Unioeste - Universidade Estadual do Oeste do Paraná
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
Colegiado de Ciência da Computação
Curso de Bacharelado em Ciência da Computação

Desenvolvimento de uma metodologia para avaliação de desempenho gráfico 3D de plataformas com suporte ao WebGL

Anderson Roberto Slivinski

CASCAVEL
2011

ANDERSON ROBERTO SLIVINSKI

DESENVOLVIMENTO DE UMA METODOLOGIA PARA AVALIAÇÃO DE DESEMPENHO GRÁFICO 3D DE PLATAFORMAS COM SUPORTE AO WEBGL

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação, do Centro de Ciências Exatas e Tecnológicas da Universidade Estadual do Oeste do Paraná - Campus de Cascavel

Orientador: Prof. Dr. Adair Santa Catarina

CASCAVEL
2011

ANDERSON ROBERTO SLIVINSKI

DESENVOLVIMENTO DE UMA METODOLOGIA PARA AVALIAÇÃO DE DESEMPENHO GRÁFICO 3D DE PLATAFORMAS COM SUPORTE AO WEBGL

Monografia apresentada como requisito parcial para obtenção do Título de Bacharel em Ciência da Computação, pela Universidade Estadual do Oeste do Paraná, Campus de Cascavel, aprovada pela Comissão formada pelos professores:

Prof. Dr. Adair Santa Catarina (Orientador)
Colegiado de Ciência da Computação,
UNIOESTE

Prof. Dr. Victor Francisco Araya Santander
Colegiado de Ciência da Computação,
UNIOESTE

Prof. Edmar André Bellorini
Colegiado de Ciência da Computação,
UNIOESTE

Cascavel, 22 de novembro de 2011

DEDICATÓRIA

Dedico este trabalho aos meus pais, Eloi e Luiza, pelo amor, apoio e suporte que sempre me ofereceram e à Aline, por me incentivar a não desistir nos momentos mais difíceis da graduação.

If you have an apple and I have an apple and we exchange these apples, then you and I still each have one apple, but if you have an idea and I have an idea and we exchange these ideas, then each of us will have two ideas. (George Bernard Shaw)

AGRADECIMENTOS

Agradeço à meus pais, Eloi e Luiza, pelo apoio, incentivo e a paciência que eles tiveram comigo.

Agradeço também à minha irmã, Angela, que apesar de nossos eventuais desentendimentos, sempre foi uma grande amiga, oferecendo ajuda quando eu mais precisei.

À Aline Vaplak Faria, minha namorada e melhor amiga, que sempre me acompanhou durante a graduação, me ajudando nos momentos mais difíceis e compartilhando os momentos mais felizes.

Ao meu professor Adair Santa Catarina, que aceitou minha proposta para este trabalho e me orientou durante o desenvolvimento.

À minha equipe de maratona de programação, Diego Rodrigo Hachmann, Dener Júnior Ribeiro da Cunha e o Tharle Josefi de Camargo, aos quais compartilho muitos momentos de alegria durante os treinos e competições, e ao nosso técnico, Josué Pereira de Castro, que sempre nos apoiou e acreditou em nós.

Aos meus professores Josué Pereira de Castro e Adriana Postal, que durante os primeiros anos da graduação me ajudaram a adquirir uma sólida base sobre Ciência da Computação que eu levarei para o resto da minha vida.

Ao professor Clodis Boscarioli, que por mais que ele queira me bater com um gato morto até ele miar, sempre me me auxiliou e me ajudou a conquistar a maioria das minhas conquistas curriculares.

Aos meus professores, por todo conhecimento compartilhado, pela amizade e dedicação do seu tempo.

Aos meus colegas de classe e curso, pelas risadas e momentos felizes, as noites acordadas, e todo o companheirismo durante o curso.

À todos que, de alguma forma, me auxiliaram durante minha formação. Muito Obrigado!

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

ACM	*Association for Computing Machinery*
API	*Application Programming Interface*
DHTML	*Dynamic HyperText Markup Language*
DOM	*Document Object Model*
FPS	*Frames Per Second*
HTML	*HyperText Markup Language*
IEEE	*Institute of Electrical and Electronics Engineers*
OpenGL	*Open Graphics Library*
OpenGL ES	*Open Graphics Library for Embedded Systems*
RIA	*Rich Internet Applications*
SGML	*Standard Generalized Markup Language*
VRML	*Virtual Reality Modeling Language*
WebGL	*Web Graphics Library*
WWW	*World Wide Web*
XHTML	*eXtensible HyperText Markup Language*
XML	*Extensible Markup Language*

