Limitações do HTML5 no desenvolvimento de jogos multiplataforma

Jean Carlo Machado

Bento Gonçalves, Dezembro 2015

Sumário

Lista de Figuras

Lista de Tabelas

1

CONTEXTUALIZAÇÃO

1.1 JOGOS

1.1.1 BENEFÍCIOS

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)

1.1.3 JOGOS E MULTIPLATAFORMA

1.1.4 HTML E MULTIPLATAFORMA

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

> O tempo de desenvolvimento de uma aplicação em HTML5 é 67que 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)

1.1.5 LIMITAÇÕES DE JOGOS MULTIPLATAFORMA COM HTML5

> Funcionalidades foram disponibilizadas de diversas fontes e não foram construídas de forma especialmente 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.

Nem todos os recursos disponíveis através das SDK's nativas estão presentes através do HTML5.

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 multipla-

taforma.

Um tratado completo sobre o assunto requiriria um comparativo entre jogos desenvolvidos nativamente e jogos em HTML5.

1.2.1 O JOGO

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?

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

Abaixo seguem os objetivos deste trabalho.

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 onde o HTML5 se sobressai, 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

Estudar as limitações de desenvolvimento de jogos nas plataformas de dispositivos inteligentes 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.

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

- Performance
- Depuração
- Diferenças em tamanho de tela
- Canvas
- Empacotadores HTML5
- Eventos de entrada
- Vibração

- Acelerômetro
- 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, colateralmente colaborando para a melhoria da própria especificação.

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

Muitos desenvolvedores estão familiarizados com as tecnologias da WEB ou apontam interesse na tecnologia. <!– referenciar –>

Estimular e avançar o estudo da implementação da Open Web;

4

REVISÃO BIBLIOGRÁFICA

4.1 JOGOS

Segundo LEMES (2009, pág. 126) 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.

Essa característica interativa e a dependência comandos sobre uma interface digital, que faz com que o projeto digital desta natureza não seja um filme ou uma animação, e sim um game.

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

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.

Já jogos multiplataforma em HTML5 tem a dificuldade adicional de ter que comportar, na mesma base de código, o feedback adequando para cada plataforma móvel.

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.

Mencionar algum jogo (como WOW) e como ele faz para prender a atenção dos usuários. Candy Crush saga

4.1.1 GÊNEROS

Segundo Isabela Granic e Engels 2014:

> 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).

- Adventure Ação RPG Estratégia Simuladores Esportes Luta Casuais - 'God' Games Educacionais Puzzle Online / Massive Multiplayer

4.1.2 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 ARQUITETURA MULTIPLATAFORMA

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 apenas uma 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; Os pontos negativos dessa abordagem são o principal foco deste trabalho. Mas a um nível macroscópico podemos listar:
- Programas que rodam na web são geralmente mais lentos que os nativos;
- Por falta de especificação ou incompletude de implementação.

4.2.2 JOGOS HÍBRIDOS

Jogos híbridos são jogos geralmente desenvolvidos com tecnologias da web. Mas rodam dentro de um contêiner nativo – possibilitando o acesso à chamadas do sistema, recursos de hardware, eliminando muitias das dificuldades da web.

4.2.3 DESENVOLVIMENTO DE JOGOS NATIVOS

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

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.

Além da replicação dos fontes, esta abordagem 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 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 e oficialmente mantida pela W3C.

O "Hyper Text"refere-se a links que conectam páginas umas as outras, fazendo a Web como conhecemos hoje Mozilla 2015.

HTML foi especificado baseado-se em SGML (Standard Generalized Markup Language), abraçando assim suas premissas:

- Deve ser declarativo, descrevendo estrutura e outros atributos,
- Deve ser ao invés de definir o processamento a ser efetuado no
- Deve ser documento: rigoroso de modo que as mesmas técnicas de
- Deve ser mineração de dados em objetos e bancos de dados possam ser
- Deve ser utilizadas;

Os criadores de navegadores constantemente introduziam elementos de apresentação com o *blink*, *<i>* itálico, *<b>* bold, 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 HTML como uma linguagem de conteúdo semântico, abrindo espaço para a expansão de outras tecnologias como o CSS para responder as demandas de apresentação.

