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

SUMÁRIO

1 CONTEXTUALIZAÇÃO

1.1 JOGOS

1.1.1 BENEFÍCIOS

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

A maioria dos desenvolvedores demonstra interesse para o HTML5. (referenciar)

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. (HASAN et al, 2012)

1.1.2 O MERCADO

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. (HASAN, 2012)

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

1.1.3 JOGOS E MULTIPLATAFORMA

1.1.4 HTML E MULTIPLATAFORMA

1.1.5 LIMITAÇÕES DE JOGOS MULTIPLATAFORMA COM HTML5

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)

Enquanto o HTML é desenvolvido muitas das funcionalidades disponibilizadas são testadas em apenas um pequeno conjunto de navegadores para um pequeno conjunto de versões (referência 2). 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. (ref. 2)

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

1.2 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 HTML5 - aplicado ao desenvolvimento de jogos. O senso comum, acostumado com soluções nativas, acabam por monopolizar à construção de jogos nativos.

Os custos introduzidos no ciclo vida de um jogo, para diversas plataformas, é muito alto para ser considerado trivial. Cerca de 65mais altos (segundo trabalho 2)

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, c omo Flash Player, Silverlight ou alternativas Desktop.

2.1.2 OBJETIVOS ESPECÍFICOS

Estudar as limitações de desenvolvimento de jogos nas plataformas Android e navegadores Desktop Google Chrome e Firefox. Os navegadores foram testados 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.

Construir um protótipo que colabore para a fundamentação prática do assunto.

Pretende-se também estudar os seguintes tópicos do desenvolvimento de jogos, relativos ao HTML5:

Não se pode detectar suporte ao HTML5, isso não faz sentido. Pode-se detectar suporte as funcionalidades individuais como canvas, vídeo e geolocalização (Pilgrim 2010).

Vahatupa 2014 cita que video, audio, drag and drop, funcionalidades de gráficos e armazenamento off-line são aspectos importantes do HTML5 para o desenvolvimento de jogos.

Com base nos estudos efetuados e na experiência adquirida elencamos os seguintes itens que serão abordados neste estudo.

- Performance

- Diferenças em tamanho de tela
- Canvas
- Empacotadores HTML5
- Eventos de entrada
- Vibração
- Armazenamento
- Disponibilização de assets (controle de tamanhos, cache, etc)
- Aplicações offline
- CSS media queries

Elaborar uma lista de limitações e correlacionar os dados de acordo com as plataformas.

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.

Tem potencial apontar os pontos chave que necessitam de melhorias no HTML, consequentemente colaborando para a melhoria da própria especificação. Estimular e avançar o estudo da implementação da Open Web;

A opinião comum tende para soluções nativas em detrimento do desenvolvimento de jogos, este trabalho pretende desafiar esta concepção.

Muitos desenvolvedores estão familiarizados com as tecnologias da WEB ou apontam interesse na tecnologia.

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.

Video games melhoram as funções cognitivas, melhoram as capacidades criativas, e motivam uma visão positiva diante a falha (Granic, Lobel e Engels 2014).

A natureza dos jogos tem mudado drasticamente na última década, se tornando cada vez mais complexos, diversos, realisticos e sociais em sua natureza (Granic, Lobel e Engels 2014).

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).

4.1.1 Jogos Web

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, sendo assim, pode-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.

4.1.2 GÊNEROS

Uma classificação de jogos em gêneros é uma tarefa complexa. Segundo (Granic, Lobel e Engels 2014):

Segundo Granic, Lobel e Engels 2014, pp. 60:

> Pela diversidade em itens e gêneros e a vasta quantidade de dimensões que os video games se encontram, uma taxonomia dos jogos contemporâneos é extremamente difícil de desenvolver (muitos já tentaram) .

Não obstante alguns padrões são discerníveis quanto aos jogos da web.

Os primeiros jogos eram limitados a tecnologias presentes, um gênero de que se desenvolveu bem foram os jogos de estratégia como o Travian.

Com as novas possibilidades do tecnológicas novas possibilidades e gêneros de jogos podem ser explorados na web.

4.1.3 MECÂNICA

A mecânica é composta pelas regras do jogo. Quais as ações disponíveis aos usuários, é fortemente influenciada pela categoria do jogo em questão.

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 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 os 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, pg XX) 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.

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 mais lentos que os 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*), 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.

Mais do que um conjunto de tecnologias Open Web, é um conjunto de filosofias as quais a web se baseia.