Sumário

Resumo

Com a popularização da internet, os navegadores *web* passaram a ser a aplicação mais utilizadas nos computadores pessoais e nos dispositivos portáteis, como *smartphones* e *tablets*. Isso se deve às características sociais inerentes ao ambiente interativo que a internet disponibiliza. A internet não somente tem se popularizado, como também as pessoas tem a utilizado com mais frequência e por mais tempo. Assim, navegadores fornecem uma maneira eficiente de centralizar as tarefas e conteúdos consumidos e gerados pelos usuários. Dessa maneira, mais desenvolvedores tem buscado criar aplicações utilizando os navegadores *web* como plataforma. Contudo, os navegadores não foram inicialmente planejados para servir como plataforma de aplicações. A fim de contornar este problema, diversas tecnologias foram criadas para fornecer melhores ambientes de desenvolvimentos aos desenvolvedores. Dentre essas tecnologias pode-se citar o HTML5 e o WebGL. Porém, estas tecnologias, por serem recentes, ainda são pouco utilizadas e continuam em desenvolvimento. Assim, o objetivo deste trabalho é realizar uma revisão sistemática para identificar trabalhos que apresentem alguma metodologia ou técnica para a avaliação de performance gráfica 3D. Os resultados dessa revisão serão analisados e utilizados posteriormente para a elaboração de uma metodologia para a avaliação da performance gráfica 3D de navegadores com suporte ao WebGL.

Palavras-chave: Avaliação de Performance, HTML5, WebGL.

Capítulo 1

Introdução

Este capítulo tem como objetivo apresentar as justificativas para a escolha deste tema, a proposta desse trabalho, os objetivos à serem alcançados bem como a estruturação do texto.

1.1 Justificativa

A crescente disponibilidade de acesso à Internet em banda larga [1], [2], tanto em computadores pessoais como em dispositivos móveis, permitiu a popularização de um novo modelo de *software*, os *WebApps* (aplicativos *web*). Esta modalidade permite que os clientes utilizem o *software* como um serviço, simplesmente acessando-os em seus navegadores *web*, sem a necessidade da instalação local do mesmo. Isso traz diversas vantagens sobre o modelo tradicional de *software* como: evitar a instalação local do aplicativo, garantir a utilização de sua versão mais atualizada e a mobilidade entre plataforma, pois o aplicativo depende apenas da existência de um navegador *web*, seja em seu computador pessoal ou em seu *smartphone*. Há também desvantagens como a necessidade do acesso à internet, desempenho e limitações do navegador disponível, pois este nem sempre implementa todos os recursos necessários para a correta execução do aplicativo.

Dentre os diversos tipos de *WebApps* disponíveis no mercado, como clientes de e-mail, reprodutores de vídeo e música, ferramentas para desenho e edição de fotos e vídeos, pacotes de aplicativos de escritório, conversores de arquivos, etc., há um tipo de *WebApp* que se destaca: os jogos. Existem diversos tipos de jogos, em todos os estilos possíveis, desde jogos de cartas à MMORPG's (*Massively Multiplayer Online Role-Playing Game*). Todavia, na sua maioria, são jogos casuais e simples, devido à limitação de se elaborar um jogo para navegadores *web*.

O padrão atual para o desenvolvimento *web*, a HTML 4.0.1 (*HyperText Markup Language*)[3], fornece recursos muito limitados para o desenvolvimento de jogos. Uma destas limitações é a renderização 3D, recurso inexistente na atual especificação. Existem alternativas para se contornar a ausência do 3D, como o Flash e o Silverlight[4], mas estes dependem da instalação de *plugins* que não estão disponíveis para todas as plataformas eliminando algumas das vantagens das *WebApps*, como a portabilidade da aplicação. Estes *plugins* também ocasionam outros problemas como lentidão, travamentos e falhas de segurança.

Entretanto, novas tecnologias se encontram em desenvolvimento, como é o caso da HTML5 [5], [6]. Esta nova versão da HTML traz muitas melhorias para o desenvolvimento de jogos. Uma significante melhoria foi a inclusão do componente *canvas*, que permite de maneira dinâmica o desenho de formas 2D e a criação de animações. Outras grandes melhorias são elementos que permitem executar áudio e vídeo nos navegadores sem a necessidade de *plugins*, a possibilidade de armazenamento local de arquivos e a inclusão dos *WebSockets*, permitindo uma *WebApp* manter uma conexão persistente com outra aplicação.

