INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA
E TECNOLOGIA DO RIO GRANDE DO SUL
CAMPUS BENTO GONÇALVES

LIMITAÇÕES DO HTML5 NO DESENVOLVIMENTO DE JOGOS MULTIPLATAFORMA

JEAN CARLO MACHADO

Bento Gonçalves, Dezembro 2015

JEAN CARLO MACHADO

LIMITAÇÕES DO HTML5 NO DESENVOLVIMENTO DE JOGOS MULTIPLATAFORMA

Monografia apresentada junto ao Curso de Tecnologia em Análise e Desenvolvimento de Sistemas no Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul - Campus Bento Gonçalves, como requisito parcial à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.
Orientador: Mr. Rafael Jaques

Bento Gonçalves, Dezembro 2015

RESUMO

LLorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

orem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Palavras-chave: HTML5, Limitações, Jogos, Multiplataforma

ABSTRACT

LLorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

orem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Palavras-chave: HTML5, Limitações, Jogos, Multiplataforma

LISTA DE FIGURAS

4.1 Exemplo] de documento HTML 11
4.2 Suíte HTML 12
4.3 Exemplo de utilização de seletores do DOM em JavaScript 13
4.4 Exemplo de Folha de Estilo 14
4.5 Os módulos do CSS 15
4.6 * 15
4.7 Exemplo de Media Query 16
4.8 Exemplo de media queries customizados 16
4.9 * 16
4.10 Exemplo de transição 17
4.11 Exemplo de transformação 17
4.12 Propostas do ECMA 7 21
4.13 * 21
4.14 Exemplo de utilização de WEB Animatios 23
4.15 * 23
4.16 Suporte das especificações do HTML nos navegadores 25
4.17 Círculo em SVG. 27
4.18 Comparação de codecs de áudio 32
4.19 Web Storage na prática 35
4.20 * 35
4.21 Adicionando um cliente em IndexedDB. 36

6.1 Diagrama de classes simplificado 42
6.2 Tabuleiro do jogo 44
6.3 Placar do jogo 45
6.4 Configurações do jogo 46
6.5 Exemplo de utilização de funções imediatamente invocadas 48

F.1 Diagrama de classes completo 63

SUMÁRIO

1 CONTEXTUALIZAÇÃO 1
1.1 JOGOS 1
1.2 Limitações 1
1.3 ESTE TRABALHO 2
2 PROBLEMA. 3
2.1 OBJETIVOS 3
2.1.1 OBJETIVO GERAL 3
2.1.2 OBJETIVOS ESPECÍFICOS 3
3 JUSTIFICATIVA 5
4 REVISÃO BIBLIOGRÁFICA 6
4.1 JOGOS 6
4.1.1 MECÂNICA 7
4.2 JOGOS MULTIPLATAFORMA 7
4.2.1 JOGOS WEB 8
4.2.2 JOGOS HÍBRIDOS 8
4.2.3 DESENVOLVIMENTO DE JOGOS NATIVOS 9
4.3 WEB 9
4.3.1 OPEN WEB 9
4.4 HTML 10
4.4.1 DOM 13
4.5 CSS 13
4.5.1 Media Queries 15
4.5.2 Transições 16
4.5.3 Transformações 3D 17
4.5.4 CSS 4 18
4.6 JAVASCRIPT 18
4.6.1 JAVASCRIPT 7 20
4.6.2 ASM.JS 20
4.6.3 WEB Assembly 22
4.6.4 Web Animations 23
4.7 NAVEGADORES 23
4.8 ANDROID 24

4.9 DETECÇÃO DE RECURSOS 26
4.10 Pré Carregamento 26
4.11 RENDERIZAÇÃO 26
4.11.1 SVG 27
4.11.2 CANVAS 27
4.11.3 WEBGL 28
4.11.4 Shaders 29
4.11.5 Bibliotecas OpenGL 29
4.11.6 Otimizações 29
4.11.7 WEBVR 29
4.12 WebCL 30
4.13 CODECS 30
4.14 AUDIO 30
4.14.1 TAG AUDIO 31
4.14.2 API DE AUDIO 31
4.15 VIDEO 31
4.16 ARMAZENAMENTO 33
4.16.1 WEB SQL 33
4.16.2 WEB STORAGE 34
4.16.3 IndexedDB 35
4.17 WEB WORKERS 36
4.18 OFFLINE 36
4.18.1 Aplicações offline 37
4.19 ENTRADA DE COMANDOS 37
4.19.1 Acelerômetro 37
4.20 HTTP/2 37
4.21 DEBUG 37
4.21.1 Source Maps 37
4.21.2 Debug 3D 38
4.22 DISPONIBILIZAÇÃO DA APLICAÇÃO 38
4.22.1 INSTALAÇÃO 38
4.23 TRABALHOS SIMILARES 38
5 METODOLOGIA 39
6 PROJETO. 40
6.1 MECÂNICA 40
6.2 Requisitos 40
6.2.1 Requisitos funcionais 40
6.2.2 Requisitos não funcionais 41
6.3 Modelagem 41

6.4 Desenvolvimento 41
6.5 Laço do jogo 41
6.6 Otimizações para jogos 43
6.6.1 CSS 43
6.6.2 JavaScript 47
7 RESULTADOS 49
7.1 LIMITAÇÕES 49
7.1.1 VERSÕES 49
7.1.2 OFFLINE 50
7.1.3 SVG 50
7.1.4 Canvas 50
7.1.5 WebGL 50
7.1.6 VIDEO 50
7.1.7 ASSETS 50
7.1.8 Codecs 51
7.1.9 ÁUDIO 51
7.1.10 INTERFACE GRÁFICA 51
7.1.11 PERFORMANCE 51
7.1.12 Acelerômetro 52
7.1.13 IMPLEMENTAÇÃO INCONSISTENTE DE APIs 52
7.1.14 TAMANHO DE TELA 52
7.1.15 JavaScript 52
7.1.16 Detecção de recursos 52
8 CONCLUSÕES 53
8.0.1 TRABALHOS FUTUROS 53
REFERÊNCIAS BIBLIOGRÁFICAS 54
A CONVERSORES PARA HTML5 56
A.0.1 Bibliotecas relevantes do desenvolvimento de jogos 56
A.1 Frameworks de jogos 56
A.2 CROSSWALK 57
A.3 PHONEGAP 57
A.4 PHONEGAP CLOUD 57
A.5 NODEJS 57
B ALTERNATIVAS AO JAVASCRIPT 58
B.1 TYPESCRIPT 58
B.2 DART 58

C SISTEMAS DE BUILDING 59
D AMBIENTES PARA DESENVOLVIMENTO HTML5 60
E METODOLOGIA DE DESENVOLVIMENTO DE SOFTWARE PARA A CONSTRUÇÃO DE GAMES 61
F Diagrama de classes 62

1 CONTEXTUALIZAÇÃO

1.1 JOGOS

Desenvolvedores de jogos web podem rapidamente satisfazer as necessidades de seus jogadores, mantendo-os leais a tecnologia HTML5 (Zhang 2012).

Mais de 80% do tempo total gasto utilizando dispositivos móveis é na utilização de aplicativos, e 32% deste tempo é para jogar vídeo games (Janiszewswi 2014).

A maioria dos desenvolvedores demonstra interesse para o HTML5. (referenciar) E muito pouco investimento é necessário para começar a desenvolver jogos utilizando as tecnologias da WEB (Kuryanovich et al. 2014).

O tempo de desenvolvimento de uma aplicação em HTML5 é 67% menor (referenciar) que aplicações nativas. Isso mostra o custo efetivo de aplicações baseadas em HTML5.

A real vantagem de aplicações em HTML5 é o suporte horizontal entre as plataformas - que é a maior razão por trás do custo efetivo («Smart Phones Application development using HTML5 and related technologies: A tradeoff between cost and quality»).

A maior dificuldade em capturar uma base de usuários é que o mercado de dispositivos móveis é muito fragmentado e não existe uma única plataforma popular.(«Smart Phones Application development using HTML5 and related technologies: A tradeoff between cost and quality»).

Segundo Janiszewswi 2014: 80% do tempo total gasto usando dispositivos móveis é para a utilização de aplicativos, e 32% é para jogar vídeo games.

Funcionalidades foram disponibilizadas de diversas fontes e não foram construídas de forma consistente com as demais. Além disso, devida a única característica da Web, erros de implementação se tornam frequentes, e muitas vezes se tornam o padrão, pois outras funcionalidades dependem destas primeiras antes que elas estejam estáveis. (W3C manual)

Os desenvolvedores de navegadores podem interpretar/implementar as especificações erroneamente aumentando os problemas de compatibilidade.

1.2 Limitações

A utilização da WEB é gigantesca, mas criar jogos dinâmicos e de tempo real não é o primeiro objetivo da WEB (Kuryanovich et al. 2014).

1.3 ESTE TRABALHO

Este projeto propõe analisar as limitações do HTML5 quanto relativo a construção de jogos multiplataforma. Através de revisão bibliográfica e da criação de um protótipo de jogo multiplataforma.

2 PROBLEMA

A carência de definições concretas sobre a viabilidade da atual versão do HTML quando aplicado ao desenvolvimento de jogos, e o senso comum, acostumado com soluções nativas, acabam por praticamente monopolizar à construção de jogos nativos.

2.1 OBJETIVOS

2.1.1 OBJETIVO GERAL

Identificar possíveis limitações no processo de desenvolvimento de jogos multiplataforma oriundas do atual estado de definição e implementação do HTML5.

Não é objetivo deste trabalho demonstrar os pontos fortes do HTML5, apenas suas limitações. Também não é objetivo deste trabalho comparar o HTML com outras tecnologias de desenvolvimento de jogos, como Flash Player, Silverlight ou alternativas Desktop.

2.1.2 OBJETIVOS ESPECÍFICOS

Este trabalho tem por objetivo estudar as limitações de desenvolvimento de jogos nas plataformas Android e navegadores Desktop Google Chrome e Firefox em suas últimas versões. Optamos por Android, e não IOS, pois o primeiro contém a vasta maioria do mercado de dispositivos inteligentes, e por termos maior experiência na já mencionada plataforma.

A fim de refinar nossa perspectiva com uma experiência prática, também é objetivo deste trabalho construir um protótipo de um jogo.

Vahatupa 2014 cita que vídeo, áudio, drag-and-drop, funcionalidades de gráficos e armazenamento offline são aspectos importantes do HTML5 para o desenvolvimento de jogos. Atém dos elementos já citados, julgamos relevante estudar os seguintes aspectos e tecnologias relacionadas aos jogos:

- HTML e tecnologias relacionadas (CSS, JavaScript)
- Tecnologias de renderização (WebGL, Canvas, SVG)
- Empacotadores HTML5
- Eventos de entrada
- Armazenamento

- Disponibilização de assets (controle de tamanhos, cache, etc)
- Aplicações offline

3 JUSTIFICATIVA

Tendo em vista que este trabalho busca mapear possíveis problemas do desenvolvimento multiplataforma em HTML ele serve para apoiar e justificar decisões relativas ao desenvolvimento de jogos multiplataforma.

Muitas pessoas no mercado de software são da opinião que o desenvolvimento nativo é a melhor escolha para o desenvolvimento de jogos. Este trabalho, por indicar problemas concretos do desenvolvimento WEB, serve para verificar essa opinião.

Grande parte dos desenvolvedores estão familiarizados com as tecnologias da WEB ou apontam interesse na tecnologia. A revisão tecnológica que este trabalho contém pode servir como um indicativo de sobre o que é preciso estudar para começar a desenvolver jogos na WEB.

4 REVISÃO BIBLIOGRÁFICA

4.1 JOGOS

Segundo (Lemes 2009) jogo digital constitui-se em uma atividade lúdica composta por uma série de ações e decisões, limitada por regras e pelo universo do game, que resultam em uma condição final.

Apesar da definição clara, uma taxonomia dos jogos não é tarefa trivial. Pela diversidade em itens e gêneros e a vasta quantidade de dimensões que os vídeo games se encontram, uma categorização dos jogos contemporâneos é extremamente difícil de desenvolver (muitos já tentaram) (Granic, Lobel e Engels 2014, pp. 60).

Para corroborar com essa complexidade Granic, Lobel e Engels 2014 afirmam que a natureza dos jogos tem mudado drasticamente na última década, se tornando cada vez mais complexos, diversos, realísticos e sociais em sua natureza.

Com as inovações nas tecnologias relacionadas ao HTML novas possibilidades e gêneros de jogos podem ser explorados na WEB. Sendo que cada gênero acompanha um conjunto de desafios específicos. Jogos de FPS (tiro em primeira pessoa) requerem menor latência de rede, já jogos de RPG podem requerer vastas quantidades de cache (Kuryanovich et al. 2014).

A inerente complexidade dos jogos os tornam tópico de interesse para pesquisas. Apesar de a quantidade de estudos sobre os malefícios dos jogos ser muito maior do que os estudos sobre seus benefícios, a quantidade de benefícios já correlacionados aos jogos é grande.

(Granic, Lobel e Engels 2014) demostra que vídeo games melhoram as funções cognitivas, as capacidades criativas, e motivam uma visão positiva diante a falha. Também segundo (Granic, Lobel e Engels 2014) postura positiva em relação a falha correlaciona-se com melhor performance acadêmica.

Benefícios em habilidades práticas também são observados em usuários de jogos. Jogadores de jogos de tiro demonstram maior alocação de atenção, maior resolução espacial no processamento visual e melhor habilidades de rotação (Granic, Lobel e Engels 2014).

Estes aspectos positivos muitas vezes não recebem a atenção devida. Habilidades espaciais derivadas de jogar jogos de tiro comercialmente disponíveis são comparáveis aos efeitos de um curso universitário que busca melhorar as mesmas habilidades (Granic, Lobel e Engels 2014). Estas habilidades derivadas dos jogos são centrais para muitas áreas de interesse humano, Granic, Lobel e Engels 2014 afirmam que habilidades espaciais estão diretamente relacionadas com o sucesso em ciência, tecnologia, engenharia e matemática.

Outra outro ponto importar dos jogos é seu aspecto social. Apesar de a mídia ter criado uma perspectiva negativa sobre jogos, especialmente os violentos, a realidade é mais complexa do que se pensa. Jogadores de jogos violentos, cuja jogabilidade encoraje a cooperatividade, são mais prováveis de exibir comportamento altruísta fora do contexto dos jogos, do que jogadores de jogos não violentos (Granic, Lobel e Engels 2014).

Kuryanovich et al. 2014 ressalta a importância do planejamento antes do desenvolvimento de jogos. Ao criar jogos deve-se planejar o que se pretende atingir e como chegar lá antes de se escrever qualquer código.

4.1.1 MECÂNICA

A mecânica é composta pelas regras do jogo. Quais as ações disponíveis aos usuários, é fortemente influenciada pela categoria do jogo em questão. Dedicar-se na elaboração de uma mecânica é tarefa quintessencial para a construção de um jogo de sucesso.

(Kuryanovich et al. 2014) afirma que se os gráficos e áudio são espetaculares mas a jogabilidade é chata o jogador vai parar de jogar. A substância do jogo é sua mecânica, então não invista muito em visual ao menos que isso desempenhe um papel essencial no jogo.

A elaboração da mecânica em jogos desenvolvidos profissionalmente pode ser integrada dentro de um processo de engenharia de software. O sumário também conta com um resumo sobre metodologias de desenvolvimento de software contextualizada para a criação jogos.

4.2 JOGOS MULTIPLATAFORMA

Jogos em plataformas móveis trazem um novo conjunto de desafios para produtores de jogos. Um destes desafios é fornecer feedback suficiente para o jogador pois muitas vezes o dispositivo é limitado em proporções, som, tela etc.

A interface tem que ser o mais intuitiva o possível. No caso de dispositivos móveis, quanto menos gestos necessários melhor. Tornar previsível causa e efeito é uma boa característica para esta classe de jogos.

Os desenvolvedores tem que evitar fazer o jogo para eles mesmos. E pela falta de crítica os designs tendem a ser ruins. Afinal o que os jogadores querem?

Lemes 2009 aponta alguns fatores procurados pelos usuários de jogos: Desafio, socializar, experiência solitária, respeito e fantasia.