A atual versão do HTML é o HTML5, desenvolvido como um trabalho em conjunto entre a WHATWG e a W3C, seu rascunho foi proposto em 2008 e ratificado em 2014. O HTML5 introduziu elementos interativos que viabilizaram a construção de jogos para a plataforma como: Canvas, áudio, vídeo.

Cada elemento HTML informa algo ao navegador sobre a informação que reside entre o abrir e fechar da tag. <!– ducket pág 20 –>

Um elemento é o abrir fechar de uma tag e todo o conteúdo que dentro dele reside.<!– ducket pág 24 –>

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

4.5 CSS

É uma linguagem de folhas de estilo, criada por Håkon Wium Lie em 1994, com intuito de definir a apresentação de páginas HTML.

> Possibilitam ligação tardia *late biding*. 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. Lie

O CSS é dividido em módulos, contendo aproximadamente 50 deles, cada qual evoluindo separadamente.

Sua última versão, o CSS3, introduziu várias funcionalidades relevantes para jogos, como *media-queries*:possibilitam regras para tamanhos de tela, transformações 3D e animações.

Os navegadores interpretam CSS através da tag "<style>".

<!– Cascading Style Sheets, PhD thesis, by Håkon Wium Lie – provides an authoritative historical reference of CSS pág. 23 –>

<!– Muitos navegadores também suportam aceleração de GPU (Unidade de processamento gráfico) para elementos que tenham transformações 3D. –>

Flow de documento, ordem e posição em que os elementos tem que aparecer na página. Modelo de caixa o que encapsula o conteúdo em um elementos. –>

<!– falar do suporte a variáveis do CSS –>

4.6 JAVASCRIPT

EMACScript, melhor conhecido como JavaScript, criada por Brendan Eich em 1992, é a linguagem 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 Platform*.

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.

JavaScript é uma linguagem de script. Segundo a Ecma Internacional 2012:

> 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.

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.

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. Não obstante, 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, 2013). A nova versão 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.

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. Outrossim, uma vez definidas as especificações, é necessário que os distribuidores do JavaScript implementem o especificado.

Alternativamente, existe uma vasta gama de conversores de código - *transpilers* - para JavaScript; possibilitando programar em linguagens formais e posteriormente gerar código JavaScript. Não obstante, 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 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 de terminal, acesso a arquivos e dispositivos, etc. No contexto de cliente, são definidos objetos como janelas, frames, 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.

4.7 DOCUMENT OBJECT MODEL (DOM)

É uma plataforma e interface agnóstica a linguagem que permite os programas e scripts dinamicamente acessar e atualizar o conteúdo, estrutura e estilo de documentos. Pode ser novamente processado e o resultado aparecer na tela. O navegador cria um DOM quando ele processa os elementos e tags encontrados em um documento HTML. Gmail é uma aplicação de única página (single-page) que se baseia fortemente no DOM para gerar conteúdo dinâmico e interativo oferecido pelo DOM.

4.7.1 CANVAS

1. Troca de tamanhos via Canvas vs DOM 2. Aceleração de GPU 3. API de Áudio (referência 2)

A nova tag <canvas> define um layer gráfico em documentos HTML que pode ser desenhado através de JavaScript.

Permite desenhar diagramas, gráficos e animações [7]. É baseado em bitmap.

Muitas vezes lento. Algumas soluções tentam arrumar isso através da utilização de GPU Apache Cordova utiliza o FastCanvas.

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

4.8 WEBGL

Baseado no OpenGL.

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?

4.9 VIDEO

4.10 AUDIO

Áudio é um componente vital para oferecer grande satisfação aos usuários de jogos. Provê feedback e imerge o usuário. Efeitos de som e música podem servir como mecanismo. Jogadores tem baixa tolerância a volume, deve ser utilizado com cautela.

4.10.1 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.

4.10.2 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.

FORMATOS DE ÁUDIO

4.11 CÂMERA

4.12 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).

Para detectar suporte aos mais variados recursos do HTML5 no navegador do cliente existem duas possibilidades. Pode-se 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 local storage.

