, 30);
61         }
62     }

```

Fonte: PORFÍRIO (2012)

Quadro 5.15 - Exemplo de utilização de Canvas parte 2

```

63 function makeShip() {
64     // Draw saucer bottom.
65     ctx.beginPath();
66     ctx.moveTo(28.4, 16.9);
67     ctx.bezierCurveTo(28.4, 19.7, 22.9, 22.0, 16.0, 22.0);
68     ctx.bezierCurveTo(9.1, 22.0, 3.6, 19.7, 3.6, 16.9);
69     ctx.bezierCurveTo(3.6, 14.1, 9.1, 11.8, 16.0, 11.8);
70     ctx.bezierCurveTo(22.9, 11.8, 28.4, 14.1, 28.4, 16.9);
71     ctx.closePath();
72     ctx.fillStyle = "rgb(222, 103, 0)";
73     ctx.fill();
74     // Draw saucer top.
75     ctx.beginPath();
76     ctx.moveTo(22.3, 12.0);
77     ctx.bezierCurveTo(22.3, 13.3, 19.4, 14.3, 15.9, 14.3);
78     ctx.bezierCurveTo(12.4, 14.3, 9.6, 13.3, 9.6, 12.0);
79     ctx.bezierCurveTo(9.6, 10.8, 12.4, 9.7, 15.9, 9.7);
80     ctx.bezierCurveTo(19.4, 9.7, 22.3, 10.8, 22.3, 12.0);
81     ctx.closePath();
82     ctx.fillStyle = "rgb(51, 190, 0)";
83     ctx.fill();
84     // Save ship data.
85     ship = ctx.getImageData(0, 0, 30, 30);
86     // Erase it for now.
87     ctx.putImageData(oldBack, 0, 0);
88 }
89 function doGameLoop() {
90     // Put old background down to erase shipe.
91     ctx.putImageData(oldBack, oldShipX, oldShipY);
92     // Put ship in new position.
93     ctx.putImageData(ship, shipX, shipY);
94 }
95 // Get key press.
96 function whatKey(evt) {
97     // Flag to put variables back if we hit an edge of the board.
98     var flag = 0;
99     // Get where the ship was before key process.
100    oldShipX = shipX;
101    oldShipY = shipY;
102    oldBack = back;
103    switch (evt.keyCode) {
104        // Left arrow.
105        case 37:
106            shipX = shipX - 30;
107            if (shipX < 0) {
108                // If at edge, reset ship position and set flag.
109                shipX = 0;
110                flag = 1;
111            }
112            break;
113        // Right arrow.
114        case 39:
115            shipX = shipX + 30;
116            if (shipX > 270) {
117                // If at edge, reset ship position and set flag.
118                shipX = 270;
119                flag = 1;
120            }
121            break;

```

Fonte: PORFÍRIO (2012)

Quadro 5.16 - Exemplo de utilização de Canvas parte 3

```

122     // Down arrow
123     case 40:
124         shipY = shipY + 30;
125         if (shipY > 270) {
126             // If at edge, reset ship position and set flag.
127             shipY = 270;
128             flag = 1;
129         }
130         break;
131     // Up arrow
132     case 38:
133         shipY = shipY - 30;
134         if (shipY < 0) {
135             // If at edge, reset ship position and set flag.
136             shipY = 0;
137             flag = 1;
138         }
139         break;
140     }
141     // If flag is set, the ship did not move.
142     // Put everything back the way it was.
143     if (flag) {
144         shipX = oldShipX;
145         shipY = oldShipY;
146         back = oldBack;
147     } else {
148         // Otherwise, get background where the ship will go
149         // So you can redraw background when the ship
150         // moves again.