Após testes de um *Canvas3D* nos laboratórios da Mozilla[7], foi elaborado o WebGL (*Web Graphics Library*) pela Khronos Group[8], responsável pela especificação do OpenGL (*Open Graphics Library*). O WebGL consiste em uma biblioteca baseada no OpenGL ES 2.0 e estende o suporte do JavaScript, tornando possível a geração de gráficos 3D iterativos, acelerados pela GPU (*Graphics Processor Unit*), direto no navegador e sem a necessidade de *plugins*.

Estas novas funcionalidades fornecidas pela HTML5 e WebGL permitem a elaboração de jogos mais complexos, alcançando o nível de jogos desenvolvidos para as plataformas disponíveis hoje no mercado. Um exemplo disto é a realização de um *port* do jogo Quake2 para *web* realizado pela Google, utilizando-se somente HTML5 e WebGL [9].

Porém, mesmo com a especificação do WebGL finalizada e da HTML5 em pleno desenvolvimento, atualmente a maioria dos navegadores não suportam com plenitude todas as suas funcionalidades ou com bons níveis de desempenho. Nem mesmo a recente guerra entre os navegadores, que os força e implementarem os mais novos recursos e otimizações na briga por uma fatia maior de usuários os levou à plena implementação do WebGL e de todos os recursos da HTML5 que poderiam ser utilizados na elaboração de um complexo jogo.

Então, como garantir que o navegador e o dispositivo em que ele se encontra sejam capazes

de executar um jogo em específico?

1.2 Proposta

Possuir a capacidade de mensurar o desempenho de um dispositivo para uma determinada tarefa é crucial, pois assim pode-se avaliar se o dispositivo em foco tem a capacidade de executar um aplicação antes de adquiri-la. Da mesma maneira, com base em testes, é possível avaliar, antes de comprar um certo dispositivo, se este tem a capacidade de executar as tarefas que se pretende realizar nele.

Isso posto, a proposta deste trabalho é a elaboração de uma metodologia para avaliação de desempenho da plataforma formada pelo *hardware* do dispositivo, dos *drivers* deste *hardware* e do navegador com suporte ao WebGL. Esta avaliação nos permite determinar se nossa plataforma tem a capacidade mínima para a execução de um aplicativo em específico. Da mesma maneira, um desenvolvedor poderia se valer desta metodologia de avaliação de desempenho evitando o uso de determinado recurso gráfico, caso a plataforma não consiga índices mínimos nos testes. Outra possibilidade também é a avaliação da implementação do WebGL de diferentes navegadores, aplicando-se o teste nos navegadores em estudo em um mesmo dispositivo.

1.3 Objetivos

Ao final deste trabalho, espera-se alcançar os seguintes objetivos:

- Realizar uma revisão sistemática que sumarize os principais trabalhos, em bases selecionadas, que apresentem as mais recentes técnicas e metodologias de avaliação de desempenho gráfico.
- Analisar as metodologias e ferramentas de avaliação de desempenho para aplicações 3D e definir a mais adequada ao projeto.
- Desenvolver uma nova metodologia, baseada nos resultados obtidos na revisão sistemática de maneira que seja possível avaliar o desempenho gráfico do WebGL.
- Propor melhorias ao processo e trabalhos futuros.

1.4 Estrutura

Este trabalho está dividido em capítulos, incluindo a introdução, da seguinte maneira

- Capítulo 2 - Apresenta a realização de uma revisão sistemática sobre as principais metodologias e técnicas para a avaliação de desempenho gráfico.
- Capítulo 3 - Descreve a fundamentação teórica, base para o desenvolvimento da ferramenta proposta.
- Capítulo 4 - Define a metodologia da aplicação para avaliação de desempenho gráfico para plataformas com suporte ao WebGL.
- Capítulo 5 - Apresenta as conclusões obtidas do trabalho e possíveis trabalhos futuros

Capítulo 2

Metodologias e Técnicas para a Avaliação de Desempenho Gráfico: Revisão Sistemática

2.1 Definição

Segundo Kitchenham[10], uma revisão sistemática é um meio de identificar, avaliar e interpretar toda a pesquisa disponível de um tema em particular. A revisão sistemática é dita como um estudo secundário; os estudos que contribuem para a mesma são ditos estudos primários.

As revisões sistemáticas são pesquisas rigorosas baseadas em estratégias bem definidas que visam detectar a maior quantidade de literatura relevante sobre o tema. Um dos primeiros passos de uma revisão sistemática é a elaboração de um protocolo de pesquisa. Este protocolo guia o pesquisador e evita pesquisas tendenciosas. Por ser baseada em um protocolo de busca criterioso e bem documentado, uma revisão sistemática permite também que leitores e outros pesquisadores possam repetir suas buscas e confirmar a veracidade dos resultados[11].