4.13 CACHE

Aplicações offline.

Algumas tecnologias desta classe são:

4.14 OFFLINE E ARMAZENAMENTO

> 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. (HASAN et all, 2012)

4.14.1 LOCAL STORAGE

Também conhecido como WebStorage 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.

4.14.2 WEB SQL

Simplesmente 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.

4.14.3 Recursos nativos atualmente indisponíveis

- Suporte à câmera; - Suporte à calendário;

4.14.4 DEBUG

4.14.5 WEINRE

4.14.6 DISPONIBILIZAÇÃO DA APLICAÇÃO

Links com manifestos

4.15 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".

4.16 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.

4.17 PHONEGAP

4.18 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.

4.18.1 O 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.

4.19 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;

4.20 IMPLEMENTAÇÃO

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.

4.21 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.

4.22 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.

4.22.1 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.

4.23 TRABALHOS SIMILARES

(Referência 2) Faz 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; devem ser plataformas mercadologicamente relevantes ao desenvolvimento de jogos, que possibilitem a criação de aplicativos em HTML e que acentuem o antagonismo de características.

Segue-se com a construção de uma lista com os recursos relevantes aos jogos que, empiricamente, 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 potencialmente limitados. Para ser considerado pronto, o protótipo deve ser testado, e estar funcional, com adaptações ou não, em cada uma das plataformas alvo definidas.

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

6

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)

6.1 LIMITAÇÕES

6.1.1

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)

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.

2. OFFLINE

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

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)

3. 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.

Desktop/Firefox Desktop/Google Chrome Smatphone/Android

7

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 apenas 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.

> Conforme JavaScript vai ganhando importância rápido progresso é feito por diferentes empresas a fim de prover boas ferramentas de debug e inspecionamento para JavaScript.

Baixa fricção quer dizer que você pode ir de um site para outro sem ter que instalar, o serviço está em demanda, você não é obrigado a tê-lo em sua tela inicial.

7.0.1 TRABALHOS FUTUROS

EMACSCRIPT 7

Bibliografia

Bruins, Stefan (2014). “The current state of cross-platform game development for different device types”. Em:

Isabela Granic, Adam Lobel e Rutger C. M. E. Engels (2014). “The Benefits of Playing Video Games”. Em: *Radbound University Nijmegen*.

Mozilla (2015). *HTML*. URL: `https://developer.mozilla.org/en-US/docs/Web/HTML`.

Apêndice A

ANEXOS

A.0.1 CONVERSORES PARA HTML5

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

A.0.2 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;

A.0.3 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.

A.0.4 HTTP

A.0.5 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. Não obstante, o intuito deste trabalho é desenvolver um jogo sem auxilio de frameworks pois estes muitas vezes escondem possíveis limitações, desenvolvendo soluções próprios.

- enchant.js: dentre suas funcionalidades constam: orientação à ; - objetos, orientado à eventos, contém um motor de animação, ; - suporta WebGL e Canvas, etc three.js: considerada leve, renderiza ; - WebGL e Canvas, arquitetura procedural ; - quintus: bom para plataformas 2D - limeJs: bom para 2d

A.0.6 INTERFACE E ESCOLHAS DE DESIGN

A.0.7 PROGRESSÃO CONTÍNUA

A.0.8 JAVASCRIPT NÃO OBSTRUTIVO

A.0.9 Arquitetura Cliente Servidor

A.0.10 NODEJS

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

A.0.11 ALTERNATIVAS AO JAVASCRIPT

Abaixo seguem algumas tecnologias que servem de alternativa ao JavaScript.

A.0.12 TYPESCRIPT

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

A.0.13 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.

A.1 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.

A.1.1 SOURCE MAPS

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

A.1.2 ASM.JS

Asm.js é um subconjunto da sintaxe do JavaScript a qual permite grandes aumentos de performance quando em comparação com JavaScript normal. No contexto dos jogos performance é usualmente um recurso estimável, asm.js consegue-o supra utilizando 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 transpilação de outras linhagens como C.

> Muita da performance adicional em relação ao JavaScript é devido a consistência de tipo e a não existência de um coletor de lixo (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.