151         back = ctx.getImageData(shipX, shipY, 30, 30);
152     }
153 }
154 </script>
155 </head>
156 <body onload="canvasSpaceGame()" >
157 <h1>
158 Canvas Space Game
159 </h1>
160 <canvas id="myCanvas" width="300" height="300">
161 </canvas>
162 </body>
163 </html>
164

```

Fonte: PORFÍRIO (2012)

Esse exemplo, utiliza de variáveis globais, que poderão ser acessadas por todas as funções. A função *canvasSpaceGame*, chamada assim que a página é carregada, guarda o canvas, através da função *getElementById*, na variável *canvas* e o contexto 2D na variável *ctx*. Então é pintado de preto uma área retangular, de 300 por 300 pixels, e usando *getImageData*, foi salvo na variável *back*, um pequeno pedaço de 30x30 pixels que será usada para sobrepor a imagem da nave quando ela se deslocar. É chamado

então as funções `stars()`, que pintará um número de estrelas em posições aleatórias, e a função `makeShip()`, que desenhará a espaçonave utilizando curvas de *bezier*, salvará a imagem na variável `ship`. A seguir é criado um temporizador que chamará a função `doGameLoop` a cada 16 milissegundos. Esses *loops* são muito usados, pois neles se realizam tarefas como atualizar movimentação, redesenhar, verificar colisão entre outros. No caso do exemplo no *loop* será apagada a nave da posição antiga e desenhada na posição nova, ambos usando a função `putImageData`, que desenha a imagem passada no primeiro parâmetro, nas posições `x` e `y`, passados pelo 2º e 3º parâmetros respectivamente. E por fim é adicionado um *listener* para o evento `keydown`, que chamará a função `whatKey`, a qual atualizará as variáveis criadas para guardar a posição antiga da nave, e a posição nova de acordo com qual tecla foi apertada. Também é atualizada a variável `back` e a `oldBack` [PORFÍRIO 2012]. A figura 5.18 mostra o resultado da execução do exemplo fornecido por PORFÍRIO.

Figura 5.18 - Amostra de Canvas



Fonte: O Autor

5.5 Testes

Existem dois tipos de testes de aceitação que são feitos pelo usuário, como Alfa e Beta, que dificilmente operam o sistema da forma prevista, e visam descobrir erros acumulativos que poderiam deteriorar o sistema no decorrer do tempo. Para que isso não aconteça, o objetivo desses testes é obter um feedback dos clientes ou usuários do sistema antes de entrar em produção ou ser disponibilizado para comercialização [Luz 2012]:

- Teste Alfa – é realizado no ambiente da organização em que o produto foi desenvolvido. Pode ser realizado pelos clientes ou por uma equipe de teste independente, que registram os problemas encontrados no seu uso e este ambiente deve ser controlado, para maior confiabilidade; e

- Teste Beta (ou teste no campo) – Já o teste Beta é realizado em uma ou mais instalações do cliente pelo usuário final do software. O desenvolvedor não deve estar presente, para que não seja controlado por ele. Assim, o teste Beta é uma aplicação real do *software*. Os problemas são registrados pelo usuário e repassados regularmente ao desenvolvedor, que vai corrigindo o software antes de lançar o produto para venda.

5.5.1 Testes com especialistas

Já para realizar uma avaliação de software educacional em suas entregas realizam a cada parte do jogo, diferentes etapas. Segundo SILVA (2002) apud Barbosa Neto (2012).

“Avaliar software educacional não é uma tarefa fácil. As diferentes modalidades existentes, tais como, exercício e prática, tutoriais, simulações, jogos, hipertexto/hipermídia, e sistemas baseados em conhecimento (sistemas especialistas e tutores inteligentes), apresentam características diferentes, sendo necessária a elaboração de critérios de avaliação específicos para cada tipo (Silva, C. M. T.; Avaliação de Software Educacional. Revista On Line Conect, Internet, v. 1, n.4, p. 1-5, 2002)”.

Essa avaliação de um software educacional, do jogo foi dividida em duas partes. Pois, segundo SILVA & ELLIOT (1998) apud Barbosa Neto (2012).