Neste conjunto inclui-se:

- Descentralização;

- Transparência;
- Relevância;
- Imparcialidade;
- Consenso;
- Disponibilidade;
- Manutibilidade;

A W3C é a empresa responsável por grande parte das especificações da web como: HTML (em conjunto com a WHATWG), CSS, entre outros.

O processo da W3C consiste na elaboração de rascunhos de (*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 ser rigoroso, está longe de perfeito. A especificação final do HTML4 contava com quatro erros publicados via errata (**HTML5**).

Outras empresas estão envolvidas nas especificações da web, como a ECMA, responsável pelo JavaScript; Kronos, responsável pelo WEBGL, etc.

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: "Hipertexto", refere-se a links que conectam páginas 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 mais 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 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. 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

```
<}!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

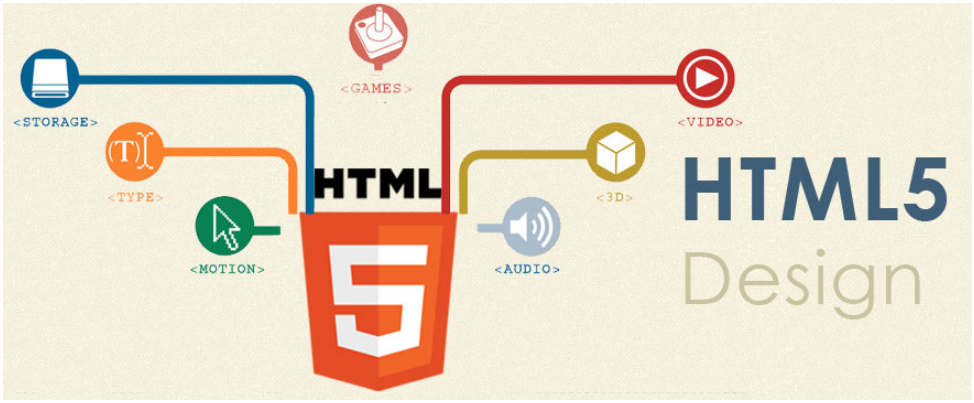

Fig. 4.2: Suíte HTML

elemento meta (<meta>) tem um atributo *charset*, que especifica o formato de codificação do documento.

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, HTML5 ou "HTML5 e amigos"é muitas vezes interpretado como um conceito guarda chuva para designar as tecnologias da web: HTML, CSS3 e JavaScript.

```
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

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 do DOM permite alto nível de precisão e performance para buscar elementos.

A figura 4.3 demonstra a utilização dos seletores do DOM em um documento 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 é peça central no desenvolvimento WEB tornando-se parte da OpenWeb, 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 do CSS, a estilização das páginas era inclusa dentro do HTML. Isso forçava que mudanças tivessem que ser replicadas em todas suas ocorrências, e não permitia apresentações diferenciadas para diferentes tipos de dispositivos.

A diferenciação entre apresentação e estrutura, sendo neste caso o CSS responsável pela apresentação, é um dos pontos chave do SGML, motivo que tornou a utilização do CSS tão difundida.

Com CSS também é possível que o usuário declare suas próprias folhas de estilo, um recurso importante para acessibilidade.

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

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

Fig. 4.4: Exemplo de Folha de Estilo

mesmo estilo em várias publicações, segundo pois os publicadores podem focar-se no conteúdo ao invés de se preocuparem-se com detalhes de apresentação.

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.

CSS é dividido em módulos, que representam conjuntos de funcionalidaes, contendo aproximadamente 50 deles, cada qual evoluindo separadamente. Além do módulos, CSS também é organizado por perfis e níveis.

Os perfis do CSS dividem a especificação por dispositivo de utilização. Existem perfis para dispositivos móveis, televisores, impressoras, etc. Já os níveis dividem 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 mais 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 CSS2.1, Seletores do CSS3 e Texto do CSS3 de recomendações para rascunhos.

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.