Designers de jogos tem as seguintes possibilidades arquiteturais quando em face de desenvolver um novo jogo: Criar um jogo web, um jogo híbrido, ou nativo. As opções serão descritas abaixo.

4.2.1 JOGOS WEB

Um jogo web é um jogo que utiliza o HTML e ferramentas correlacionadas para sua construção. Este tipo de jogo é o que será abordado neste trabalho.

Jogos da WEB podem se beneficiar da estrutura social da internet.

Não há muito que os títulos de jogos da web residiam em jogos como *Traviam*, desprovidos de animações. Compostos basicamente por formulários, imagens e textos. Não obstante, publicar jogos baseados em texto é uma atividade cada vez mais rara, podendo-se concluir que interface gráfica se tornou uma funcionalidade mandatória (Vahatupa 2014).

Vahatupa 2014 afirma que:

> Estudos sugerem que flexibilidade, em essência facilidade de entrada e saída, é uma das duas razões principais da utilização de jogos de navegadores. A outra razão primária é o fator social envolvido no jogo.

Entre seus pontos positivos pode-se listar:

- Necessitam de uma única base de código e pode rodar em todas as plataformas;
- Contém a mais vasta gama de desenvolvedores e muitos interessados em aprendê-la;
- Seus custos são inferiores, aos do desenvolvimento nativo devido a inexistência de duplicação da base de código;
- Não requerem instalação ou atualizações manuais
- Sua distribuição é superior ao estilo convencional de aplicações desktop (Vahatupa 2014)

Os pontos negativos dessa abordagem são o principal foco deste trabalho.
Mas a um nível macroscópico podemos citar:

- Programas que rodam na web são geralmente lentos se comparados aos nativos;
- Por falta de especificação ou incompletude de implementação.
- Nem todos os recursos disponíveis através das SDK's nativas estão presentes através do HTML5.

Além dos jogos web, há a possibilidade de criar jogos híbridos e nativos.

4.2.2 JOGOS HÍBRIDOS

Jogos híbridos são jogos geralmente desenvolvidos com tecnologias da web que rodam nativamente.

4.2.3 DESENVOLVIMENTO DE JOGOS NATIVOS

Pontos fortes:

- Habilita a melhor experiência de usuário pois permite utilizar ao máximo os recursos e funcionalidades dos dispositivos.

Pontos fracos:

- Porém, devido a cada plataforma conter seu próprio sistema operacional, com seus próprios *SDK's* totalmente incompatíveis, os desenvolvedores são forçados a desenvolver uma versão do jogo para cada plataforma alvo.
- Requer mais pessoas, e maior custo com possivelmente parte do mercado não atendido de qualquer forma.

4.3 WEB

4.3.1 OPEN WEB

A OWP (*Open Web Platform*) é uma coleção de tecnologias livres, amplamente utilizadas e padronizadas. Quando uma tecnologia se torna amplamente popular, através da adoção de grandes empresas e desenvolvedores, ela se torna candidata a adoção pela OWP.

A Open Web, mais que uma coleção de tecnologias, é um conjunto de filosofias as quais a WEB se fundamenta. Entre outras, a Open Web busca transparência, imparcialidade nos processos de criação e padronização de novas tecnologias. Retro compatibilidade com as especificações anteriores. Consenso entre o mercado e o meio acadêmico, nunca um distanciando-se muito do outro.

Várias pessoas, empresas e comunidades estão interessadas neste processo. A W3C é uma comunidade responsável por boa parte das especificações da web como: HTML (em conjunto com a WHATWG), CSS, entre outras[1]. Outros grupos detém responsabilidade por outras tecnologias da OWP, como a ECMA, responsável pelo JavaScript; ou Kronos, responsável pelo WebGL.

Na W3C o processo de desenvolvimento de especificações consiste na elaboração de rascunhos (*working drafts*) que passam por vários passos de revisão até se tornarem recomendações. As recomendações podem ser implementadas com segurança de que a especiação não mudará substancialmente.

Apesar do processo da W3C ser rigoroso, está longe de perfeito. A especificação final do HTML4 contava com quatro erros publicados via errata (**HTML5**). Não obstante, o cenário é animador (Kuryanovich et al. 2014) cita que as tecnologias da Open WEB tem evoluído desde

[1]Uma lista completa das especificações mantidas pela W3C pode ser encontrada em: http://www.w3.org/TR/

os princípios da internet e já provaram sua robusteza e estabilidade enquanto outras tecnologias crescem e morrem ao redor dela.

A tecnologia chave que inaugurou e alavancou este processo é o HTML.

4.4 HTML

HTML (*Hyper Text Markup Language*) é uma linguagem de marcação que define a estrutura semântica do conteúdo das páginas da web. Criada por Tim Berners Lee em 1989 no CERN. HTML é a tecnologia base para a criação de páginas web e aplicativos online. A parte denominada: "*Hyper Text*", refere-se a links que conectam páginas HTML umas as outras, fazendo a Web como conhecemos hoje (*HTML HyperText Markup Language*).

A última versão do HTML é o HTML5, iniciado pela WHATWG e posteriormente desenvolvido em conjunto com a W3C. Seu rascunho foi proposto em 2008 e ratificado em 2014. Após 2011, a última chamada de revisão do HTML5, a WHATWG decidiu renomear o HTML5 para HTML (*HTML is the new HTML5*). Não obstante, o termo HTML5 permanece em utilização pela W3C.

Além da nomenclatura, exitem pequenas diferenças nas especificações da W3C e WHATWG. A W3C vê a especificação do HTML5 como algo fechado, inclusive já iniciou o desenvolvimento do HTML 5.1. Já a WHATWG vê o HTML5 como uma especificação viva. A postura da W3C tende a criar uma especificação estável, já a da W3C reflete mais a realidade dos navegadores, que nunca implementam uma versão completamente. A Mozilla utiliza a especificação da WHATWG no desenvolvimento do Firefox e recomenda a da W3C para sistemas que requeiram maior estabilidade. Neste trabalho optamos pela nomenclatura da WHATWG, utilizamos o termo HTML em detrimento a HTML5, sempre que semanticamente viável.

HTML foi especificado baseando-se no padrão SGML (*Standard Generalized Markup Language*).

Alguns benefícios do SGML são:

- Documentos declaram estrutura, diferentemente de aparência , possibilitando otimizações nos ambientes de uso (tamanho de tela, etc);
- São portáveis devido a definição de tipo de documento (*document type declaration*);

Apesar de o SGML especificar a não definição de aparência, os criadores de navegadores constantemente introduziam elementos de apresentação como o piscar, itálico, e negrito, que eventualmente acabavam por serem inclusos na especificação. Foi somente nas últimas versões que elementos de apresentação voltaram a ser proibidos reforçando as propostas chave do HTML como uma linguagem de conteúdo semântico, incentivando a utilização de outras tecnologias como o CSS para responder as demandas de apresentação.

Além do HTML, existe o XHTML, que é uma iniciativa de utilização de XML nas páginas da web. O XML é um padrão mais rigoroso que SGML e resulta em páginas sem problemas de

```
<}!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset="UTF-8">
<title></title>
</head>
<body>
    <video>
        <span>Seu navegador não suporta vídeo</span>
    </video>
</body>
</html>
```

Fig. 4.1: Exemplo] de documento HTML

sintaxe e tipografia. Alguns estimam que 99% das paginas HTML de hoje contenham ao menos um erro de estrutura (Pilgrim 2010). Uma das maiores vantagem do XML é que sistemas sem erros de sintaxe que podem ser facilmente interpretadas por outras tecnologias como sistemas de indexação, buscadores, etc.

Para transformar o HTML em algo visível os navegadores utilizam motores de renderização. O primeiro passo efetuado por esses sistemas é decodificar o documento HTML para sua representação em memória. Este processo dá-se através da análise (*parsing*) e posterior tokenização, que é a separação do HTML em palavras chave que o interpretador pode utilizar. Diferentemente do XHTML, HTML não pode ser decodificado através de tokenização tradicional. Deve-se ao HTML ser amigável ao programador, aceitando erros de sintaxe, dependente de contexto, buscando entregar a melhor aproximação possível. Segundo Garisel e Irish 2011 essa é a maior razão do HTML ser tão popular - ele perdoa os erros e torna a vida dos autores da WEB mais fácil. Esta característica deu origem a uma especificação para renderizar HTML (*HTML parser*).

Antes do HTML5 várias versões foram propostas, algumas radicais em seus preceitos. O XHTML 2.0, por exemplo, quebrava com toda a compatibilidade das versões anteriores e acabou por sendo descontinuado. Outrossim, a maioria das versões HTML de grande sucesso foram versões de retrospectiva (*retro-specs*). Versões que não tentavam idealizar a linguagem, buscando alinhar-se com os requerimentos do mercado (Pilgrim 2010). Não obstante, a ideia que a melhor forma de ajustar o HTML é substituindo ele por outra coisa ainda aparece de tempos em tempos (Pilgrim 2010).

Uma página HTML consiste em elementos que podem ter seu comportamento alterado através de atributos. Um elemento é o abrir fechar de uma tag e todo o conteúdo que dentro dele reside (*HTML and CSS - Design and Build Websites*, pp. 10–11). Por exemplo, na figura 4.1 o elemento meta (<meta>) tem um atributo *charset*, que especifica o formato de codificação do documento.

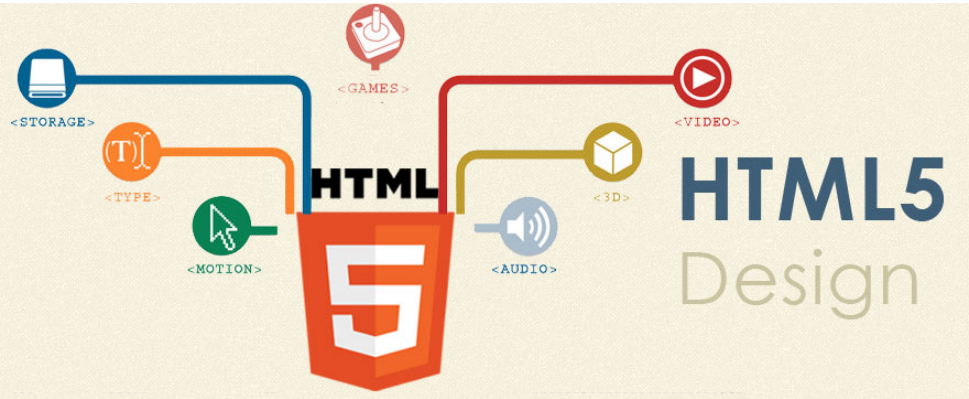

Fig. 4.2: Suíte HTML

Na sua versão inicial, o HTML contava com 18 elementos; atualmente existem aproximadamente cem (Pilgrim 2010). Não obstante, foi no HTML5 que a maior parte dos elementos que viabilizam a construção de jogos foram adicionados.

Uma das características do HTML que o torna tão popular é seu interesse em manter manter a retrocompatibilidade. Interpretadores HTML atingem isso ignorando os elementos que não conhecem, tratando seu vocabulário exclusivamente. Esse mecanismo permite que os desenvolvedores incluam marcação de reserva dentro dos elementos que podem não ser suportados. O elemento *span* na figura 4.1 só aparecerá para o usuário caso seu navegador não suporta a tag vídeo.

Além da convencional linguagem de marcação, HTML é muitas vezes interpretado como um conceito guarda chuva para designar as tecnologias da web. Segundo (Pilgrim 2010) algumas dessas tecnologias (como geolocalização) estão em especificações separadas mas são tratadas como HTML5 também. Outras tecnologias foram removidas do HTML5 estritamente falando, mas são tratados como HTML5 (como a API de armazenamento de dados).

Uma tecnologia fortemente entrelaçada com o HTML é o DOM. Tendo uma relação próxima de um para um com a marcaçaõ (Garisel e Irish 2011). DOM permite a interação entre documentos HTML e as demais tecnologias da WEB de uma forma fácil e padronizada.

4.4.1 DOM

O modelo de documento de objetos (*Document Object Model*) é a representação em memória de uma árvore de elementos HTML. Esta representação é definida por um conjunto de padrões que torna interoperável a manipulação de elementos através de JavaScript.

A primeira versão do DOM, DOM nível zero, foi parcialmente especificada no HTML 4 e permitia manipulação parcial dos elementos. Foi somente com a especificação do JavaScript em 1998 que o DOM nível 1 foi especificado, permitindo a manipulação de qualquer elemento. DOM nível 2 e 3 seguiram com melhorias nas consultas aos elementos e CSS.

A API de seletores (*querySelector*) do DOM permite alto nível de precisão e performance para buscar elementos. A figura 4.3 exemplifica a utilização dos seletores do DOM em um do-

```
var elementos = document.querySelector( ".main, #sceen"  );
var elementosB = document.querySelectorAll( "a.minhaClasse, p"  );
```

Fig. 4.3: Exemplo de utilização de seletores do DOM em JavaScript

cumento JavaScript. O método *querySelector* seleciona o primeiro elemento em conformidade com o padrão especificado. Já o método *querySelectorAll* seleciona todos os elementos que estão em acordo com o padrão especificado.

4.5 CSS

CSS (*Cascading Style Sheets*) é uma linguagem de folhas de estilo criada por Håkon Wium Lie em 1994 com intuito de definir a apresentação de páginas HTML. CSS, juntamente com JavaScript e HTML, compõem as tecnologias centrais no desenvolvimento WEB tornando-se parte da OWP; sua especificação é atualmente mantida pela W3C.

O termo *Cascading* refere-se ao fato de regras serem herdadas pelos filhos de um elemento, eliminando grande parcela de duplicação antes necessária para estilizar uma página. Segundo Kuryanovich et al. 2014 pode-se expressar regras gerais que são "cascateadas"para muitos elementos, e então sobrescrever os elementos específicos conforme a necessidade.

Segundo «Cascading Style Sheets», pp. 23–24:

> CSS possibilita a ligação tardia (*late biding*) com páginas HTML. Essa característica é atrativa para os publicadores por dois motivos. Primeiramente pois permite o mesmo estilo em várias publicações, segundo pois os publicadores podem focar-se no conteúdo ao invés de se preocuparem-se com detalhes de apresentação.

Esta ligação tardia permitiu diferenciação entre apresentação e estrutura, sendo neste caso o CSS responsável pela apresentação. Esta característica é uma das ideias pioneiras do SGML, motivo que tornou a utilização do CSS tão conveniente para o desenvolvimento WEB.

Antes do CSS era impossível ter estilos diferenciados para diferentes tipos de dispositivos, limitando a aplicabilidade dos documentos. Com CSS também tornou-se possível que o usuário declare suas próprias folhas de estilo, um recurso importante para acessibilidade.

Estruturalmente falando, CSS é formado por um conjunto de regras, dentro de uma tag HTML denominada *style*, que são agrupadas por seletores em blocos de declaração. Os elementos selecionados são denominados o assunto do seletor (*Selectors*). Seletores tem o intuito de definir quais partes do documento HTML serão afetadas por determinado bloco de declaração.

A figura 4.4 exemplifica este processo. O seletor em questão é o elemento <*p*>, simbolizando todos os parágrafos. O bloco de declaração é o que está dentro das chaves, aplicando alinhamento e cores aos parágrafos.

```
<style>
p {
    text-align: center;
    color: red;
}
</style>
```

Fig. 4.4: Exemplo de Folha de Estilo

CSS é dividido em módulos, que representam conjuntos de funcionalidades, contendo aproximadamente 50 deles. Cada módulo evolui separadamente, esta abordagem é preferível pois permite uma maior entrega de novas funcionalidades visto que novos recursos não dependem da aceitação de outros para serem disponibilizados.

Além do módulos, CSS também é organizado por perfis e níveis.

Os perfis do CSS organizam a especificação por dispositivo de utilização. Existem perfis para dispositivos móveis, televisores, impressoras, etc. A aplicabilidade das regras do CSS varia dependendo do perfil. O conteúdo do elemento *strong*, por exemplo, pode ser traduzido em uma entonação mais forte em um leitor de telas, já em um navegador convencional pode ser apresentado como negrito.