“O fato de um sistema de hipermídia para uso educacional ser bastante diferente de um programa de computador tradicional e, portanto, demandar novos critérios para sua avaliação, gere algumas

recomendações. As abordagens indicadas para o julgamento desses sistemas, assim como não se recomenda apenas a avaliação realizada por especialistas. Além disso, é aconselhável estabelecer a relação entre a nota dada ao programa ou o julgamento feito pelos especialistas e os dados reais quanto à eficácia e impacto deste no ambiente educacional (Silva, C. M. T., Elliot, L. G., Avaliação de Software Educacional Hipermedia: A Contribuição de Especialistas e Usuários. Ensaio. Avaliação e Políticas Públicas em Educação, Rio de Janeiro, v. 5, n.16, p. 299-311, 1998).”

5.6 Considerações Finais

Esse capítulo apresentou os passos para desenvolver um jogo, desde o estágio do design até a etapa de teste. Na etapa do design foi mostrado o processo para um jogo e o design instrucional. Na etapa do desenvolvimento foi mostrado o básico para a programação com HTML, CSS e JavaScript. Na etapa de testes foram mostrados diferentes testes que devem ser realizados antes do produto ser considerado pronto para o mercado.

6 Considerações Finais

6.1 Contribuições

Este trabalho mostrou a importância que os jogos possuem na educação, descrevendo como os jogos foram utilizados nesse aspecto com o passar da história. Foi destacado um exemplo de jogo, o Browserquest, que mostrou o potencial que HTML 5 tem para o desenvolvimento de jogos. O principal objetivo do trabalho foi apresentar um *guideline* com o intuito de auxiliar no desenvolvimento de jogos para web com HTML 5, com *design* voltado a educação.

O trabalho focou bastante no básico para a utilização de HTML e CSS, com alguns exemplos da utilização de suas *tags*, o que auxilia no desenvolvimento de jogos mais simples. Introduziu também o uso de JavaScript neste contexto e forneceu alguns exemplos de como é utilizado para criar a interação com o usuário.

6.2 Limitações

O *guideline* proposto não aprofunda tanto na parte do design de um jogo, particularmente um jogo educativo, faltando o uso de exemplos e de técnicas que são usadas pelos designers no processo da criação. Esse fato se dá ao pouco conhecimento que o autor possui com relação ao assunto.

Também não foi focado no desenvolvimento de jogos mais complexos. Os exemplos mostrados no *guideline* são simples comparados à produção de jogos. A utilização de um maior número de exemplos, sendo eles jogos educativos, ou partes deles, beneficiaria mais no aprendizado do leitor com relação ao assunto.

6.3 Trabalhos futuros

Para trabalhos e estudos futuros, poderá ser aprofundado mais na etapa de *design*, fornecendo exemplos de suas etapas, além de técnicas e ferramentas que são utilizadas para auxiliar no processo da criação da ideia, e avaliar os mesmos. Seria interessante o acompanhamento de um especialista em design de jogos, ou mesmo um número maior de referências no assunto.

Também é proposta como trabalho futuro uma avaliação de jogos educativos já presentes no mercado, mostrando os pontos fortes e fracos de seus designs e o que

poderia ser melhorado de modo a atrair mais o público em cada um. Portanto, fica proposto para trabalhos futuros incrementar o *guideline* para abranger jogos mais complexos, com tratamento de Física, colisão, utilização de inteligência artificial, comunicação, entre outros.

Fica proposta também a apresentação de *engines*, com bibliotecas mais completas, que facilitam muito o desenvolvimento dos jogos. A utilização de *engines* torna o desenvolvimento de jogos mais complexos, uma tarefa bem mais simples.

E por fim, fica proposta a validação do *guideline* com usuários finais, mostrando-os jogos que foram desenvolvidos seguindo o *guideline*, para que através do seu *feedback*, ser capaz de identificar os pontos fortes e fracos do trabalho.

Referências Bibliográficas

ARIÈS, P.; **História social da criança e da família**. 2ed. Rio de Janeiro: LTC, 1981.

ANDRADE, D.A., MARCH, K.R.C.; **HTML5 – Novos padrões para Web Semântica**

Disponível em:

<http://antigo.unipar.br/~seinpar/guia/arquivos/artigos/DiogoAraujoDeAndrade.pdf>

Acessado em: Julho de 2013.