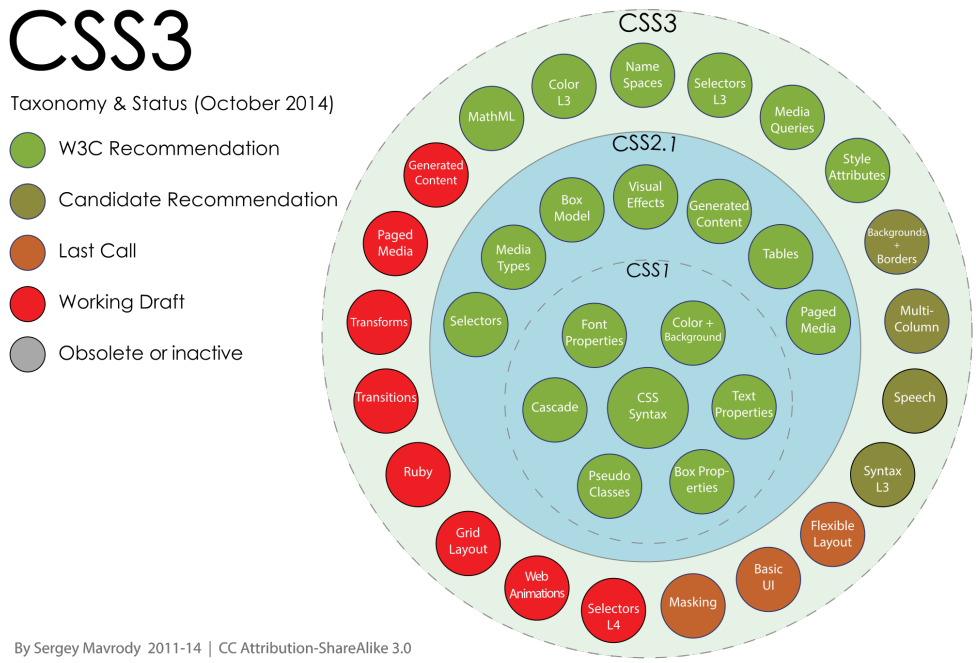

Fig. 4.5: The modules of CSS, 2014

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

Fig. 4.6: Exemplo de media query

A especificação prevê a possibilidade de aplicar seletores entro de arquivos CSS, ou aplicar arquivos inteiros dependentemente das configuração do aparelho em questão.

A figura 4.6 demostra a aplicação de uma regra via seletor Media query, aplicando o background apenas para dispositivos com no minimo 1024 px de resolução.

4.5.2 Transições

É 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 a trasinção. Habilitando o desenvolvedor a controlar o tempo de execução de uma tranisção e seus estados intermediários.

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

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.

A transição demonstrada na figura 4.7 escala o tamanho do elemento com a classe (*test*)

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

Fig. 4.7: Exemplo de transição

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.

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 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.

4.5.3 Otimizando o CSS para jogos

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 websites 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. Para desabilitar a seleção pode-se utilizar as regras CSS demonstradas abaixo.

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

Recursos muito 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 mais 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

	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. 4.8: Propostas do ECMA 7

Fig. 4.9: *
Fonte: https://github.com/tc39/ecma262

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 4.9 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. No contexto dos jogos performance é um fator de extrema importancia. ASM.JS se destaca por utilizar recursos que permitam otimizações antes do tempo (*ahead of time optimizations*). Entretanto, não é trivial escrever código em asm.js e geralmente a geração de código asm.js é feita através da conversão de outras linhagens como C.

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.

4.7 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.

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

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 mais 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/.

4.8 NAVEGADORES

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.

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 4.12 apresenta o gráfico de suporte por versões de navegadores em dezembro de 2015.