Já os níveis organizam o CSS por camadas de abstração. Os níveis inferiores representam as funcionalidades vitais do CSS, os níveis superiores dependem dos inferiores para construir as funcionalidades elaboradas.

A primeira especificação do CSS, CSS1 (ou nível 1) foi lançada em 1996. Em 1997 foi lançado o CSS2 com o intuito de ampliar a completude do CSS1. Em 1998 iniciou-se o desenvolvimento do CSS3 que ainda continua em 2015. Além do nível 3 existem módulos de nível 4 no CSS, não obstante o termo CSS3 ainda é o mais utilizado.

Apesar da clara evolução das versões do CSS, esse processo nem sempre é linear. Em 2005 o grupo de trabalho do CSS decidiu aumentar a restrição de suas especificações rebaixando o CSS 2.1, Seletores do CSS3 e Texto do CSS3 de recomendações para rascunhos.

A última versão do CSS, o CSS3, introduziu várias funcionalidades relevantes para jogos, como *media-queries*, transições, transformações 3D, entre outros.

4.5.1 Media Queries

Media Queries permitem aplicar regras a dispositivos específicos, dependendo de suas capacidades, como resolução, orientação, tamanho de tela, entre outros. A especificação prevê a possibilidade de condicionalmente carregar arquivos JavaScript ou CSS, ou utilizar seletores dentro do CSS de acordo com regras de Media Queries.

Esse carregamento condicional permite implementar fluidez e adaptabilidade de layout para diferentes resoluções. Que segundo Janiszewswi 2014 é o mais importante aspecto do desenvolvimento de jogos multiplataforma com as tecnologias da WEB.

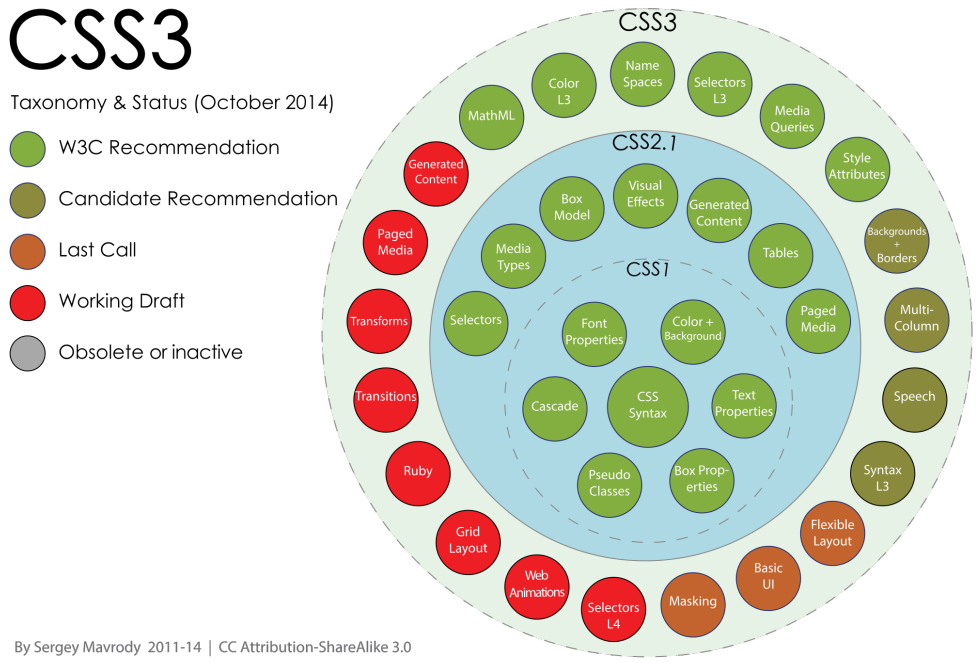

Fig. 4.5: Os módulos do CSS

Fig. 4.6: *
Fonte: https://commons.wikimedia.org/wiki/File:CSS3_taxonomy_and_status-v2.png

```
@media only screen and (min-width: 1024px) {
    background-color: green;
}
```

Fig. 4.7: Exemplo de Media Query

```
@custom-media --narrow-window (max-width: 30em);

<script>
CSS.customMedia.set('--foo', 5);
</script>
```

Fig. 4.8: Exemplo de media queries customizados

Fig. 4.9: *
Fonte: https://developer.mozilla.org/en-US/docs/Web/CSS/MediaQueries

A figura 4.7 demostra a aplicação de uma regra via seletor Media Query, aplicando o a cor de fundo para dispositivos com no minimo 1024 pixels de resolução.

CSS nível 4 permite a utilização de media queries (*Custom Media Queries*) criados pelo usuário, com regras e definições customizadas. A figura 4.8 demostra as novas possibilidades de definição de media queries tanto em CSS como em JavaScript.

4.5.2 Transições

Transições são uma forma de adicionar animações em uma página web. Estas animações são compostas por um estado inicial e um final. A especificação de transições permite grande controle sobre seus estados, habilitando o desenvolvedor a controlar o tempo de execução, os estados intermediários, e efeitos aplicados uma transição.

Para utilizar transições, assim omo em uma máquina de estados, precisamos identificar estados e ações. Estados são seletores do CSS e ações são modificações realizadas entre esses dois seletores CSS (Kuryanovich et al. 2014).

Transições são interessantes em jogos, especialmente pois muitos navegadores suportam aceleração de GPU (Unidade de processamento gráfico) para estas operações. Isso garante grandes benefícios de performance sobre implementações diretamente em JavaScript.

Segundo Kuryanovich et al. 2014 transições nos permitem construir jogos degradáveis pois os interpretadores de CSS são amigáveis; se eles encontrarem propriedades desconhecidas eles simplesmente as ignoram e continuam a funcionar.

A figura 4.10 demostra a utilização de uma transição de tamanho em uma *div* quando o mouse está sobre o elemento. No período de 2 segundos a largura da *div* vai de 100 pixels ara 300 pixels.

Atualmente um conjunto finito de propriedades podem ser animadas com transições, e essas lista tende a mudar com o tempo, cabe ao desenvolvedor assegurar-se que determinada propriedade está disponível (*Using CSS transitions*).

```
div {
    width: 100px;
    height: 100px;
    background: red;
    transition: width 2s;
}

div:hover {
    width: 300px;
}
```

Fig. 4.10: Exemplo de transição

```
<style>
.test:hover
{
        -webkit-transform: scale(1.2);
        -ms-transform: scale(1.2);
        transform: scale(1.2);
}
</style>
```

Fig. 4.11: Exemplo de transformação

4.5.3 Transformações 3D

Transformações é outra tecnologia do CSS3 que permite grande flexibilidade na construção de jogos. Transformações permitem que elementos sejam traduzidos, rotacionados, escalados e distorcidos em um espaço de duas dimensões (Kuryanovich et al. 2014).

A transformação demonstrada na figura 4.11 escala o tamanho do elemento com a classe (*test*) para vinte porcento a mais do seu tamanho original. Perceba também os comandos repetidos com o prefixo ms e WebKit. Esse tipo de abordagem é comum para tecnologias que não passam de rascunhos na especificação.

Assim como transições, as transformações são muitas vezes aceleradas via GPU incrementando a performance de animações criadas com a tecnologia.

4.5.4 CSS 4

Apesar de o termo CSS 4 ser bastante utilizado, o grupo de trabalho do CSS não considera mais a existência de versões, como foi até o CSS3. Não obstante existem recursos cuja especificação está avançada e não estavam presentes no CSS 3 quando este foi lançado, dentre estas funcionalidades inclui-se:

- Suporte a variáveis no CSS

- Media queries customizadas
- Funções de cores como: color(), hwb() e gray()
- Suporte a filtros

Recursos recentes do CSS muitas vezes não estão presentes nos navegadores, não obstante muitos deles são interessantes no contexto de desenvolvimento de jogos, como o suporte a variáveis.

O projeto cssnext http://cssnext.io/ é uma iniciativa para permitir a utilização dos recentes recursos do CSS mesmo sem os mesmos estarem implementados nos navegadores. O projeto funciona compilando o código não suportado em algo compatível com versões para as versões implementadas pelos navegadores.

Além da apresentação, recurso vital para jogos, e aplicativos web em geral, é a iteratividade. Com as tecnologias da WEB esta iteratividade é atingida através do JavaScript.

4.6 JAVASCRIPT

EMACScript, melhor conhecida como JavaScript, criada por Brendan Eich em 1992, é a linguagem de script da Web. Devido a tremenda popularidade entre comunidade de desenvolvedores a linguagem foi abraçada pela W3C e atualmente é um dos componentes da Open Web.

As definições da linguagem são descritas na especificação ECMA-262. Esta possibilitou o desenvolvimento de outras implementações além da original (SpiderMonkey) como o Rhino, V8 e TraceMonkey; bem como outras linguagens similares como JScript da Microsoft e o ActionScript da Adobe.

Segundo a *ECMAScript 2015 Language Specification*:

> Uma linguagem de script é uma linguagem de programação que é usada para manipular e automatizar os recursos presentes em um dado sistema. Nesses sistemas funcionalidades já estão disponíveis através de uma interface de usuário, uma linguagem de script é um mecanismo para expor essas funcionalidades para um programa protocolado.

No caso de JavaScript na web, os recursos manipuláveis são o conteúdo da página, elementos HTML, elementos de apresentação, a própria janela do navegador e variados outros recursos que tem suporte adicionado por novas especificações.

A intenção original era utilizar o JavaScript para dar suporte aos já bem estabelecidos recursos do HTML, como para validação, alteração de estado de elementos, etc. Em outras palavras, a utilização do JavaScript era opcional e as páginas da web deveriam continuar operantes sem a presença da linguagem.

Não obstante, com a construção de projetos Web cada vez mais complexos, as responsabilidades delegadas ao JavaScript aumentaram a ponto que a grande maioria dos sistemas web não funcionarem sem ele. JavaScript não evoluiu ao passo da demanda e muitas vezes carece de definições expressivas, completude teórica, e outras características de linguagens de programação mais bem estabelecidas, como o C++ ou Java (Barnett 2014).

A última do JavaScript, o JavaScript 6, é um esforço nessa direção. JavaScript 6 ou EMACScript Harmonia, contempla vários conceitos de orientação a objetos como classes, interfaces, herança, tipos, etc. Não obstante o suporte ao JavaScript 6 é apenas parcial em todos os navegadores. O site http://kangax.github.io/compat-table/es6/ apresenta um comparativo de suporte das funcionalidades do JavaScript.

Segundo *ECMAScript 6 Today* o suporte no início de 2015 era o seguinte:

- Chrome: 30%
- Firefox: 57%
- Internet Explorer : 15%
- Opera: 30%
- Safari: 19%

Estes esforços de padronização muitas vezes não são rápidos o suficiente para produtores de software web, demora-se muito até obter-se um consenso sobre quais as funcionalidades desejadas em determinada versão e seus detalhes de implementação. Além da espera por especificações, uma vez definidas, é necessário que os navegadores especificado.

O projeto babel https://github.com/babel/babel é um compilador de JavaScript 6 para JavaScript 5. Permitindo que, mesmo sem suporte, os desenvolvedores possam usufruir dos benefícios da utilização do JavaScript 6 durante o tempo de desenvolvimento, gerando código em JavaScript 5 para rodar nos navegadores.

Alternativamente, existe uma vasta gama de conversores de código - (*transpilers*) - para JavaScript; possibilitando programar em outras linguagens posteriormente gerando código JavaScript. Entretanto, essa alternativa tem seus pontos fracos, necessita-se de mais tempo de depuração, visto que o JavaScript gerado não é conhecido pelo desenvolvedor, e provavelmente o código gerado não será tão otimizado, nem utilizará os recursos mais recentes do JavaScript.

Mesmo com suas fraquezas amplamente conhecidas, JavaScript está presente em praticamente todo navegador atual. Sendo uma espécie de denominador comum entre as plataformas. Essa onipresença torna-o integrante vital no processo de desenvolvimento de jogos multiplataforma em HTML5. Vários títulos renomeados já foram produzidos que fazem extensivo uso de JavaScript, são exemplos: Candy Crush Saga, Angry Birds, Dune II, etc.

Jogos Web são geralmente escritos na arquitetura cliente servidor, JavaScript pode rodar em ambos estes contextos, para tanto, sua especificação não define recursos de plataforma.

Distribuidores do JavaScript complementam a o JavaScript com recursos específicos para suas plataformas alvo. Por exemplo, para servidores, define-se objetos como: console, arquivos e dispositivos; no contexto de cliente, são definidos objetos como: janelas, quadros, DOM, etc.

Para o navegador o código JavaScript geralmente é disposto no elemento script dentro de arquivos HTML. Quando os navegadores encontram esse elemento eles fazem a requisição para o servidor e injetam o código retornado no documento, e a não ser que especificado de outra forma, iniciam sua execução.

4.6.1 JAVASCRIPT 7

Antes da finalização da especificação 6, algumas funcionalidades do JavaScript 7 já haviam sido propostas. Na página https://github.com/tc39/ecma262 pode-se conferir os itens propostos e seu estágio de evolução.

Alguns dos recursos esperados para o JavaScript 7 são: guards, contratos e concorrência no laço de eventos (*A first look at what might be in ECMAScript 7 and 8*).

A figura 4.13 é a tabela de funcionalidades sugeridas e seu estágio no caminho da especificação.

4.6.2 ASM.JS

Asm.js é um subconjunto da sintaxe do JavaScript a qual permite grandes benefícios de performance quando em comparação com JavaScript normal. Entretanto, não é trivial escrever código em asm.js e geralmente a criação de código asm.js é feita através da conversão de outras linhagens como C. O projeto Emscripten https://github.com/kripken/emscripten pode ser utilizado para gerar código em asm.js é utilizado pelo motor de jogos Unity 3D e Unreal.

No contexto dos jogos performance é um fator de extrema importância asm.js se destaca por utilizar recursos que permitam otimizações antes do tempo (*ahead of time optimizations*). Grade parcela da performance adicional, em relação ao JavaScript, é devido a consistência de tipo e a não existência de um coletor de lixo (*garbage collector*) - a memória é gerenciada manualmente através de um grande vetor. Esse modelo simples desprovido de comportamento dinâmico, sem alocação e desalocação de memória, apenas um bem definido conjunto de operações de inteiros e flutuantes possibilita grade performance e abre espaço para otimizações.

O desenvolvimento do asm.js iniciou-se no final de 2013 não obstante a maioria dos navegadores não implementam ou implementam parcialmente o rascunho. O motor JavaScript da Mozilla, SpiderMonkey, é a exceção, implementando a grande maioria dos recursos do asm.js.

4.6.3 WEB Assembly

Web Assembly é uma tecnologia recentemente proposta que pretende se tornar o formato de máquina da Web. Irá permitir que outras linguagens além do JavaScript gerem código

	Proposal	Champion	Stage
	Array.prototype.includes	Domenic Denicola, Rick Waldron	4
	Exponentiation Operator	Rick Waldron	3
	SIMD.JS - SIMD APIs + polyfil	John McCutchan, Peter Jensen, Dan Gohman, Daniel Ehrenberg	3
	Async Functions	Brian Terlson	3
	Object.values/Object.entries	Jordan Harband	3
	String padding	Jordan Harband & Rick Waldron	3
	Trailing commas in function parameter lists and calls	Jeff Morrison	3
	function.sent metaproperty	Allen Wirfs-Brock	2
	Rest/Spread Properties	Sebastian Markbage	2
	ArrayBuffer.transfer	Luke Wagneer & Allen Wirfs-Brock	1
	Additional export-from Statements	Lee Byron	1
	Class and Property Decorators	Yehuda Katz and Jonathan Turner	1
	`Function.prototype.toString` revision	Michael Ficarra	1
	Observable	Kevin Smith & Jafar Husain	1
	String.prototype.{trimLeft,trimRight}	Sebastian Markbage	1
	Class Property Declarations	Jeff Morrison	1
	String#matchAll	Jordan Harband	1
	Shared memory and atomics	Lars T Hansen	1
	Callable class constructors	Yehuda Katz and Allen Wirfs-Brock	1
	System.global	Jordan Harband	1

Fig. 4.12: Propostas do ECMA 7

Fig. 4.13: *
Fonte: https://github.com/tc39/ecma262

que rode nativamente nos navegadores com grande ganhos de performance e flexibilidade, visto que será possível utilizar qualquer linguagem para gerar código especificado pelo Web Assembly.