BARBOSA NETO, J. F.; **Uma metodologia de desenvolvimento de jogos educativos em dispositivos móveis para ambientes virtuais de ensino**. Disponível em:

<http://www.scribd.com/doc/117719219/Jose-Francisco-Barbosa-Neto-Uma-Metodologia-de-Desenvolvimento-de-Jogos-Educativos-em-Dispositivos-Moveis-para-Ambientes-Virtuais-de-Ensino-2012-Di> . Acessado em: Julho de 2013.

BELLIS, M.; **The History of HTML, About.com Guide**. Disponível em:

<http://inventors.about.com/od/computersoftware/a/html.htm>. Acessado em: Julho de 2013.

BOTELHO, L.; **Jogos educacionais aplicados ao e-learning**. Disponível em:

http://www.ead.sp.senac.br/portal/news_artigos_show.asp?cod=290&cod_sis=7&cod_cat=14. Acessado em: Julho de 2013.

Browserquest. Disponível em: <http://browserquest.com.br/> Acessado em: Setembro de 2013

CAMPOS, J.A.S.; **Simulações e Jogos Educacionais**. Disponível em:

http://www.nce.ufrj.br/ginape/iga502/Material_aulas/Atividade%20Jogos%20Educacionais.doc. Acessado em Julho de 2013.

CASTRO, E.; **Considerações Históricas dos Jogos no âmbito educacional.**

Disponível em: <http://meuartigo.brasilecola.com/educacao/consideracoes-historicas-dos-jogos-no-ambito-educacional.htm>. Acessado em: Julho de 2013.

CHEN, G.; **WebGL Massively Multiplayer Online Game.** Disponível em:

<https://www.seas.upenn.edu/~cis497/projects2013/GianniChenProposal.pdf> . Acessado em: Setembro de 2013.

FABIANO, M.; **Criação de jogos game maker - férias.** Disponível em:

<http://www.slideshare.net/michelfabiano/criao-de-jogos-game-maker-frias> Acessado em: Setembro de 2013

FALKEMBACH, G. A. M.; **Concepção e Desenvolvimento de material educativo digital.** RNOTE - Revista Novas Tecnologias na Educação, UFRGS/POA, v. 3, 01 mar. 2005.

FALKEMBACH, G. A. M, GELLER, M., SILVEIRA, S. R.; **Desenvolvimento de Jogos Educativos Digitais utilizando a Ferramenta de Autoria Multimídia:** um estudo de caso com o Tool Book Instructor. 2006. Disponível em:

<http://seer.ufrgs.br/renote/article/view/13874/7794> Acessado em Julho de 2013.

FONTOURA, A.M, PEREIRA, A. T. C.; **A criança e o design - aprender brincando**
The child and the design - to learn playing. Publication Name: avaad.ufsc.br

Disponível em: <http://www.avaad.ufsc.br/moodle/prelogin/publicarartigos/323.pdf> .

Acessado em: Setembro de 2013.

G1; **Os profissionais de games** Disponível em: <http://g1.globo.com/Noticias/0,,IIF721-5604,00.html> Acessado em: Setembro de 2013.

GONÇALVES, C.; **Brincar, o despertar psicomotor.** Rio de Janeiro, Sprint: 1996.

Disponível em:

<http://www.avm.edu.br/monopdf/6/VER%C3%94NICA%20GON%C3%87ALVES%20DA%20SILVA.pdf> Acessado em: Julho de 2013.

HAGUENAUER, C. J., CARVALHO, F. S., VICTORINO, A. L. Q., LOPES, CORDEIRO FILHO, F.; **Uso de Jogos na Educação Online: a Experiência do LATEC/UFRJ**. Revista - Volume 1- no 1- Janeiro/Abril de 2007 Disponível em: <http://www.latec.ufrj.br/revistas/index.php?journal=educaonline&page=article&op=view&path%5B%5D=143> Acessado em: Julho de 2013.

HALES, W.; **HTML5 and JavaScript Web Apps**. First Edition, Published by O'Reilly Media, Inc., 2012.