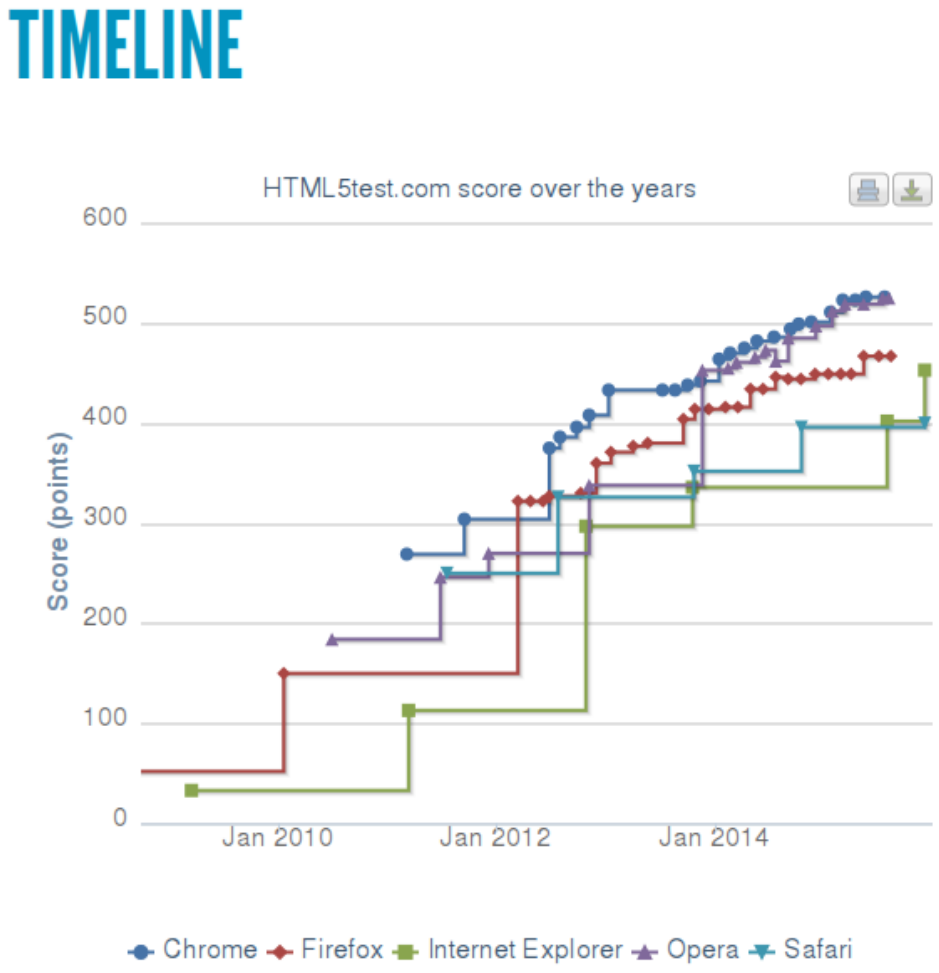

Fig. 4.10: Suporte das especificações do HTML nos navegadores

4.9 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.

4.10 RENDERIZAÇÃO

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

4.10.1 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;
- 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).