WebAssembly (short: wasm) is a new binary format for the WEB, created by Google, Microsoft, Mozilla and others. It will be used for performance critical code and to compile languages other than JavaScript (especially C/C++) to the WEB platform. It can be seen as a next step for asm.JavaScript

In general, by keeping the non-Web path such that it doesn't require Web APIs, WebAssembly could be used as a portable binary format on many platforms, bringing great benefits in portability, tooling and language-agnosticity (since it supports C/C++ level semantics).

External code bases, especially those in C/C++, will be easy to port to the WEB platform, via WebAssembly.

First, this is a collaborative effort, no single company goes it alone. At the moment, the following projects are involved: Firefox, Chromium, Edge and WebKit. Third, this is not about replacing JavaScript engines, it is more about adding a new feature to them. That greatly reduces the amount of work to implement WebAssembly and should help with getting the support of the WEB development community. j

WebAssembly is not bytecode: Bytecode is linear and (usually) stack-, register- or SSA-based, WebAssembly is a binary format for an abstract syntax tree (AST). Compared to bytecode, this has the following advantages:

WebAssembly is relatively easy to add to all current JavaScript engines, because it is high-level and similar to parts of JavaScript. That is, it builds on the infrastructures of engines, instead of replacing them. Engines will continue to have different compilation strategies and/or bytecode, which is good for the WEB ecosystem, because the differences have fostered experimentation and innovation.

WebAssembly is not versioned. It uses the same evolution strategy as JavaScript (feature detection and polyfills).

For the Unity Game Engine, first tests we're made with WebAssembly.

WebAssembly will include both a binary notation, that compilers will produce, and a corresponding text notation, suitable for display in debuggers or development environments. Early prototypes are already showing some of the expected advantages; the binary representation is 20 times faster to parse than the equivalent asm.JavaScript.

4.6.4 Web Animations

Web Animations defines a model for supporting animation and synchronization on the Web platform. It is intended that other specifications will build on this model and expose it's features through declarative means. In addition, this specification also defines a programming interface to the model that may be implemented by user agents that provide support for scripting.

A especificação ainda está em estado de rascunho.

```
elem.getAnimations().filter(
  animation =>
    animation.effect instanceof
    KeyframeEffectReadOnly &&
    animation.effect.getFrames().some(
      frame => frame.hasOwnProperty('transform')
    )
).forEach(animation => {
  animation.currentTime = 0;
  animation.playbackRate = 0.5;
});
```

Fig. 4.14: Exemplo de utilização de WEB Animatios

Fig. 4.15: *
Fonte: http://www.w3.org/TR/WEB-animations/

4.7 NAVEGADORES

Navegadores são as plataformas de WEB, onde as tecnologias até então descritas são interpretadas e geram um conteúdo útil para os usuários.

Aplicações do lado do cliente geralmente se comunicam com um servidor através de documentos em HTTP. Quado o navegador recebe um destes pacotes em HTML ele começa o processo de renderização. A renderização pode requisitar outros arquivos a fim de completar a experiência desenvolvida para o endereço em questão.

Nos navegadores os usuários necessitam localizar a página que desejam, sabendo o endereço, ou pesquisando em buscadores. Isso é um processo árduo para a plataformas móveis pois necessitam maior interação do usuários e não são "naturais" se comparado ao modo normal de consumir aplicativos nestas mesmas plataformas – simplesmente adquirindo o aplicativo na loja e abrindo-o no sistema operacional. Alguns contornos para este problema serão descritos nas tecnologias offline.

Para transformar as instruções retornadas pelo servidor em algo útil para o usuário final os navegadores geralmente fazem uso de bibliotecas externas capazes de interpretar HTML5 e gerar o conteúdo iterativo.

Os navegadores são geralmente compostos por um motor de renderização (*engine*) e por um motor de JavaScript.

Alguns motores de renderização incluem:

- Blink: Utilizada no Chromium e projetos relacionados, Opera
- Gecko: Utilizada nos produtos da Mozilla
- KHTML: Utilizada no navegador Konkeror, esta serviu de base para o Blink

- WebKit: Utilizada no Safari e versões antigas do Google Chrome.

Para interpretar HTML o motor WebKit utiliza a biblioteca Bison, já o Gecko utiliza uma biblioteca própria (Garisel e Irish 2011).

Alguns motores de JavaScript incluem:

- SpiderMonkey: Primeiro motor, desenvolvido por Brendan Eich, escrito em C++
- Rhino: Criada pela Netscape, escrito em Java
- Nitro: Criada pela Apple
- V8: Criada pelo Google
- TraceMonkey: Criada pela Mozilla

O suporte ao HTML vem crescendo com o tempo, o site HTML5Test http://html5test.com/about.html, oferece um placar atualizado dinamicamente, conforme utilização dos navegadores, sobre os recursos do HTML.

A figura 4.18 apresenta o gráfico de suporte por versões de navegadores em dezembro de 2015.

4.8 ANDROID

É um sistema operacional open-source desenvolvido pela Google. Utiliza o kernel Linux . Softwares para Android são geralmente escritos em Java e executados através da máquina virtual Dalvik.

É similar a máquina virtual Java, mas roda um formato de arquivos diferenciado (dex), otimizados para consumir pouca memória, que são agrupados em um único Android Package (apk) Android permite a renderização de documentos HTML através de sua própria API WEBVIEW. Ou através do navegador disponibilizado por padrão, ou outros de terceiros como o Google Chrome, Firefox, Opera, etc.

No quesito jogos para dispositivos móveis é preferível disponibilizar os jogos através da interface nativa pois dá a sensação de continuidade para com os demais aplicativos instalados no dispositivo.

4.9 DETECÇÃO DE RECURSOS

Visto que nenhum navegador implementa as especificações HTML completamente cabe ao desenvolvedor detectar os navegadores que não comportam as necessidades tecnológicas dos aplicativos que cria. Ao deparar-se com uma funcionalidade faltante o desenvolvedor tem duas possibilidades: notificar o usuário sobre o problema ou utilizar polyfills.

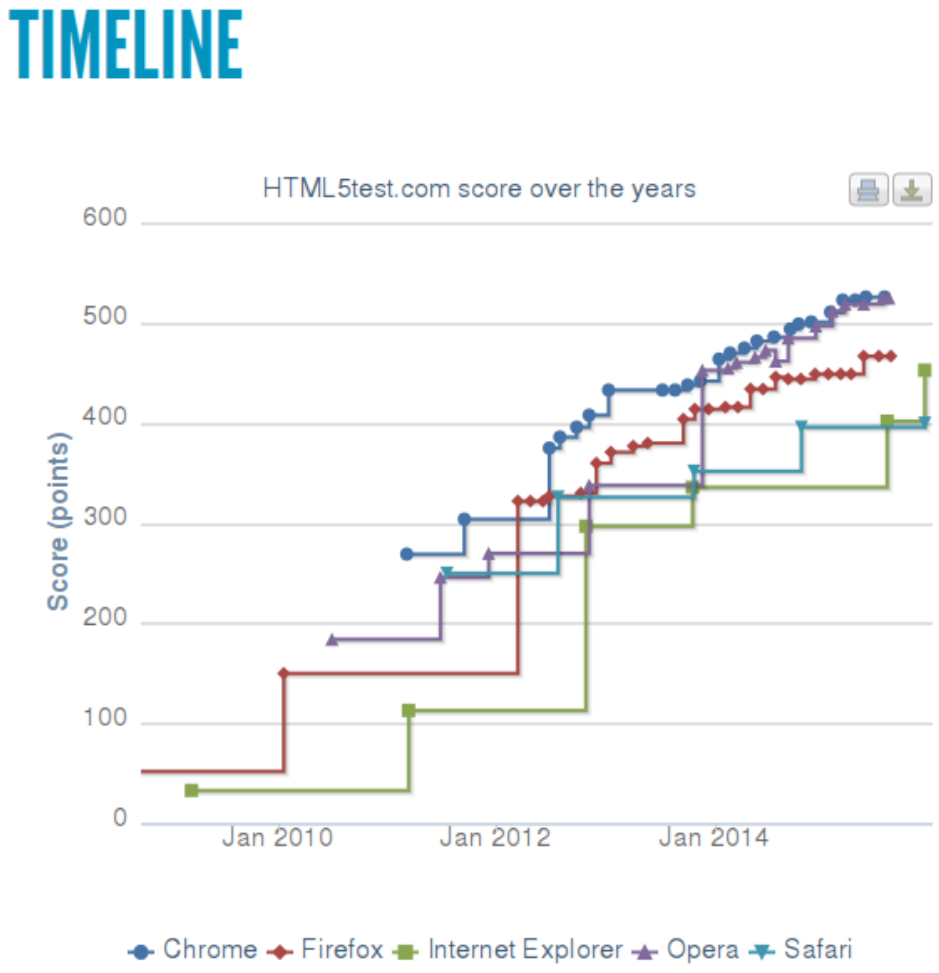

Fig. 4.16: Suporte das especificações do HTML nos navegadores

Polyfills são recursos que simulam uma funcionalidade não disponível nativamente nos navegadores. A biblioteca Gears https://developers.google.com/gears é um exemplo. Gears serve para prover recursos de Geolocalização para navegadores que não implementam a especificação do HTML5.

Essa capacidade de suportar tecnologias que não estão ainda disponíveis (ou nunca estarão no caso de dispositivos legados) através de polyfills é uma das características que faz a WEB uma plataforma de tão grande abrangência. Novas tecnologias são criadas a todo o momento; entretanto, o suporte a essas funcionalidades geralmente não acompanham o passo das inovações. E ainda assim os usuários podem se beneficiar de uma taxa substancial delas através de polyfills.

Algumas funcionalidades do HTML, como geolocalização e vídeo foram primeiramente disponibilizadas através de plugins. Outras funcionalidades, como o Canvas, podem ser totalmente emuladas via JavaScript (Pilgrim 2010).

Detectar suporte aos variados recursos do HTML5 no navegador pode ser uma tarefa entediante. É possível implementar testes para cada funcionalidade utilizada abordando os detalhes de implementação de cada uma ou então fazer uso de alguma biblioteca especializada neste processo. O Modernizr é uma opção open-source deste tipo de biblioteca, este gera uma lista de booleanos sobre grande variedade dos recursos HTML5, dentre estes, geolocalização, canvas, áudio, vídeo e armazenamento local.

A quantidade de especificações que um aplicativo complexo como um jogo utiliza pode ser bem grande, e muitas vezes é difícil dizer qual quais navegadores implementam o quê. Uma boa referência do suporte a recursos nos navegadores é o site http://caniuse.com/.

4.10 Pré Carregamento

4.11 RENDERIZAÇÃO

Renderização é parte fundamental de muitos jogos. As tecnologias atualmente existes são o SVG e Canvas.

4.11.1 SVG

SVG (*Gráficos de vetores escaláveis*), é uma linguagem baseada em XML especializada na criação de vetores bidimensionais (Kuryanovich et al. 2014). Por usar XML SVG permite a utilização da API do DOM para manipular os elementos.

Entre os benefícios do SVG encontram-se:

- Não há diferença de qualidade em resoluções pois os vetores são escaláveis;
- Suporta animações nativamente;

```
<svg height="100" width="100">
  <circle cx="50" cy="50" r="40" stroke="black" stroke-width="3"/>
  Sorry, your browser does not support inline SVG.
</svg>
```

Fig. 4.17: Círculo em SVG.

- Conta com integração através da api do DOM. Tornando simples a integração com as outras tecnologias da web.

Um aspecto negativo do SVG é que é muito difícil atingir a perfeição na posição dos pixels, por ser uma linguagem vetorizada (Kuryanovich et al. 2014).

4.11.2 CANVAS

A tag *canvas* define uma camada de mapa de bits em documentos HTML que pode ser usada para criar diagramas, gráficos e animações 2D. Foi criada pela Apple em 2004 para renderizar elementos de interface no Webkit, logo foi adotado por outros navegadores e se tornou um padrão.

A API canvas cresceu com o tempo, e algumas funcionalidades - como suporte a texto, foram adicionadas tardiamente. E alguns navegadores lançaram antes da especificação estar completa e hoje tem problema de suporte para essas áreas (Pilgrim 2010). Apesar de muitas vezes incompleto, o canvas é suportado em todos os maiores navegadores à partir do Internet Explorer 9. Não obstante existe a possibilidade de utilizar a biblioteca https://github.com/google/canvas-5-polyfill para suprir aos navegadores que ainda não implementaram, os últimos recursos da especificação CANVAS, como o recurso de desenhar elipses e o caminho 2D.

Apesar de ser geralmente bem suportado nos navegadores atuais, canvas ainda sofre de problemas de performance. Para navegadores antigos - abaixo do Internet Explorer 9 - existe o polyfill Explorer Canvas do Google, que emula em JavaScript as funcionalidades do canvas. O FastCanvas é uma iniciativa híbrida para Android que busca mitigar os problemas de performance do Canvas com uma API nativa. Não obstante, o FastCanvas não suporta a especificação do canvas completamente, não permite ser integrado com outros elementos do DOM.

Em um documento HTML, Canvas é um retângulo onde pode-se usar JavaScript para desenhar (Pilgrim 2010, pp. 113).

Canvas tem uma característica peculiar quanto ao seu tamanho. Em suma existem dois tamanhos, o tamanho do elemento e da superfície de desenho. Quando o tamanho do elemento é maior do que o da superfície de desenho do documento escala a superfície de desenho para preencher o elemento, o que pode gerar resultados inesperados.

Algumas outras fraquezas do canvas são:

- Não há suporte a animações

- Não é possível alterar uma parte já desenhada a não ser sobrescrevendo ou pintando o canvas novamente
- Por ser um vetor de pixeis não existe possibilidade de utilização do DOM nem seus eventos, limitado muito a iteratividade

O canvas até aqui descrito trata-se de sua forma, ou contexto 2D. A especificação 3D do canvas é o WebGl.

4.11.3 WEBGL

WebGL é uma API JavaScript para desenhar gráficos em três dimensões. Apesar de ter sido desenvolvido com foco em 3D, WebGL pode ser utilizados para 2D também (Matti e Arto 2011, pp. 6).

A especificação do WebGL foi desenvolvida baseando-se em OpenGL, especificamente OpenGL ES 2.0, uma versão do OpenGL otimizada para dispositivos móveis. O órgão que especifica o WebGL é o mesmo que especifica o OpenGL: Kronos Working Group.Os primeiros rascunhos do WebGL iniciaram em 2006, não obstante o grupo de trabalho não foi formado até 2009. E a primeira versão do foi lançada em 2011.

O elemento do DOM que provê a interface do WebGL é o canvas, no contexto 3D. Essa integração com o DOM permite que o WebGL seja manipulado assim como os demais elementos HTML.

É digno de nota que é recomendável se testar WebGL em todos os navegadores e plataformas alvo. Pode haver diferenças substanciais de performance em plataformas diferentes e em navegadores diferentes. Existe também uma lista de placas gráficas com drivers bloqueados por não funcionarem corretamente no Firefox (Matti e Arto 2011, pp.42).

WebGL só se tornou uma possibilidade performática para jogos pois parte do seu processamento pode ser delegado para a GPU. A especificação é composta por uma API de controle em JavaScript e o processamento shaders. Shaders são responsáveis por grande parcela dos efeitos 3D e são executados do lado da GPU.

4.11.4 Shaders

Shaders definem níveis de cor ou efeitos especiais sobre um modelo 2d ou 3D. Contam com grande performance, possibilitando conteúdo em tempo real - como no caso dos jogos - por executarem dentro da GPU. São utilizados no cinema, em imagens geradas por computadores e vídeo games.

Existem dois shaders principais, de vértices e fragmentos. Shaders de vértices são chamados para cada vértice sendo desenhado definindo suas posições e shaders de fragmentos a cor para cada pixel a ser desenhado (Matti e Arto 2011, pp.15).