HARRIS, A.; **HTML5 Game Development for Dummies**, Published by John Wiley & Sons, Inc., 2013

History of Ecma Disponível em:

<http://www.ecma-international.org/memento/history.htm> Acessado em: Setembro de 2013

HTML 4 Changes Disponível em:

<http://www.w3.org/TR/html401/appendix/changes.html> Acessado em: Setembro de 2013

KISHIMOTO, Tizuko Morchida (org.). **Jogo, brinquedo, brincadeira e a educação**. 3ª edição, SP: Cortez, 1999.

LAWSON, L., SHARP, R.; **Introdução ao HTML5**. 1ª Edição, Rio Janeiro: Editora Alta Books, 2011.

LEBOVICI, S.; DIATKINE, R.; **Significado e Função do Brinquedo na Criança**. Porto Alegre: Artes Médicas, 1985.

LEWINSKI, S. M.; **O desempenho de equipes em jogos empresariais: um estudo sobre a coesão e maturidade de equipes.** Disponível em:

<http://www.pg.utfpr.edu.br/dirppg/ppgep/dissertacoes/arquivos/180/Dissertacao.pdf>

Acessado em: Agosto de 2013.

Little Workshop – Browserquest. Disponível em:

<http://www.littleworkshop.fr/browserquest.html> Acessado em: Setembro de 2013.

LUCENA, D. A., OLIVEIRA, A.C.C.; **"HTML5: novidades e contribuições."**

Disponível em: <http://sites.setrem.com.br/stin/2012/anais/Daniel.pdf> Acessado em Julho de 2013.

LUZ, L. C. S.; **Teste de Aceitação: problemas, desafios e abordagens.** 2012.

Disponível em: <http://www.slideshare.net/synergiaufmg/teste-de-aceitao-problemasdesafios-e-abordagens>. Acessado em: Setembro 2013

MACHADO, M. P., SOUZA, S. F.; **Métricas e Qualidade de Software.** 2011.

Disponível em: <http://soterio.files.wordpress.com/2011/06/artigoqualidadesw.pdf>

Acessado em Setembro de 2013.

MARCELO, A., PESCUITE, J.; **Design de jogos – Fundamentos.** Rio de Janeiro, Brasport, 2009.

MENDES, T. G.; **GAMES E EDUCAÇÃO: Diretrizes de Projeto para Jogos Digitais Voltados á Aprendizagem.** 2012, Disponível em:

<http://www.lume.ufrgs.br/bitstream/handle/10183/61009/000860539.pdf?sequence=1>

Acessado em Setembro de 2013.

MESQUITA, T. F.; **A Importância da Ludicidade e dos Jogos para o Ensino da Matemática na Educação Infantil.** Disponível em:

http://www.ledum.ufc.br/arquivos/produtos/tcc/TCC_Tarciana.pdf Acessado em: Julho de 2013.

NBR ISO/IEC 9126-1; **Engenharia de software - Qualidade de produto Parte 1: Modelo de qualidade**. Disponível em: [http://luizcamargo.com.br/arquivos/NBR%20ISO IEC%209126-1.pdf](http://luizcamargo.com.br/arquivos/NBR%20ISO%20IEC%209126-1.pdf) Acessado em: Setembro de 2013.

NEGRINE, Airton. **Aprendizagem e desenvolvimento infantil**. Porto Alegre: PRODIL, 1994.

NOTEPAD++; **Notepad++ Home**. Disponível em: <http://notepad-plus-plus.org/> Acessado em: Setembro de 2013.

NGUYEN, B. **Linux Dictionary V 0.16**. Disponível em: <http://www.tldp.org/LDP/Linux-Dictionary/html/index.html> Acessado em: Agosto de 2013.

PACÍFICO, J. S., BARBOSA, R. A.; **Modelagem de Jogos Eletrônicos Educativos e Divertidos**. 2011. Disponível em: <http://www.slideshare.net/chichiab/modelagem-dejogos-eletrnicos-educativos-e-divertidos>. Acessado em: Setembro 2013

PORFÍRIO, A.; **O que é HTML 5?** 2012. Disponível em: <http://www.alexandreporfirio.com/2012/04/12/artigos/o-que-e-html-5/> Acessado em: Julho de 2013

PORFÍRIO, A.; **HTML5 – Movimentando um objeto na tela**. 2012. Disponível em: <http://www.alexandreporfirio.com/2012/04/13/games/html5-movimentando-u-objeto/> Acessado em: Setembro de 2013.