4.10.2 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.

```
<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. 4.11: Círculo em SVG.

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.

Apesar de ser largamente 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.

https://github.com/google/canvas-5-polyfill Esta serve para suprir aos navegadores que ainda não implementaram, os últimos recursos da especificação CANVAS, como o recurso de desenhar elipses e o caminho 2D.

4.11 WEBGL

É Baseado em OpenGL, o órgão que especifica o WEGGL é o mesmoq que especifica o OpenGL: Kronos.

WebGL não foi utilizada no trabalho apesar de ser de grande relevância no processo de jogos pois ainda não está completamente especificada e a dificuldade e escopo do projeto aumentariam muito se tivessem de incluir um jogos 3D. Versão da especificação atual?

CocoonJS é uma aplicativo híbrido que preenche a fraca implementação de OPENGL nos dispositivos móveis possibilitando se desenvolver em WEBGL.

4.12 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.

4.13 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).

4.13.1 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.

4.13.2 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

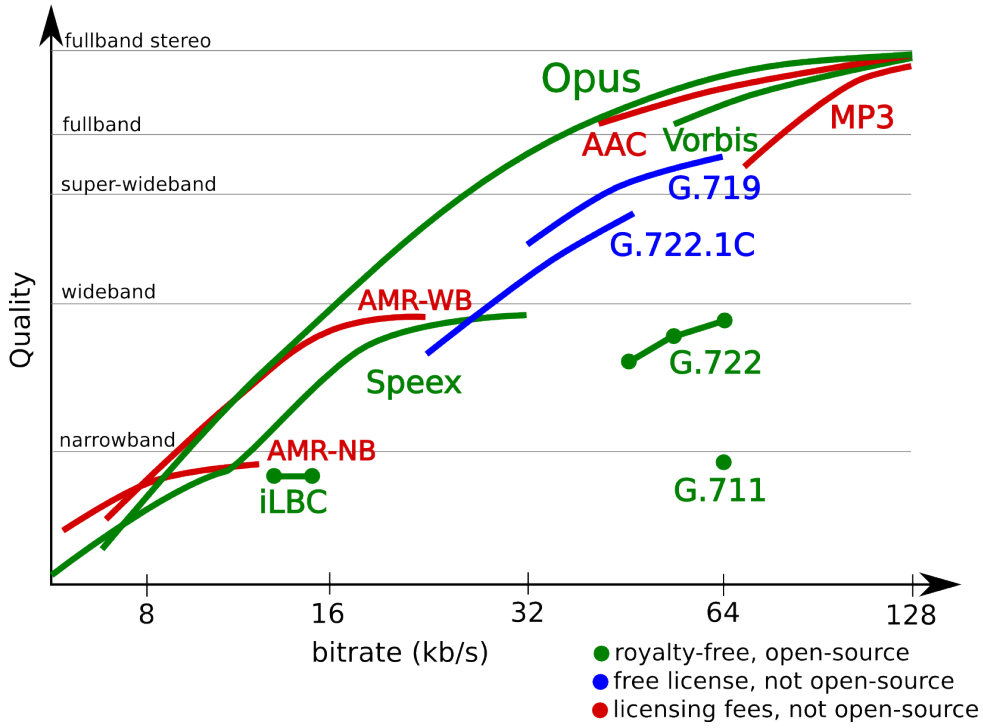

Fig. 4.12: Comparação de codecs de áudio

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

A figura 4.12 é um comparativo interessante sobre os formatos existentes, considerando o fator taxa de bits versus qualidade.

4.14 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

4.15 OFFLINE E ARMAZENAMENTO

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).

Uma das grades limitações do HTML era a ausência de capacidade de armazenamento de dados. Armazenamento no lado do cliente é um requerimento básico para qualquer aplicação moderna. 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»).

4.15.1 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).

4.15.2 LOCAL STORAGE

Também conhecido como Web Storage na especificação do HTML5. Provê uma forma de armazenar os dados como chave valor dentro do navegador. Os dados são persistido mesmo que o navegador seja fechado.

4.15.3 WEB SQL

Um banco de dados SQLite embebido no navegador. Permite tabelas relacionais. O tamanho padrão do banco de dados é 5 megabytes e pode ser estendido pelo usuário.