4.11.5 Bibliotecas OpenGL

CocoonJS é uma aplicativo híbrido que preenche a fraca implementação de WebGL nos dispositivos móveis possibilitando se desenvolver em WEBGL, CSS. Conta com suporte a dispositivos legados à partir do Android 2.3 e IPhone 5.

Treejs é uma abstração sobre WebGL que permite os autores se focarem na criação de conteúdo para web, ao invés de dispenderem tempo manipulando os detalhes da WebGL. Possibilita trabalhar com efeitos, luzes, cenas e outras abstrações em detrimento de shaders, vértices, e conceitos mais primitivos.

4.11.6 Otimizações

CPU? JavaScript, DOM, etc. GPU?

- Texture compression - Mipmaps (LOD) - Draw calls

Um site interessante para explorar exemplos WebGL avançados é o blog http://learningwebgl.com que conta com tutoriais cobrindo áreas como diferentes tipos de iluminação, carrecamento de modelos em json, gerenciando eventos do mouse e teclado; e como renderizar uma cena WebGL em uma textura (Matti e Arto 2011, pp.42).

Apesar da relevância, WebGL não foi utilizada no trabalho pois ainda não está completamente especificada e a dificuldade e escopo do projeto aumentariam muito se tivessem de incluir um jogos 3D.

Uma tecnologia que se integra profundamente como ambientes virtuais em três dimensões criados via OpenGL é o WebVR.

4.11.7 WEBVR

Segundo Brooks 1999 realidade virtual é uma experiência em que o usuário é efetivamente imerso em um mundo virtual responsivo. Realidade virtual é uma área nem tão nova mas que recebeu interesse renovado recentemente. Isso se dá, pelo menos me parte, pela massificação dos dispositivos móveis inteligentes. O hardware necessário para fornecer uma experiência minimamente viável como acelerômetros, câmeras e telas de alta resolução está disponível em praticamente todos os dispositivos comercializados.

Realidade virtual é uma área de grande interessa para os produtores de jogos, pois pode oferecer alto nível de imersão nos já interativos e desafiadores ambientes dos jogos.

A WebVR é uma especificação que pretende trazer os benefícios da realidade virtual para dentro do mundo da WEB. Em termos simples a especificação define uma forma de traduzir movimentos de acelerômetros e outros sensores de posição e movimentos para dentro do contexto de uma um contexto 3D através de JavaScript.

Atualmente a especificação do WebVR se encontra em fase de rascunho e as últimas versões do Firefox, e versões compiladas manualmente do Google Chrome já permitem a utilização.

4.12 WebCL

É uma API em JavaScript para o recursos de OpenCL que permitem computação paralela com grandes ganhos de performance. Aplicativos como moteres de física e renderizadores de imagens, ambos relevantes para os jogos, podem se beneficiar grandemente de processamento feito em paralelo, possivelmente na GPU. OpenCL é um framework para escrever programas que funcionem em plataformas com diversas unidades de processamento, assim como WebGL e WebCL é especificada e desenvolvida pelo grupo Kronos.

A primeira versão da especificação foi no início de 2014 mas até então nenhum navegador implementa o definido.

4.13 CODECS

Para falar sobre audio e vídeo precisamos primeiramente introduzir o conceito de codecs. Codec - é o algorítmo usado para encodificar o video em um conjunto de bits Pilgrim 2010.

Codecs é uma área problemática do HTML5. Segundo Pilgrim 2010 não existe uma única combinação de containers e codecs que funcionem em todos os navegadores.

4.14 AUDIO

Áudio é um componente vital para oferecer imersão e feddback aos usuários de jogos. Efeitos de som e música podem servir como mecanismo. Por outro lado, jogadores tem baixa tolerância a volume, deve ser utilizado com cautela.

Segundo Vahatupa 2014 em sido difícil construir aplicações sofisticadas e interativas sem a utilização de plugins para áudio.

O componente de áudio é especialmente útil para jogos de ação (Vahatupa 2014).

4.14.1 TAG AUDIO

A tag <audio> define um som dentro de um documento HTML. Quando o elemento é renderizado pelos navegadores, ele carrega o conteúdo que pode ser reproduzido pelo player de audio do navegador. Existem muitas discrepâncias entre os formados aceitáveis pelos navegadores. É um tanto limitada quanto comparada ao áudio de múltiplos canais disponibilizados por SDKs nativas.

4.14.2 API DE AUDIO

É uma interface de audio experimental para JavaScript. Provê maior flexibilidade na manipulação de audio. Essa tecnologia é muito mais nova do que a tag áudio. Abaixo segue alguns dos codecs de audio tradicionais:

- MP3: comum, mas não é livre de patentes

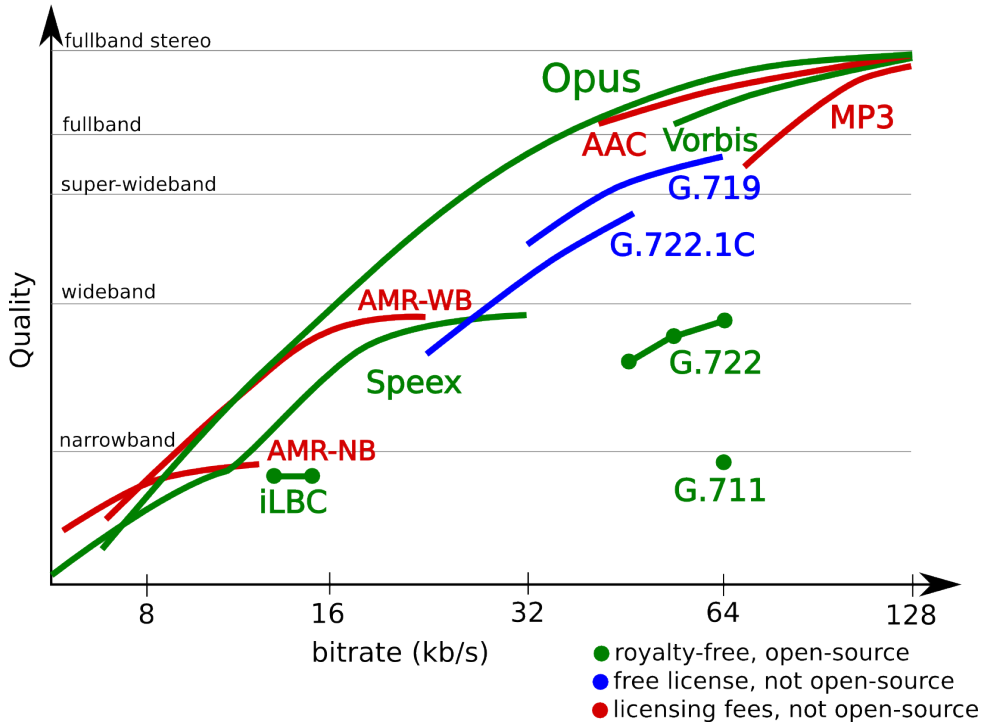

Fig. 4.18: Comparação de codecs de áudio

- ACC: advanced audio coding, is patent encumbered
- Vorbis - patent free

A figura 4.18 é um comparativo interessante sobre os formatos existentes, considerando o fator taxa de bits versus qualidade.

4.15 VIDEO

Antes do HTML5 era impossível adicionar vídeos nas páginas sem a utilização de algum plugin.

A especificação define uma tag *video* que pode ser embida em uma página HTML. Segundo Pilgrim 2010 não existem restrições quando ao codec de vídeo ou áudio, um elemento vídeo pode fazer referência a múltiplos arquivos de vídeos, cabe ao navegador decidir qual arquivo de fato será executado.

Os navegadores não concordam em qual formato de vídeo suportar. Uma tag vídeo pode apontar para vários arquivos em diversos formatos, e os navegadores que suportarem determinado irão escolhê-lo.

Um formato de vídeo é a combinação de várias tecnologias.

AVI e MP4 são apenas containers de formatos. Como um arquivo zip, podendo conter qualquer coisa dentro de si (Pilgrim 2010).

Existem formatos desenvolvidos especificamente para a Web. Buscam uma razão de tamanho e qualidade aceitável, mas prezando por tamanho. A maioria dos codecs de vídeo não

mudam todo o conteúdo de um quadro para o próximo, possibilitando maiores taxas de compressão, que resulta em arquivos menores (Pilgrim 2010).

O projeto *Vídeo for Everybody*

`http://camendesign.com/code/video_for_everybody`

é um polyfill que recorre à flash quando o vídeo não é suportado pelo navegador.

Segue uma lista de alguns dos containers de video:

- MPEG4
- Flash Video
- Ogg (for video Theora), (audio Vorbis)
- WebM
- Matroska
- Audio Video Interleave

Alguns codecs de vídeo

- H.264, is one of the video codecs mandated by the Blue-Ray specification
- Theora, native in Linux
- VP8 royality free from Google

4.16 ARMAZENAMENTO

Uma das grades limitações do HTML era a ausência de capacidade de armazenamento de dados no lado do cliente. Antes do HTML5 a única alternativa era usar cookies, os quais tem um armazenamento de no máximo 4k e trafegam em toda a requisição, tornando o processo lento. Essa área era ode as aplicações nativas detinham grande vantagem sobre as aplicações web. O HTML5 solucionou este problema introduzindo várias formas de armazenamento de dados («Smart Phones Application development using HTML5 and related technologies: A tradeoff between cost and quality»).

Armazenamento local é um recurso importante para jogos, tanto por diminuir a latência da persistência na rede, quanto para possibilitar um experiência offline.

Existem algumas especificações sobre armazenamento, mas a grande parte delas não conta como suporte completo em todos os navegadores comuns, um polyfill interessante para Web Storage e IndexedDB é o projeto localForge *https://github.com/mozilla/localForage* da Mozilla.

4.16.1 WEB SQL

A especificação Web SQL introduz uma API para manipular banco de dados relacionais em SQL. A especificação suporta transações, operações assíncronas e um tamanho de armazenamento substancial: 5 megabytes, o qual pode ser estendido pelo usuário.

O grupo de trabalho do Web SQL iniciou-se em 2010 e foi suspendido ainda como rascunho. Apesar de ser um recurso desejável para muitos desenvolvedores, foi descontinuada pelos motivos descritos abaixo.

Segundo Pilgrim 2010

> Todos os implementadores interessados em Web SQL utilizaram a mesma tecnologia (Sqlite), mas para a padronização ficar completa é necessário múltiplas implementações. Até outro implementador se interessar em desenvolver a especificação a descrição do dialeto SQL apenas referencia o SQLITE, o que não é aceitável para um padrão.

Não obstante, a especificação ainda é suportada pelo Google Chrome, Safari, Opera e Android, entre outros. Mas até que outros implementadores se prontifiquem a especificação continuará suspensa. No lugar do Web SQL a W3C recomenda a utilização do Web Storage e do IndexedDB.

4.16.2 WEB STORAGE

Web Storage, também conhecido como Local Storage, provê uma forma de armazenar os dados como chave valor dentro do navegador. Os dados são persistidos mesmo que o navegador seja fechado.

É um recurso similar a cookies, contudo algumas diferenças substanciais são perceptíveis. Web Storage não requer que os dados sejam trafegados como cabeçalhos nas requisições. Também provê maiores espaços de armazenamento quando comparado a cookies.

A tecnologia começou como parte da especificação do HTML5 mas agora conta com um documento próprio mantido pela W3C. A especificação é suportada pela grande maioria dos navegadores populares.

A especificação oferece duas áreas de armazenamento, o armazenamento local e de sessão. O armazenamento local é persistido por domínio e outros scripts provindos deste mesmo domínio poderão fazer uso da informação. O armazenamento de sessão é para informações que podem variar de aba para aba e que não é interessante que sejam persistidos para demais acessos além do atual.

A API do Web Storage é simples, consistindo em uma interface para buscar dados e outra para armazenar, no formato chave/valor.

A figura 4.19 exemplifica a utilização do Web Storage, para utilizar o armazenamento de sessão utiliza-se o objeto sessionStorage. Já para utilizar o armazenamento local utiliza-se o objeto localStorage.

```
// Store value on browser for duration of the session
sessionStorage.setItem('key', 'value');

// Retrieve value (gets deleted when browser is closed and re-opened)
alert(sessionStorage.getItem('key'));

// Store value on the browser beyond the duration of the session
localStorage.setItem('key', 'value');

// Retrieve value (persists even after closing and re-opening the browser)
alert(localStorage.getItem('key'));
```

Fig. 4.19: Web Storage na prática

Fig. 4.20: *
Fonte: https://en.wikipedia.org/wiki/Web_storage#usage

Web Storage é uma solução simples que comporta muitos casos de uso. Não obstante muitas vezes é necessário um controle mais refinado sobre os dados, ou mais performance em uma base de dados massiva. Para responder a estes desafios existe a especificação do IndexedDB.

4.16.3 IndexedDB

IndexedDB é um banco de dados que suporta o armazenamento de grandes quantidades de dados no formato de chave/valor o qual permite alta performance em buscas baseadas em índices. A tecnologia é uma recomendação da W3C desde janeiro de 2015 e suportada, pelo menos parcialmente, por praticamente todos os navegadores populares.

Inicialmente IndexedDB permitia operações síncronas e assíncronas. Não obstante, a versão síncrona foi removida devido a falta de interesse da comunidade. Operações assíncronas permitem que aplicativos JavaScript nunca esperam pelo resultado para continuar a execução. Outrossim, cada interação com o banco de dados é uma transação que pode retornar um resultado ou um erro. Os eventos da transação são internamente eventos DOM cuja propriedade *type* do elemento foi setada para *success* ou *error*.

Ao invés de tabelas, IndexedDB trabalha com repositórios de objetos. Cada entrada, tupla em SQL, de um determinado repositório pode ser de um formato diferenciado, com exceção da chave única que deve estar presente em cada uma das entradas.

A figura 4.21 demonstra um exemplo simplificado da utilização do IndexedDB, como cada iteração com o banco de dados é construído através de uma nova requisição e o tratamento do resultado é dado dentro de eventos.

Apesar de ser desenvolvido com objetivo de ser uma solução para todas as necessidades de armazenamento no Frontend IndexedDB ainda sofre algumas limitações.

Abaixo segue uma lista com algumas das limitações do IndexedDB.

```
var db;
var request = window.indexedDB.open("Mydb", 9);
request.onsuccess = function(event) {
db = event.target.result;
var transaction = db.transaction(["customers"], "readwrite");
var objectStore = transaction.objectStore("customers");
var request = objectStore.add({email: "mymail@domain.com", name: "foo"});
request.onsuccess = function(event) {
console.log('customer added')
};
}
```

Fig. 4.21: Adicionando um cliente em IndexedDB.

- Tem limites de armazenamento e as regras variam de navegador para navegador.
- O comportamento em abas anônimas não está especificado e os resultados também variam.
- Existe uma pequena probabilidade de os dados se perderem, no caso do Firefox a API não espera confirmação do sistema operacional para considerar um dado válido, essa foi uma escolha em detrimento de performance.
- Não existe a possibilidade de fazer buscas em textos como o *LIKE* do SQL.
- o usuário pode configurar o navegador para não aceitar armazenamento local para determinado domínio.

A característica assíncrona do IndexedDB, é fundamentada na premissa de não perturbar o fluxo principal da aplicação enquanto processamento não vital, e possivelmente demorado, ocorre. Outra tecnologia da web que utiliza os mesmos princípios é o Web Workers.

4.17 WEB WORKERS

É uma API que possibilita executar vários scripts (*threads*) JavaScript ao mesmo tempo. O script que cria uma thread é chamado de pai da thread, e a comunicação entre pai e filhos pode acontecer de ambos os lados através de mensagem encapsuladas em eventos. Um script que não seja pai de uma thread não pode se comunicar com ela, a não ser que a thread seja em modo compartilhado.

O contexto global (objeto *window*) não existe em uma thread, no seu lugar o objeto *DedicatedWorkerGlobalScope* pode ser utilizado. Workers compartilhados podem utilizar o *SharedWorkerGlobalScope*. Estes objetos contém grande parte das funcionalidades proporcionadas pelo window com algumas exceções, por exemplo threads não podem fazer alterações no DOM.