ROSA, A. R.; AZUAYA, A.C. A.; **Jogos de Empresa na Educação Superior no Brasil: Perspectiva para 2010**. In: Encontro Anual da Associação dos Programas de Pós-graduação em Administração, 30, 2006, Salvador, EPQA312, CD-ROM. Disponível em: <http://www.anpad.org.br/enanpad/2006/dwn/enanpad2006-epqa-0312.pdf> Acessado em: Agosto de 2013.

Rowell, E.; **HTML5 Canvas Cookbook**. www.packtpub.com, Published by Packt Publishing Ltd., 2011

PAUL, R.; **ArsTechnica** (2012) <http://www.littleworkshop.fr/browserquest.html>

SOARES, J. M.; **A Importância do Lúdico na Alfabetização Infantil**. Disponível em: <http://www.planetaeducacao.com.br/portal/imagens/artigos/diario/ARTIGO%20JIANE%20JOGO1.pdf>. Acessado em: Julho de 2013.

SULZBACH, L.; **A importância das atividades lúdicas no processo de alfabetização e letramento dos educandos do 1º ano do ensino fundamental**. Disponível em: <http://www.ijui.com/artigos/48244-a-importancia-das-atividades-ludicas-no-processo-de-alfabetizacao-e-letramento-dos-educandos-do-1-ano-do-ensino-fundamental-por-loriane-sulzbach.html#sthash.8c38VbYY.dpuf>. Acessado em: Julho de 2013.

SYLVESTER, T.; **Designing Games**. Published by O'Reilly Media, Inc., 2013.

TAROUCO, L. M. R., ROLAND, L. C., FABRE, M. C. J. M. e KONRATH, M. L. P.; **Jogos educacionais**. 2004. Disponível em: <http://www.cinted.ufrgs.br/ciclo3/af/30-jogoseducacionais.pdf> Acessado em: Julho de 2013.

TEIXEIRA, C.C. P.; **“AS EXPRESSÕES ARTÍSTICAS NO 1.º E 2.º CICLOS DO ENSINO BÁSICO – UMA ABORDAGEM INTERDISCIPLINAR”** Disponível em: http://repositorio.utad.pt/bitstream/10348/2401/1/msc_ccpteixeira.pdf Acessado em: Julho de 2013.

VALENTINE, T., REID, J.; **JavaScript Programmer's Reference**, Springer Science+Business. Media New York, www.apress.com. 2013

VASCONCELLOS, T.; **Jogos e Brincadeiras no Contexto Escolar**. Disponível em <http://www.tvbrasil.org.br/fotos/salto/series/165801Jogos.pdf> Acessado em: Setembro de 2013

VEER, E. V.; **JavaScript For Dummies**. 4th Edition, Published by Wiley Publishing, Inc., Indianapolis, Indiana, 2005.

VILELA, J. F.F.; **Projeto e Implementação de um Software Educativo Multi Agente com Tropos e JADE**. 2011. Disponível em:
http://www.univasf.edu.br/~ccomp/monografias/monografia_6.pdf

Acessado em: Julho de 2013.

W3SCHOOLS; W3Schools Online Web Tutorials, Disponível em:
<http://www.w3schools.com/> Acessado em Setembro de 2013.

WAJSKOP, G.; **O brincar na educação infantil**. Cadernos de Pesquisa, n° 92, pp 62-69. 1995. Disponível em
<http://www.fcc.org.br/pesquisa/publicacoes/cp/arquivos/742.pdf> Acessado em: Julho 2013.

WEBSOCKET.ORG; **About HTML5 WebSockets**. Disponível em:
<http://www.websocket.org/aboutwebsocket.html> Acessado em Setembro de 2013.