Segundo Pilgrim 2010

> Todos os implementadores interessados 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 descriação do dialeto SQL apenas referencia o SQLITE, o que não é aceitável para um padrão.

Web Sql foi depreciado em favor da especificação do IndexedDb. Não obstante, IndexedDb ainda é comummente utilizado em sistemas móveis (*HTML5 Test, how well does your browser support HTML5?*).

4.15.4 IndexedDb

4.16 ENTRADA DE COMANDOS

Na construção da grande maioria dos jogos é muitas vezes imprescindível alta flexibilidade na gestão de entrada de dados. Este fator muito se amplia na criação de jogos multiplataforma, seja através de teclado, tela sensível ou sensor de movimentos, o importante é oferecer a melhor experiência possível por plataforma. 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).

4.16.1 DISPONIBILIZAÇÃO DA APLICAÇÃO

Links com manifestos

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

4.17 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".

4.18 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).

Neste trabalho, 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

Para a análise prática das limitações foi escolhido um jogo de matemática simples. Consistindo na geração de equações com um candidato de resposta. Cabe ao usuário informar se o resultado apontado pelo jogo está correto ou não.

Porquê escolhi esse tipo de jogo?

Não tenho grande experiência com o desenvolvimento de jogos nem com o desenvolvimento em HTML5. Também para não interferir na pesquisa busquei não me distanciar do que é considerado padrão em ferramentas e métodos. Comecei escrevendo o aplicativo para o Navegador do desktop pois era o que estava mais acessível no momento. Mais tarde descobri que de fato é assim que de desenvolve.

6.1 MECÂNICA

O jogo consiste em simplesmente em uma tela que apresenta equações e um possível resultado. Cabe ao jogador decidir se o resultado está certo ou errado. O tempo é um fator levado em consideração, quão mais rápido o jogador acertar se a afirmação está correta ou não, mais pontos ele receberá.

Argumentos à favor da escolha do game: Tem profundidade, permite a adição de novos recursos no futuro; É facilmente traduzível em tamanhos de telas diferentes e tipos de entrada de dados diferentes;

6.2 Requisitos