4.18 OFFLINE

Não é extraordinário que um jogo tenha que ser reverso para um estado válido anterior por motivo de um erro na base de dados (Vahatupa 2014, pp. 5).

4.18.1 Aplicações offline

Na sua versão mais simples, um aplicativo offline é um conjunto de URLs para arquivos HTML, CSS, JavaScript, imagens ou qualquer outro tipo de recurso (Pilgrim 2010).

4.19 ENTRADA DE COMANDOS

Na construção da grande maioria dos jogos é muitas vezes imprescindível grande flexibilidade na gestão de entrada comandos. Esta necessidade amplia na criação de jogos multiplataforma, em determinadas plataformas a entrada de comandos pode-se dar través de teclado, em dispositivos móveis através tela sensível ou sensor de movimentos.

O HTML5 trata todos estes casos abstratamente na forma de eventos, os quais podem ser escutados através de *listeners*. Os eventos básicos são: keydown (tecla baixa), keyup (tecla solta) e keypress (tecla pressionada).

4.19.1 Acelerômetro

4.20 HTTP/2

HTTP/2 é a última verão do protocolo de trocas de documentos da WEB. Quando um cliente requisita algum documento de um servidor esta requisição é feita, geralmente, através do protocolo HTTP2.

A versão 2 trouxe grandes benefícios em potencial para performance, visto que o htt2 abre apenas uma conexão por host. Htt2 também não recomenda a utilização de minificação nos arquivos, o que antes era um benefício de performance agora pode ser um malefício. No contexto dos jogos onde o carregamento da tela inical pode demorar muito isso é um fator vital.

4.21 DEBUG

Com o inspetor do WebGL é possível conferir os estados dos buffers, informações de texturas, frames individuais e outras informações úteis (Kuryanovich et al. 2014).

4.21.1 Source Maps

Source Maps é uma tecnologia que permite mapear códigos fontes minificados para seus respectivos originais. Este recurso é interessante pois permite que os desenvolvedores

visualizem o código fonte em sua versão original, legível, enquanto entregam ao usuário final a versão minificada, optimizada para performance. Para o usuário final não há diferença pois Source Maps são carregados apenas se as ferramentas de desenvolvimento estão abertas e com a funcionalidade de Source Maps habilitada.

Source Maps foi desenvolvido como um trabalho em conjunto entre a Mozilla e Google em 2010, atualmente na terceira revisão o projeto é considerável estável e não recebe modificações na especificação desde 2013.

As ferramentas de desenvolvimento do Google Chrome e Firefox permitem a utilização de Source Maps.

É possível informar o navegador a localização do arquivo original seguindo a seguinte sintaxe.

```
//# sourceMappingURL=/path/to/script.JavaScript.map
```

Ou através de cabeçalhos HTTP como demostrado abaixo.

```
X-SourceMap: /path/to/script.JavaScript.map
```

Arquivos de Source Map contém metadados sobre os arquivos fonte e algumas outras configurações. Para gerar os arquivos é possível utilizar ferramentas como o https://github.com/mishoo/Ug

4.21.2 Debug 3D

Debugar aplicativos 3D pode ser complexo, o debugger do chrome é uma boa opção.

4.22 DISPONIBILIZAÇÃO DA APLICAÇÃO

Aplicativos baseados na web não requerem instalação ou atualizações manuais e sua distribuição é superior ao estilo convencional de aplicações desktop (Vahatupa 2014).

Links com manifestos

4.22.1 INSTALAÇÃO

Este método é benéfico pois possibilita ao usuário a mesma experiência ao adquirir uma aplicação normal. Este tipo de aplicação é comummente referido como "híbrido".

4.23 TRABALHOS SIMILARES

Barnett 2014 elaborou uma revisão de aspectos do HTML5 através da construção de um jogo. O autor foca muito nos aspectos de criação de jogos e feedback do desenvolvimento. Troca de tecnologias e não especificamente nas limitações conforme o meu trabalho. Em outras palavras seu escopo é mais genérico e não tão preciso quanto este

5 METODOLOGIA

O primeiro passo consiste em definir as plataformas alvo do trabalho. Estas devem ser relevantes mercadologicamente ao desenvolvimento de jogos em HTML5.

Segue-se com a construção de uma lista com os recursos relevantes aos jogos que, sofrem ou são comummente ligados à limitações multiplataforma. Segue-se uma pesquisa para aprofundar teoricamente cada um dos recursos, possivelmente elegendo novos.

Com um baseamento teórico substancial, o próximo passo é a criação do protótipo de um jogo multiplataforma que utilize recursos analisados.

Para capturar todas as limitações presentes, o protótipo deve ser construído sem a ajuda de frameworks ou bibliotecas de terceiros. Eliminando a possibilidade de a limitação ter sido tratada por terceiros.

Motores de jogos são bibliotecas que agregam várias funcionalidades usualmente úteis para o desenvolvimento de jogos (Vahatupa 2014, pp. 5). Elas podem incluir controle de usuário, cena, áudio, física, etc. Também servem como uma camada de segurança adicional (Vahatupa 2014).

Motores de jogos também foram descartados, visto que na maioria dos casos dependem de bibliotecas de terceiros. Outrossim, motores de jogos não são amplamente difundidos no mercado de jogos de navegadores (Vahatupa 2014).

Com o protótipo concebido, o passo que segue é a enumeração, e descrição das limitações detectadas no processo de desenvolvimento e testes do jogo. Este detalhamento deve responder as seguintes perguntas:

- Quais as limitações foram encontradas no jogo?
- Em quais plataformas?
- Sob quais circunstâncias?
- As limitações puderam ser contornadas?
- Algum efeito colateral das limitações no jogo?
- Qual a categoria do problema: usabilidade, funcionalidade, manutibilidade, portabilidade ou performance? (segundo ISO)

Como recomendação geral, busca-se abster-se da utilização de plugins pois estes muitas vezes escondem limitações do HTML em si.

6 PROJETO

Este capítulo tem por objetivo detalhar os aspectos do desenvolvimento do protótipo. Mecânica, o loop de eventos, diagramas entre outros aspectos do jogo serão abordados aqui.

6.1 MECÂNICA

Para a análise prática das limitações foi escolhido um jogo de matemática simples. O qual consiste na geração de equações com uma resposta candidata. Cabe ao usuário informar se o resultado apontado pelo jogo está correto ou não. A cada resposta dada o nível de complexidade da equação cresce. O tempo é um fator determinante no resultado do jogo pois quão mais rápido o jogador acertar se a afirmação está correta ou não mais pontos ele receberá.

Esta categoria de jogo foi selecionada pois oferece uma experiência interessante, tem profundidade - oferecendo a possibilidade de explorar diversos recursos do HTML5. Por ser de razoável tradutibilidade em tamanhos de telas diferentes e tipos de entrada de dados diferentes. E, por fim, por oferecer uma dificuldade técnica não tão desafiadora. Visto que não disponho de experiência profunda no desenvolvimento de jogos para HTML5.

Jogos como o Math Workout para Android tem uma temática similar. Não obstante, este jogo não sugere respostas para o usuário e o questiona sobre a veracidade delas, apresenta apenas a equação e requer uma resposta. O jogo aqui desenvolvido parece ter uma melhor jogabilidade para dispositivos móveis, pois não requer a presença de um teclado. Os botões de verdadeiro ou falso contém todas as possibilidades.

6.2 Requisitos

Abaixo estão dispostos os requisitos do sistema.

6.2.1 Requisitos funcionais

As funcionalidades que o sistema deve apresentar estão descritas abaixo.

- O sistema deve prover equações matemáticas de dificuldade crescente para o usuário informar se estão corretas ou não.

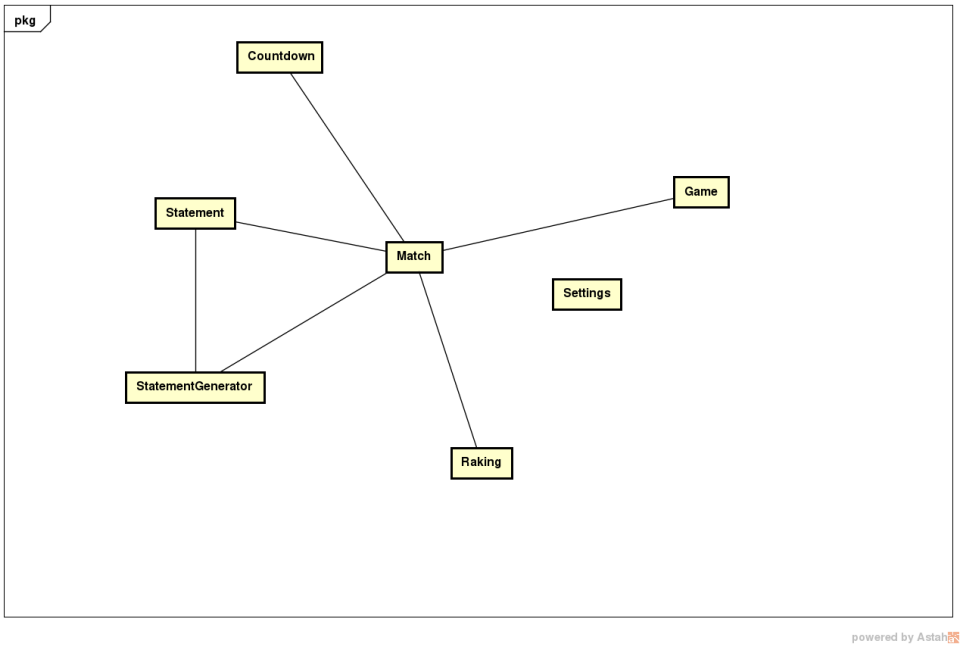

Fig. 6.1: Diagrama de classes simplificado

- O sistema deve pontuar as respostas dadas com maior agilidade com uma pontuação maior, que as respondidas com menor agilidade.
- O sistema deve apresentar um ranking com os resultados dos jogos anteriores.

6.2.2 Requisitos não funcionais

Outros aspectos requisitados mas que todavia não fazem parte da regra de negócio.

- O sistema deve ser desenvolvido utilizando as ferramentas da web.
- O sistema deve funcionar para a plataforma desktop e Android.
- O sistema deve ser desenvolvido sem a utilização de nenhuma biblioteca ou framework.

6.3 Modelagem

Abaixo segue o diagrama de classes simplificado.
Nos anexos pode-se encontrar a versão completa.

6.4 Desenvolvimento

O desenvolvimento se deu com a estratégia de melhoria progressiva. Desenvolvendo a versão mais simples possível para atingir as requisitos funcionais. Essa versão mais simples, apesar de não ter sido testado, deve funcionar na grade maioria dos navegadores.

O primeiro passo foi a criação do documento HTML. Utilizei divs para simbolizar telas do jogo. Caracterizando o jogo como uma aplicação SPA (*Single Page Application*). A alternativa seria depender de um servidor para mandar as páginas prontas, mas isso acarreta na necessidade de acesso à internet constante, coisa nem sempre viável em dispositivos móveis.

Seguindo a construção do markup, veio o desenvolvimento da classe Match, a qual simboliza uma partida dentro do jogo.

No início da construção não havia um gerador de equações, o jogo contava apenas com um vetor de equações pré-estabelecidas que eram selecionadas aleatoriamente a cada turno.

Isso se provou uma boa escolha pois possibilitou que o desenvolvimento se focasse em outros aspectos importantes como a elaboração do laço do jogo.

6.5 Laço do jogo

O laço do jogo (*game loop*) é peça central em praticamente qualquer jogo eletrônico. A cada iteração do jogo o laço é executado e o processamento de entrada de usuário e computações correlacionadas são executadas.

Para escrever o laço é possível utilizar as funções do JavaScript *window.settimeout*. Não obstante, a forma mais recomendada é utilizar o *window.requestAnimationFrame* reduz ou completamente para a execução do laço enquanto o usuário está em outra aba. Isso reduz o consumo de bateria, uma característica importante para dispositivos móveis.

O aspecto central do laço do jogo consiste em apresentar uma equação ao usuário com a possibilidade de ele responder se ela está correta ou errada.

Uma estratégia interessante que foi adotada na construção foi declarar todos os objetos relativos ao objeto window. Isso se demonstrou uma boa forma de separar os objetos, tornando o conflito de variáveis globais um problema irrelevante.

Outro aspecto positivo foi a utilização de um meta objeto para encapsular os demais, neste caso utilizei o nome MyMath, funcionando como um namespace, garantindo que problemas de conflitos de nomes não aconteçam.

Devido ao fato deste trabalho explorar as limitações dos jogos em HTML5, optei por evitar a utilização de plugins e ferramentas de terceiros que pudessem ocultar alguma limitação.

Escolhi a simplicidade para não precisar ficar muito tempo aprendendo as coisas em detrimento do refinamento da pesquisa.

Como armazenamento local foi optado por Web Storage por ser uma API simples, visto que os requerimentos do protótipo não demandam grande performance ou armazenamento massivo de dados a opção mais modesta foi preferida.

6.6 Otimizações para jogos

Navegadores tentam otimizar a experiência de navegação definindo um conjunto de regras e configurações razoáveis para a maioria dos casos. Não obstante, nem sempre estes valores