6.3 Modelagem

6.4 Desenvolvimento

O loop do jogo deve executar as seguintes ações...

Finalizada a revisão bibliográfica será descrito os detalhes da implementação do jogo.

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.

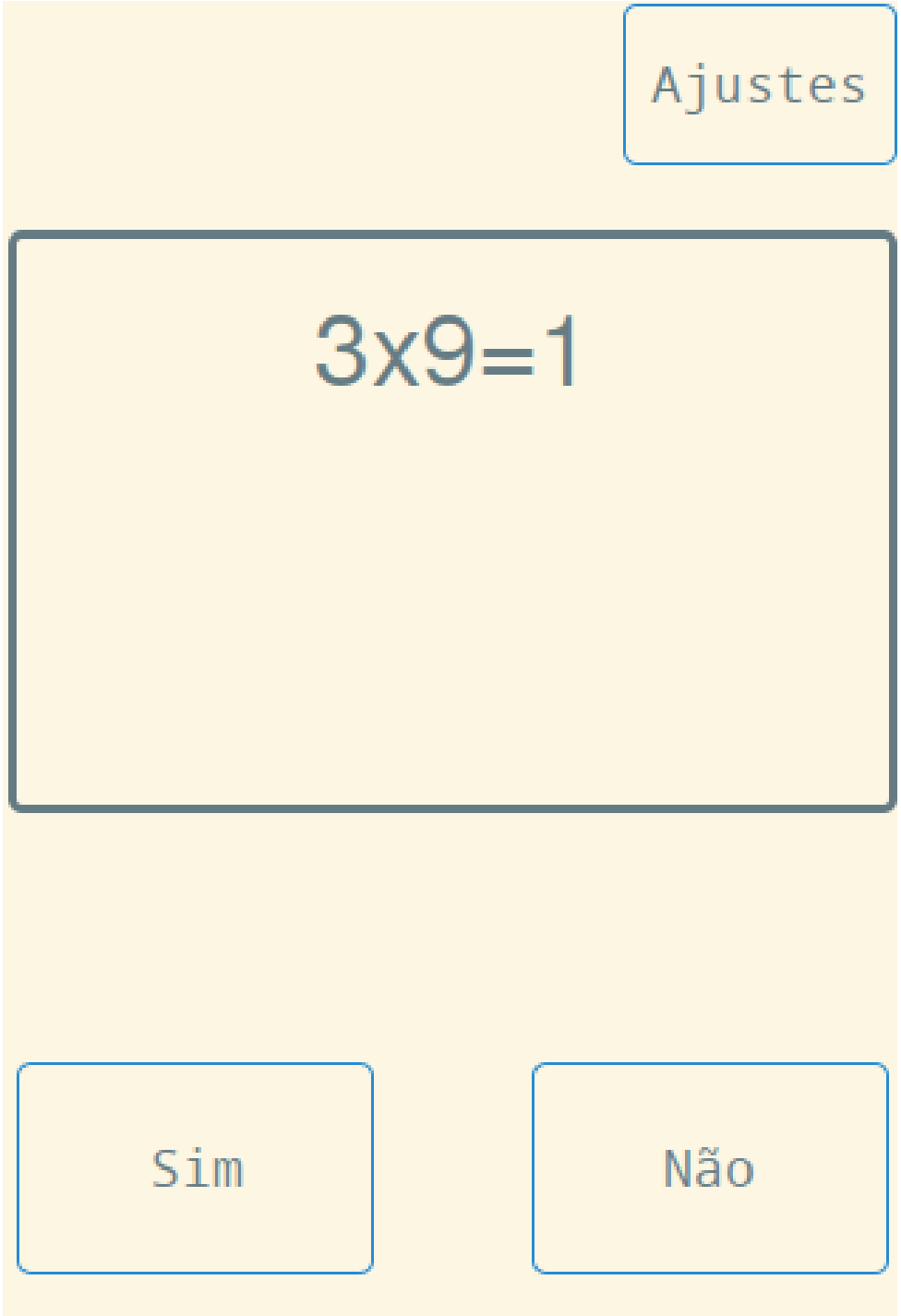

Fig. 6.1: Tabuleiro do jogo

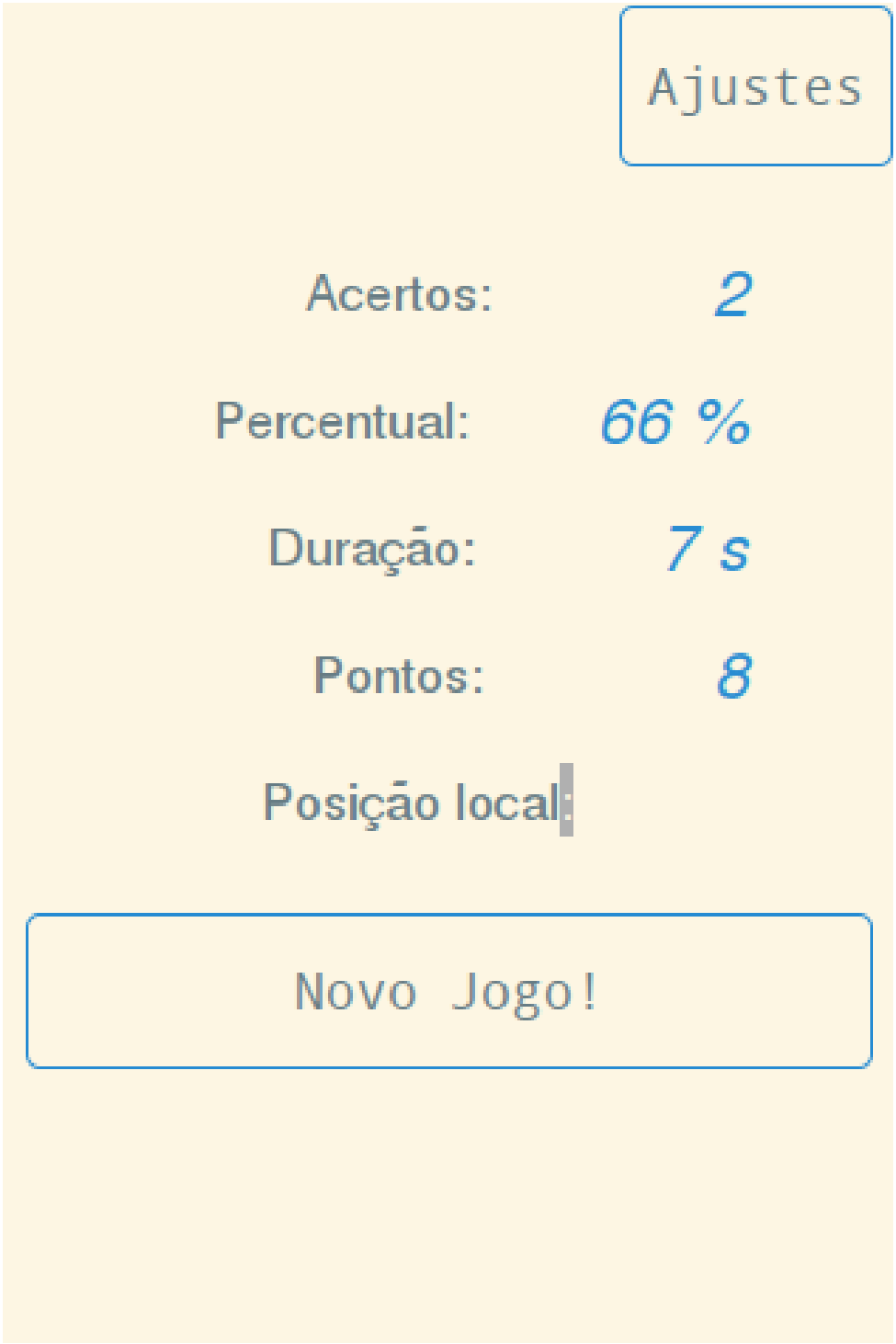

Fig. 6.2: Placar do jogo

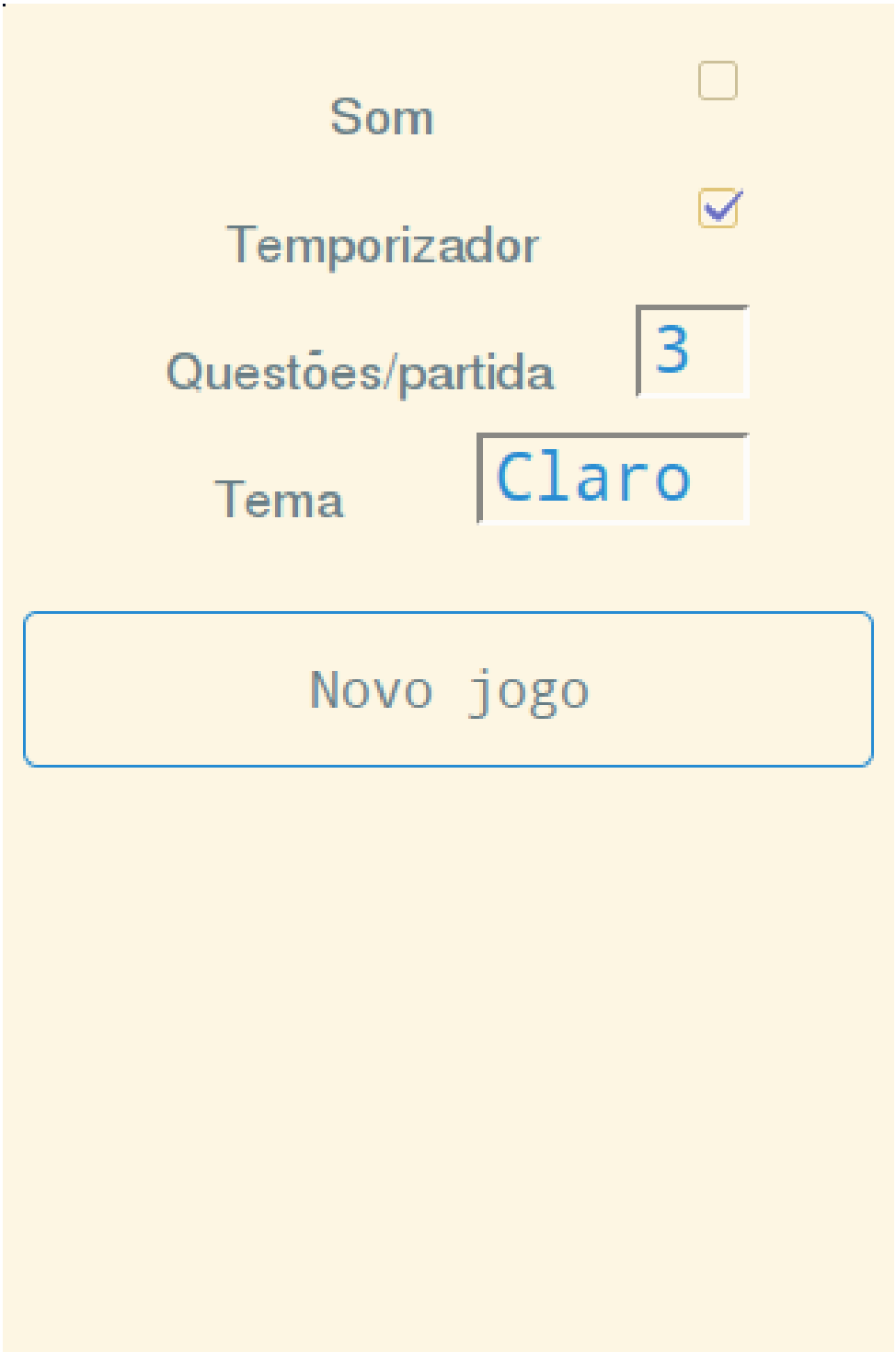

Fig. 6.3: Configurações do jogo

7 RESULTADOS

Durante a construção do jogo utilizei a estratégia de declarar todos os objetos relativos ao window e limitar o escopo. Isso se demonstrou uma boa forma de separar as responsabilidades.

Abaixo constam as limitações encontradas durante a pesquisa e concepção do jogo.

Muitos dos problemas dos jogos multiplataforma não são específicos dos jogos, mas aplicam-se a todos tipos de software (Bruins 2014).

7.1 LIMITAÇÕES

Apesar da grande maioria dos recursos dos dispositivos estar presente em HTML5 ainda existem muitas funcionalidades faltando para este tipo de aplicação. Por exemplo, não podemos mudar a imagem de fundo do dispositivo, ou adicionar toques etc. Similarmente, existem muitas APIs 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. (HASAN, 2012)

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.

7.1.2 OFFLINE

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

7.1.3 AUDIO

Api de som quebra quando executado diversas vezes. 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.

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. (HASAN et al, 2012)

7.1.4 SVG

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

7.1.5 VIDEO

Codecs

4. ASSETS

Trafegar muitos assets deixa o sistema lento.

Contorno Utilizando páginas de carregamento e/ou cache;

5. UI

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.

6. 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.

ACELERAÇÃO DE GPU

7. Acelerômetro

8. IMPLEMENTAÇÃO INCONSISTENTE DE APIs

9. 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.
9. CÂMERA

10 . JavaScript Ciclo de vida demorado pois necessita que todos os consumidores da especificação entrem em consenso e implementem a.

11 Detecção de recursos Muitas das detecções de recursos é feita via JavaScript, isso força os desenvolvedores a fazer ao menos parte da marcação em JavaScript(Pilgrim 2010).