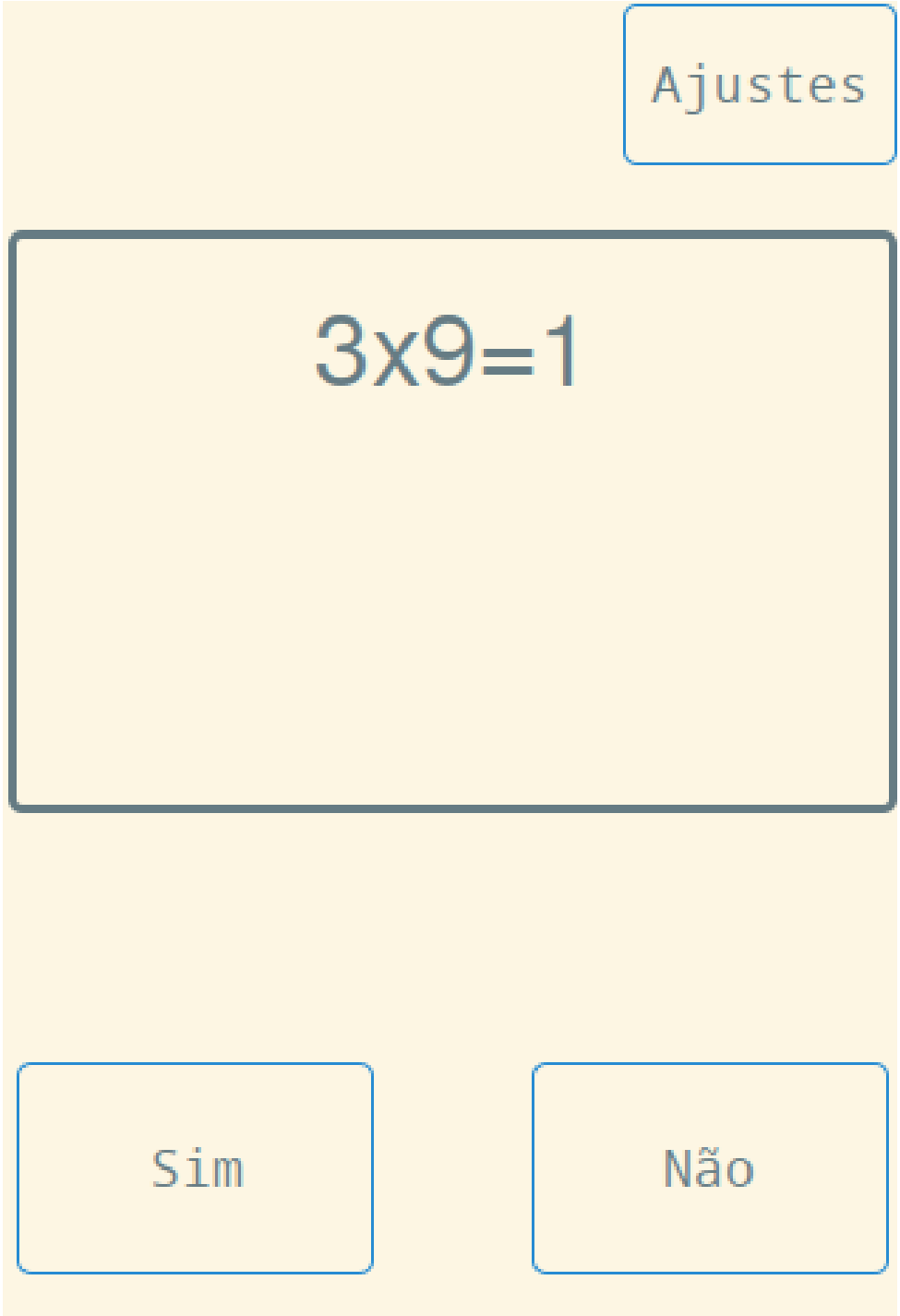

Fig. 6.2: Tabuleiro do jogo

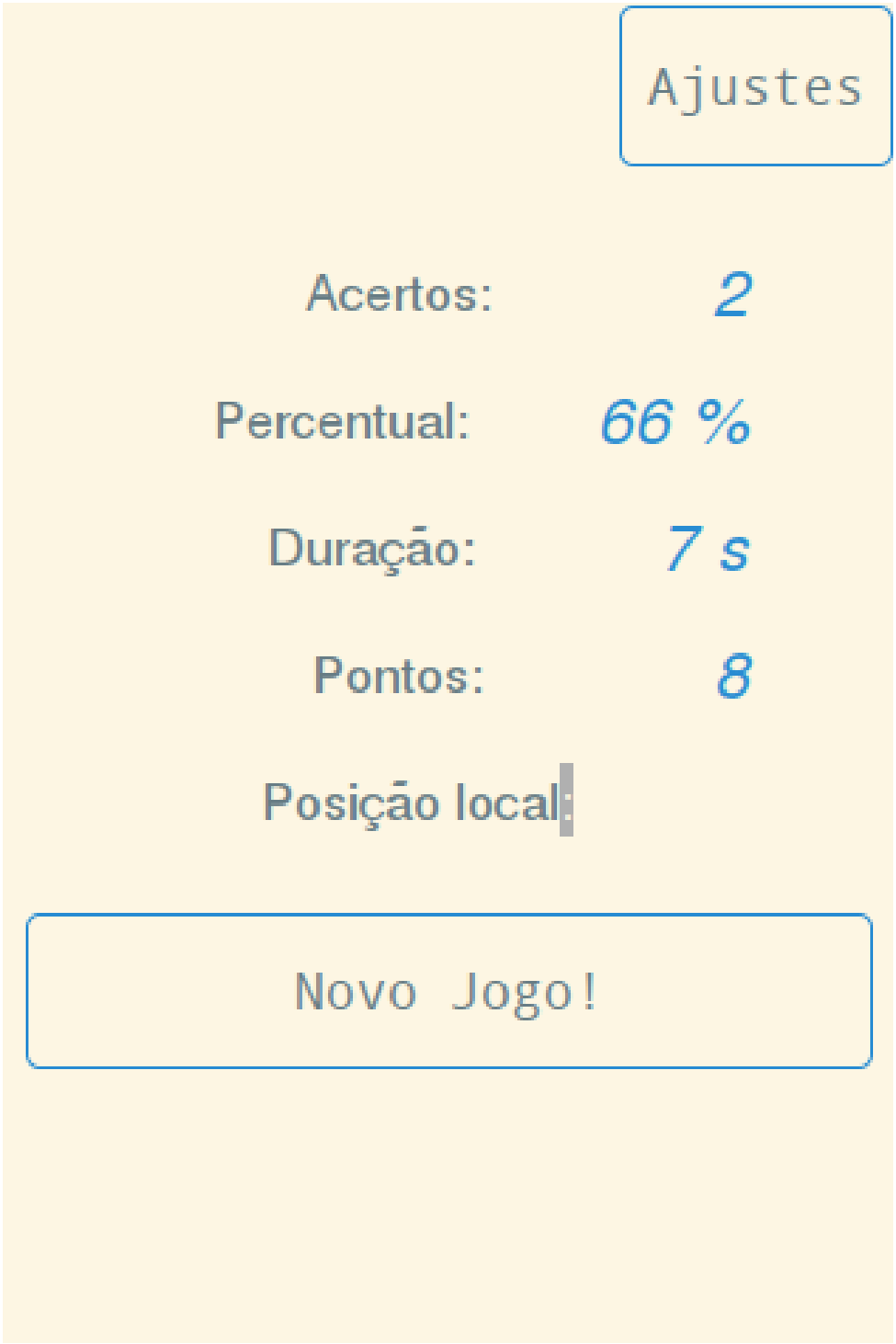

Fig. 6.3: Placar do jogo

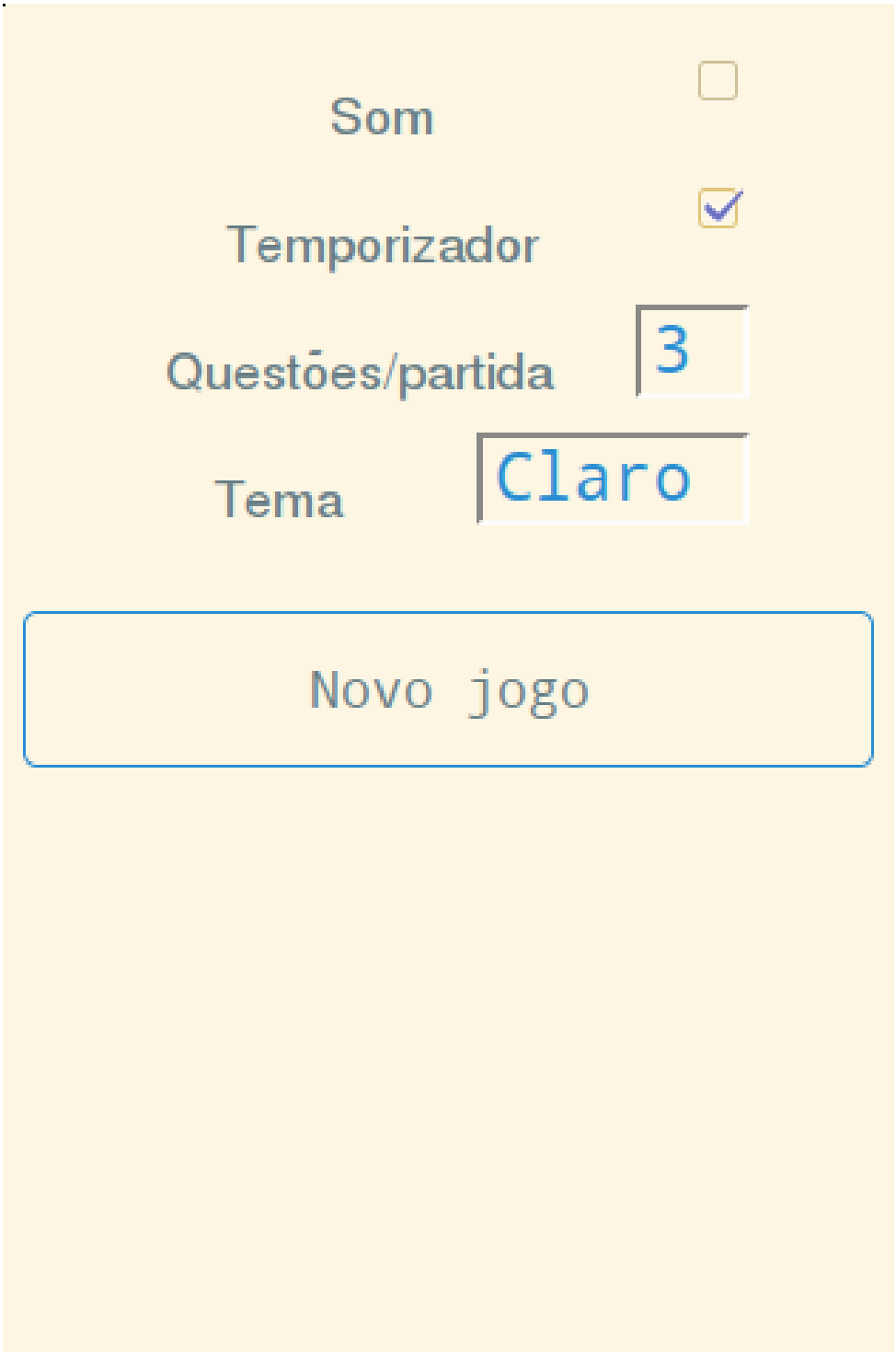

Fig. 6.4: Configurações do jogo

padrões são as melhores opções no contexto de jogos. Abaixo segue uma lista de configurações interessantes para se fazer com CSS no contexto de desenvolvimento de jogos. Abaixo seguem algumas otimizações que foram utilizadas no jogo, e são aplicáveis em grande parte dos jogos.

6.6.1 CSS

Scroll é um recurso interessante para longas páginas de texto, o mesmo não se pode dizer à respeito de jogos. Principalmente aqueles dependente de contato com a tela, pois no contato a tela pode se mover e desconcentrar o usuário. Para remover este comportamento deve-se utilizar o *overflow: hidden;* do seletor do corpo do documento (*body*).

A barra de endereço é outro recurso de pouca utilidade no contexto de jogos, e muitas vezes um empecilho para jogos em dispositivos móveis, devido ao limitado tamanho da tela.

Para desabilitar a barra em dispositivos da Apple pode-se utilizar a seguinte configuração:

```
<meta name="apple-mobile-web-app-capable" content="yes" />
```

Para os demais dispositivos não existe meio oficial de esconder a barra de endereço. Não obstante, alguns sites recomendam a solução descrita abaixo:

```
<body onload="setTimeout(function() {window.scrollTo(0, 1)}, 100)">
</body>
```

Apesar de não fazer parte da especificação, a maioria dos navegadores implementa a possibilidade de desativar a seleção de elementos na tela. Em jogos essa possibilidade é útil, pois não é natural a seleção de texto neste tipo de software. Kuryanovich et al. 2014 cita que desabilitar a seleção de texto em jogos é uma otimização importante para a experiência do usuário. Para desabilitar pode-se utilizar as regras CSS demonstradas abaixo.

```
-moz-user-select: none;
-webkit-user-select: none;
-ms-user-select: none;
```

6.6.2 JavaScript

Modo estrito

Algumas das recomendações nesta seção podem não se aplicar exclusivamente ao desenvolvimento de jogos. Outrossim são de grande relevância para a criação de sistemas de grande complexidade em geral, característica comum da grande maioria dos jogos.

Um recurso interessante do JavaScript é seu modo estrito, este faz um conjunto de modificação na semântica do interpretador de modo que alguns recursos suportados, mas propensos a problemas, sejam desabilitados. Um exemplo é variáveis sem o prefixo var.

```
(function() {
    'use strict';

    function bar() {
        return 'foo';
    }

    window.bar = bar;
})();
window.bar();
```

Fig. 6.5: Exemplo de utilização de funções imediatamente invocadas

O modo estrito pode ser entendido como uma variante mais rígida do JavaScript. O modo restrito pode ser habilitado utilizando o termo *"use strict";* nos cabeçahos de arquivos ou funções permitindo que código não estrito trabalhe em conjunto com código estrito, característica conveniente para a utilização em sistemas legados.

Funções imediatamente invocadas

Um problema comum de sistemas complexos em JavaScript é que muitos objetos vivem em ambiente global. Isso pode causar uma coleção de problemas, desde conflitos de nomes à sobrescrita de variáveis. Para contornar esse problema pode-se utilizar as funções imediatamente invocadas IFE (*Immediatly invoked function expression*).

A figura 6.5 demonstra a utilização deste padrão. As funções definidas no mesmo nível que bar não estarão no contexto global - a não ser que seja especificado diretamente - e não sofrerão conflitos de nomes e outros problemas relativos ao contexto global.

HTML

Um problema que jogos sofrem em geral é a demora no carregamento da grande quantidade de recursos que precisam estar em memória para o jogo funcionar. Muitos jogos utilizam uma tela de carregamento enquanto os recursos são adquiridos.

Um recurso do HTML interessante para este tipo de situação é o pré carregamento de recursos (*Link Prefetching*). Esta tecnologia possibilita que o navegador , em seu tempo livre, adquira recursos que provavelmente serão necessários em um futuro próximo.

Nem todos os recursos necessitam ser pré carregados, mas uma impressão muito superiora é criada se os recursos estão imediatamente disponíveis quando uma nova fase é carregada (Seidelin 2010, pp. 39).

7 RESULTADOS

Muitos dos problemas dos jogos multiplataforma não são específicos dos jogos, mas aplicam-se a todos tipos de software (Bruins 2014). Alguns problemas são inerentes das tecnologias, outras dos dispostivos ou da característica multiplataforma. Abaixo constam as limitações encontradas durante a pesquisa e concepção do jogo.

7.1 LIMITAÇÕES

Apesar da grande maioria dos recursos oferecidos nativamente nos dispositivos estar presente em HTML5 ainda existem algumas funcionalidades faltando para este tipo de aplicação.

«Smart Phones Application development using HTML5 and related technologies: A tradeoff between cost and quality» cita algumas limitações no contexto geral.

Não podemos mudar a imagem de fundo do dispositivo, ou adicionar toques etc. Similarmente, existem muitas API's de nuvem como os serviços de impressão do ICloud ou Google Cloud que estão disponíveis para aplicações nativas mas não para HTML5. Outros serviços utilitários como o C2DM do Google que está disponível para desenvolvedores Android para utilizar serviços de push também não estão disponíveis para o HTML5.

Vahatupa 2014 apresenta as seguintes limitações no contexto de jogos de navegador

- Não podem ser instalados em um dispositivo móvel como um aplicativo separado
- Não tem acesso a funcionalidades específicas dos dispositivos e recursos como notificações, use de hardware nativo ou comunicação entre aplicativos.

7.1.1 VERSÕES

A grande maioria dos dispositivos atualmente no mercado utilizam obsoletas de seus softwares. Isso dificulta o desenvolvimento. Se a tecnologia de tradução para o navegador utilizar o a classe Webview do Android - como o Apache Cordova faz - as versões mais antigas podem ser penalizadas com problemas de performance ou falta de recursos.

Segundo Barnett 2014

Enquanto o HTML é desenvolvido muitas das funcionalidades disponibilizadas são testadas em um pequeno conjunto de navegadores para um pequeno conjunto de versões. Isso acarreta em suporte inconsistente. A forma mais segura de garantir suporte é testando em todas as versões alvo, todavia essa solução não é prática.

7.1.2 OFFLINE

Refresh duplo para ver assets cacheados. Ver: http://buildnewgames.com/game-asset-management/

Quando o download offline falha o browser emite um evento mas não há indicação de qual problema aconteceu. Isso pode tornar a depuração tinda complicada que o usual (Pilgrim 2010).

7.1.3 SVG

Segundo Kuryanovich et al. 2014 a grande desvantagem do SVG é que quão maior o documento mais lenta a renderização.

7.1.4 Canvas

Os aspectos negativos do canvas é que a performance varia de plataformas para plataformas e não existe implementação nativa para animações (Kuryanovich et al. 2014).

7.1.5 WebGL

Como WebGL é baseada na versão otimizada para dispositivos móveis do OpenGL não é possível utilizar muitos recursos especiais disponível para os ambientes desktop.

Segundo Kuryanovich et al. 2014 um dos problemas do WebGL é sua alta curva de aprendizagem e o fato de não ter suporte para o Internet Explorer. Entretanto o suporte foi adicionado na última versão do Internet Explorer (11). Não obstante a dificuldade de utilização ainda persiste, forçando a maioria dos desenvolvedores a utilizarem abstrações criadas por bibliotecas de terceiros.

Para ver um programa em WebGL é necessário um navegador recente, uma placa gráfica recente e um sistema operacional que suporte a tecnologia (Kuryanovich et al. 2014)

7.1.6 VIDEO

7.1.7 ASSETS

Trafegar muitos assets deixa o sistema lento. Pode-se contornar este problema utilizando páginas de carregamento e/ou cache;

Outro problema relativo aos assets é descrito por Garisel e Irish 2011 scripts requerendo informações de estilo durante o processo de parsing. Se o estilo ainda não foi carregado o script vai utilizar informações erradas, causando uma série de problemas.

7.1.8 Codecs

7.1.9 ÁUDIO

Segundo Kuryanovich et al. 2014 a limitação do elemento de áudio do HTML5 é que seu propósito é para executar apenas um som, como o som de fundo dentro de um jogo.

A API de som no protótipo nem sempre emite o som quando o evento é disparado sucessivamente. Confirmando a afirmação de Kuryanovich et al. 2014 a API de som é boa se você deseja apenas tocar alguma música, mas se você está lançando eventos em um jogo ela é problemática.

Os navegadores variam na disponibilização de formatos aceitáveis Somente um áudio pode ser tocado no Navegador do Android

Não é possível trocar o volume no IOS.

Alguns navegadores favorecem formatos ogg (vorbis) e outros, como o Safari, favorecem o MP3.

Segundo «Smart Phones Application development using HTML5 and related technologies: A tradeoff between cost and quality»

> O maior problema com as API's de áudio e de vídeo do HTML5 é a disputa entre os codecs dos navegadores. Por exemplo, Mozilla e Opera suportam Theora, já o Safari suporta H.264 que também é suportado pelo IE9. Ambos, Iphone e Android suportam H.264 em seus navegadores. A W3C recomenda OggVorbis e OggTheora para áudio e vídeo respectivamente.

7.1.10 INTERFACE GRÁFICA

É muito custoso desenvolver uma interfaces que pareçam nativas para cada dispositivo sem a utilização de plugins e ferramentas especializadas. Em termos gerais, trabalhar com proporções é positivo. Não obstante há casos, como o dos botões de certo e errado que a proporções ficam exageradas, nesses casos a utilizada de max-width é uma solução conveniente.

7.1.11 PERFORMANCE

De acordo com uma pesquisa, para um usuário uma tarefa é instantânea se ele leva até 0.1 segundos para ser executada. Se a tarefa toma aproximadamente um segundo então a demora será notada mas o usuário não se incomodará com ela. Entretanto, se a tarefa leva

aproximadamente 10 segundos para terminar o usuário então começa a ficar aborrecido e esse é o limite que algum feedback deve ser dado para um usuário.

Otimizações de performance dependem do ambiente em que estão . sendo feitas E aquelas que hoje tem um impacto positivo hoje podem . se tornar inúteis, ou mesmo prejudiciais, amanhã (Kuryanovich et al. 2014, pp. . 131) .

7.1.12 Acelerômetro

7.1.13 IMPLEMENTAÇÃO INCONSISTENTE DE APIs

7.1.14 TAMANHO DE TELA

Em alguns casos o tamanho das telas pode ser um fator limitante – como no caso de jogos de estratégia. Jogadores com telas menores podem sair em desvantagem.

7.1.15 JavaScript

O JavaScript, por ser uma tecnologia desenvolvida por consenso, tem um ciclo de vida de atualizações demorado; pois necessita que todos os consumidores da especificação entrem em consenso e implementem a.

Apesar da performance ter notavelmente melhorado, ainda é geralmente menos eficiente produzir animações em JavaScript do que utilizando transições e animações do CSS, que por sua vez são mais otimizados e acelerados via hardware (Kuryanovich et al. 2014).

Erros numéricos resultam no valor NaN (*not a number*). Todas as operações com NaN como operadores irão retornar outro NaN. Isso torna a depuração de erros desnecessariamente complexa (Kuryanovich et al. 2014).

7.1.16 Detecção de recursos

Grande parte da detecção de funcionalidades é feita através de JavaScript, isso força os desenvolvedores a criarem pelo menos parte da marcação em JavaScript, isso pode ser um fator limitante para o uso generalizado de HTML5 (Pilgrim 2010).

Desktop/Firefox Desktop/Google Chrome Smatphone/Android

8 CONCLUSÕES

Apesar dos problemas mencionados neste trabalho, tomando uma perspectiva mais abrangente as tecnologias de desenvolvimento de jogos evoluiu muito e existem muitas coisas boas para se dizer à respeito.

A manipulação de erros é bastante consistente nos navegadores mas incrivelmente não faz parte da especificação (Garisel e Irish 2011). Não pude testar todos os métodos e ferramentas e versões à disposição, um trabalho completo demandaria esforços conjuntos de muitos indivíduos ou um período de tempo bem mais extenso.

Se uma empresa deseja produzir jogos nativos elas precisarão de vários desenvolvedores. Eu sozinho fui capaz de produzir um jogo em tempo razoável trabalhando com a plataforma WEB.

Por não utilizar frameworks e bibliotecas estou me distanciando dos casos da vida real.

Só poderemos considerar o HTML como uma especificação pronta quando for possível fazer tudo o que se faz nativamente com os dispositivos através de uma API WEB padronizada.

Muitas das limitações do HTML5 são contornáveis através de JavaScript.

O futuro dos jogos em HTML5 parece brilhante.

Neste trabalho revisamos tecnologias relevantes no desenvolvimento de jogos.

8.0.1 TRABALHOS FUTUROS

Trabalhos que explorem os benefícios mercadológicos do HTML5 em comparação com alternativas nativas. EMACSCRIPT 7.

HTTP2 também traz boas perspectivas em relação a performance.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Joshua Barnett. «Building a Cross-Platform Mobile Game with HTML5». Em: *University of East Anglia* (2014).

[2] Frederick P. Jr. Brooks. «What's real about virtual reality». Em: (1999).

[3] Stefan Bruins. «The current state of cross-platform game development for different device types». Em: (2014).

[4] Jon Duckett. *HTML and CSS - Design and Build Websites*, pp. 1–514.

[5] *ECMAScript 2015 Language Specification*.

[6] Tali Garisel e Paul Irish. «How Browsers Work: Behind the scenes of modern web browsers». Em: (2011).

[7] Isabela Granic, Adam Lobel e Rutger C. M. E. Engels. «The Benefits of Playing Video Games». Em: *Radbound University Nijmegen* (2014).

[8] Yousuf Hasan et al. «Smart Phones Application development using HTML5 and related technologies: A tradeoff between cost and quality». Em: ().

[9] Ian Hickson. *HTML is the new HTML5*. Available at `https://blog.whatwg.org/html-is-the-new-HTML5` (2015/11/11).

[10] *HTML HyperText Markup Language*.

[11] Marek Janiszewswi. Tese de doutoramento. Vrije Universiteit Amsterdam, 2014.

[12] Egor Kuryanovich et al. *HTML5 Games Most Wanted: Build the Best HTML5 Games*. 2014.

[13] David de Oliveira Lemes. Tese de doutoramento. Pontifícia Universidade Católica de São Paulo, 2009.

[14] Håkon Wium Lie. «Cascading Style Sheets». Tese de doutoramento.

[15] Anttonen Matti e Salminen Arto. «Building 3D WebGL Applications». Em: (2011).

[16] Belchin Moises. *ECMAScript 6 Today*.

[17] Mark Pilgrim. *Dive into HTML5*. 2010.

[18] Axel Rauschmayer. *A first look at what might be in ECMAScript 7 and 8*.

[19] Jacob Seidelin. *HTML5 Games - Creating fun with HTML5, CSS3, and WebGL*. 2010.

[20] *Selectors*. Available at `http://www.w3.org/TR/CSS21/selector.html` (2015/11/17).

[21] *Using CSS transitions*.

[22] Juha-Matti Vahatupa. «On the development of Browser Games - Current Technologies and the Future». Em: (2014).

[23] Yu Zhang. «Developing Effect of HTML5 Technology in Web Game». Em: *International Journal on Computational Sciences and Applications (IJCSA)* (2012).

A CONVERSORES PARA HTML5

Além da possibilidade de escrever em HTML, pode-se optar pela alternativa de utilizar-se um conversor de linguagens.

A.0.1 Bibliotecas relevantes do desenvolvimento de jogos

Desenvolvedores da WEB são geralmente de mente aberta e desenvolveram uma variedade de bibliotecas e frameworks pela internet (Kuryanovich et al. 2014).

Seidelin 2010 ressalta a importância de sermos moderados quanto a escolha de bibliotecas no contexto WEB multiplataforma.

> Muitos desenvolvedores da WEB utilizam bibliotecas como jQuery e o Prototype de modo que se vejam livres de terem que lidar com partes triviais do desenvolvimento WEB, como selecionar e manipular elementos do DOM. Muitas vezes essas bibliotecas incluem várias funcionalidades que não são utilizadas. É recomendável cautela para verificar se realmente é necessário adicionar 50-100k de bibliotecas, ou se alguma coisa mais simples e menor não trará os mesmo benefícios, especialmente quando desenvolvendo multiplataforma onde uma rápida conexão a internet nem sempre é garantida.

Esta preocupação aumenta no contexto de desenvolvimento de jogos. Ainda segundo Seidelin 2010 o site MicroJS https://microjs.com oferece uma coleção de micro bibliotecas focadas em áreas particulares em detrimento de grandes bibliotecas cheias de funcitnalidades.

A.1 Frameworks de jogos

Com o intuito de simplificar o processo para os desenvolvedores, auxiliando-os a focarem-se apenas nas soluções que estão desenvolvendo, foram criados os frameworks para desenvolvimento de jogos. Alguns frameworks reconhecidos são:

- enchant.js: dentre suas funcionalidades constam: orientação à, orientado à eventos, contém um motor de animação,
- suporta WebGL e Canvas, etc three.js: considerada leve, renderiza, WebGL e Canvas, arquitetura procedural

- limeJs: bom para 2d
- quintus: especialista em jogos de plataforma 2D

A.2 CROSSWALK

Crosswalk empacota os fontes juntamente com uma versão do Chromium, a versão Open-source do Google Chrome. Isso faz com que o software se comporte da mesma forma para todas as versões de dispositivos Android.

A.3 PHONEGAP

A.4 PHONEGAP CLOUD

Este serviço possibilita que se faça upload de um arquivo compactado contendo os fontes – ou apontando para um repositório no GitHub – que no tempo desta pesquisa não estava funcionando; e se gere o APK para o Android nativamente.

A.5 NODEJS

Permite rodar JavaScript fora do navegador. Utiliza um modelo dirigido à eventos sem bloqueio, tornando-o rápido e eficiente.

B ALTERNATIVAS AO JAVASCRIPT

Abaixo seguem algumas tecnologias que servem de alternativa ao JavaScript.

B.1 TYPESCRIPT

Conhecido como uma versão estendida do JavaScript que compila para JavaScript normal.

B.2 DART

Google. DartVM é uma máquina virtual que está embebido no Google Chrome. Significante melhorias em performance quando comparado ao JavaScript. Existe o dart2js que compila código em Dart para JavaScript.

C SISTEMAS DE BUILDING

Aquivos JavaScript são requisitados do servidor assincronamente. Isso pode levar a tempos de requisição pouco desejáveis. Uma saída seria escrever o código em apenas um arquivo mais isso leva a gerência de código bagunçada. A saída mais comum entre desenvolvedores é utilizá ruma ferramenta que junta todos os arquivos e disponibiliza apenas um para o usuário.

Utiliza o conceito de streams para aplicar todas as modificações sobre um arquivo de uma vez só.

D AMBIENTES PARA DESENVOLVIMENTO HTML5

Na pesquisa efetuada sobre estes frameworks full-stack foram identificadas as seguintes tecnologias:

Segundo (PRADO, 2012) o GWT é um framework essencialmente para o lado do cliente (client side) e dá suporte à comunicação com o servidor através de RPCs Remote Procedure Calls (ou procedimento de chamadas remotas). Ele não é um framework para aplicações clássicas da web, pois deixa a implementação da aplicação web parecida com implementações em desktop. Este é utilizado em muitos produtos de grande porte como o Google Adwords e Google Wallet. Outra característica interessante é que a plataforma opera sobre a licença Apache versão 2;

Construct 2 - é um editor na nuvem focado para usuários sem - conhecimento prévio em programação orientado a comportamento; - PlayCanvas - é uma plataformas para a construção de jogos 3D na nuvem, desenvolvida com foco em performance. Permite a hospedagem, controle de versão e publicação dos aplicativos nela criados, possibilita também a importação de modelos 3D de softwares populares como: Maya, 3ds Max e Blender;

- o ambiente HTML5 da Intel, este fornece uma solução na nuvem, completa para o desenvolvimento em plataforma cruzada, com serviços de empacotamento, serviços para a criação e testes de aplicativos com montagem de interfaces puxa e arrasta (Intel XDK) e bibliotecas para a construção de jogos utilizando aceleração de hardware, o que garante até duas vezes mais performance que aplicativos mobile baseados em Web tradicionais. Esta solução é gratuita, open-source e funciona através de um plugin para o Google Chrome, ou seja, o desenvolvimento também é multiplataforma e devido ao fato de os binários ficarem hospedados na nuvem, possibilitou a Intel criar compiladores para cada uma das plataformas disponibilizadas pelo PhoneGap, que é o framework polyfill utilizado na solução.

E METODOLOGIA DE DESENVOLVIMENTO DE SOFTWARE PARA A CONSTRUÇÃO DE GAMES

Como o jogo é um software complexo demanda-se a utilização de metodologias de engenharia de software, dentre os processos de software mais conhecidos academicamente destacamos:

- OpenUP: este é bem detalhado e de característica iterativa e incremental. Gerando assim, um levantamento mais apurado dos riscos, requisitos e outros detalhes do sistema e a criação incremental do sistema, com requisitos maleáveis;
- Cascata: processo antigo, caracteriza-se por ser pouco maleável aos requisitos mapeados posteriormente ao processo de análise;
- Processo ágil - SCRUM: sua utilização é flexível e sendo um método ágil especifica pouca documentação, ou como dizem, somente a documentação necessária, este processo é bem conhecido e aceito na comunidade de desenvolvimento de software. Suas principais características são: divisão do processo de desenvolvimento através uma série de iterações chamadas sprints. Cada sprint consiste tipicamente em duas a quatro semanas. É bem aplicado a projetos que mudam constantemente e que demandam rápidas adaptações;
- Processo ágil – XP: tem muitas características similares ao SCRUM por este também ser um processo ágil. Dentre suas especifidades destaca-se: versões frequentes, pequenos ciclos de desenvolvimento que buscam aumentar a produtividade, introduzem checkpoints onde os clientes podem agregar novas funcionalidades;

F Diagrama de classes

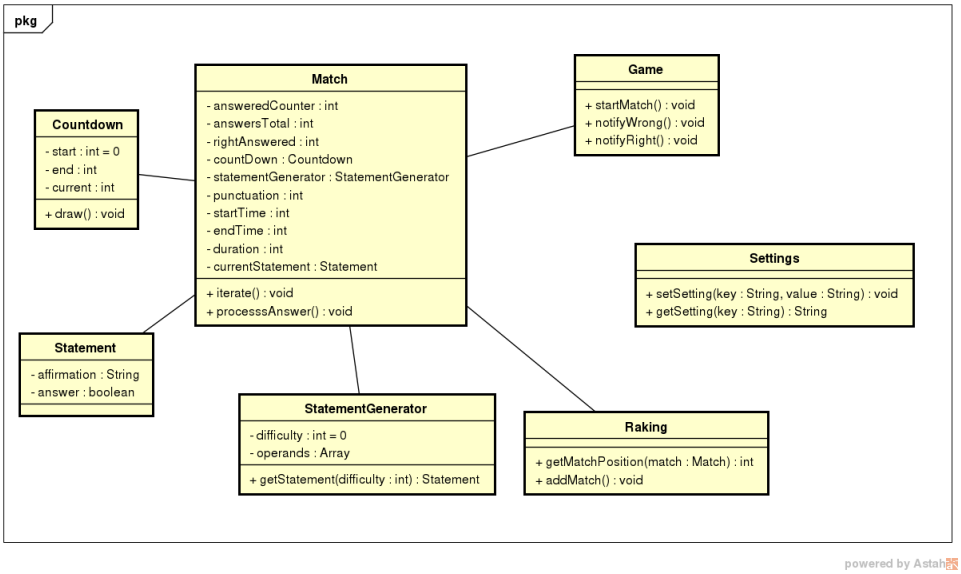

Fig. F.1: Diagrama de classes completo