Desktop/Firefox Desktop/Google Chrome Smatphone/Android

8 CONCLUSÕES

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

REFERÊNCIAS BIBLIOGRÁFICAS

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

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

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

[4] *ECMAScript 2015 Language Specification*.

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

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

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

[8] *HTML HyperText Markup Language*.

[9] *HTML5 Test, how well does your browser support HTML5?*

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

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

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

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

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

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

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

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

[18] *Using CSS transitions*.

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

[20] 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.

B 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;

C 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.

C.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

C.2 JAVASCRIPT NÃO OBSTRUTIVO

C.3 NODEJS

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

D ALTERNATIVAS AO JAVASCRIPT

Abaixo seguem algumas tecnologias que servem de alternativa ao JavaScript.

D.1 TYPESCRIPT

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

D.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.

E 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.

E.1 SOURCE MAPS

Para encontrar os arquivos minificados a fim de ajudar o desenvolvedor a debugar a aplicação.

E.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.

E.3 PHONEGAP

E.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.

E.4.1 BIBLIOTECAS WEB

O Google Chrome utilizá o Webkit para renderizar seu conteúdo HTML5. O webkit foi criado pela Apple baseando-se no motor de renderização do Konkeror do projeto KDE. Safari e Opera também fazem uso do Webkit. V8 para JavaScript.

O motor de renderização do HTML5 do Firefox é o XXX. O motor de JavaScript é o.