A revisão sistemática apresentada neste capítulo tem como objetivo investigar trabalhos que contenham metodologias e técnicas para a avaliação de desempenho gráfico 3D.

2.2 Protocolo

O protocolo de revisão sistemática especifica as questões a serem respondidas da pesquisa, bem como os métodos utilizados para se realizar esta pesquisa. Ele também deve definir o escopo da busca, critérios para inclusão/exclusão dos resultados.

As próximas subseções descrevem cada um destes passos do protocolo utilizado nesta revi-

são [11], [10].

2.2.1 Questão da Pesquisa

A definição das questões de pesquisa é um passo muito importante da revisão sistemática, pois elas irão definir toda a pesquisa e a partir delas serão extraídas as palavras-chave utilizadas nas buscas. Desta maneira, as questões foram definidas tendo em mente as metodologias utilizadas para a avaliação de desempenho gráfico 3D. São elas:

- Quais tipos de abordagens estão sendo utilizadas para a avaliação de desempenho gráfico 3D?
- Quais tipos de abordagens estão sendo utilizadas para a avaliação de desempenho gráfico 3D utilizando o WebGL?

A questão primária visa selecionar todos os trabalhos que contenham algum tipo de metodologia ou técnica que descreva como realizar uma avaliação do desempenho gráfico 3D, em qualquer tipo de dispositivo. Já a questão secundária limita este escopo à metodologias e técnicas para avaliação do desempenho gráfico 3D com a utilização do WebGL. Após definidas as questões da pesquisa, passamos para a próxima etapa da pesquisa que é criação das estratégias de busca.

2.2.2 Estratégias de Busca

Nesta etapa define-se o escopo de busca, bases de pesquisa e palavras-chave utilizadas.

Escopo da pesquisa

O escopo da pesquisa será a pesquisa em bases de dados eletrônicas, anais de conferências e *journals*.

Bases de pesquisa

Como bases de busca, foram utilizados periódicos disponíveis através do site Periódicos do Portal da CAPES. Estas bases foram escolhidas devido a sua relevância internacional na área de Ciência da Computação. Seguem elas:

- ACM Digital Library

- IEEE Xplore
- SciELO.ORG
- SCOPUS (Elsevier)
- ScienceDirect (Elsevier)
- ScienceDirect - E-Books (Elsevier)
- SpringerLink (MetaPress)

Palavras-chave utilizadas

Estas palavras chaves foram extraídas das questões da pesquisa e em sua maioria são derivações e sinônimos de termos das perguntas traduzidas para o inglês. Isso se deve ao fato de serem bases internacionais, o que tornaria pouco relevante utilizar palavras-chaves em português.

As palavras-chave utilizadas são: *Methodology, benchmark, performance, evaluation, 3D graphics,* HTML5, *WebGL, video card*.

***Strings* de busca**

Com base nestas palavras-chave foram elaboradas as strings de busca, que foram utilizadas nas pesquisas. Seguem elas:

1. Título=(3D graphics benchmark OR 3D graphics benchmark methodology) OU
   Assunto=(3D graphics benchmark OR 3D graphics benchmark methodology)
2. Título=(3D graphics performance OR Evaluation 3D graphics performance) OU
   Assunto=(3D graphics performance OR Evaluation 3D graphics performance)
3. Título=(WebGL benchmark OR WebGL benchmark methodology) OU
   Assunto=(WebGL benchmark OR WebGL benchmark methodology)
4. Título=(WebGL performance OR Evaluation WebGL performance) OU
   Assunto=(WebGL performance OR Evaluation WebGL performance)

5. Título=(HTML5 3D graphics benchmark OR HTML5 3D graphics benchmark methodology) OU Assunto=(HTML5 3D graphics benchmark OR HTML5 3D graphics benchmark methodology)

6. Título=(HTML5 3D graphics performance OR Evaluation HTML5 3D graphics performance) OU Assunto=(HTML5 3D graphics performance OR Evaluation HTML5 3D graphics performance)

7. Título=(Video card performance OR Evaluation Video card performance) OU Assunto=(Video card performance OR Evaluation Video card performance)

8. Título=(Video card benchmark OR Video card benchmark methodology) OU Assunto=(Video card benchmark OR Video card benchmark methodology)

9. Título=(Video card 3D graphics performance OR Evaluation Video card 3D graphics performance) OU Assunto=(Video card 3D graphics performance OR Evaluation Video card 3D graphics performance)

10. Título=(Video card 3D graphics benchmark OR Video card 3D graphics benchmark methodology) OU Assunto=(Video card 3D graphics benchmark OR Video card 3D graphics benchmark methodology)

2.2.3 Critérios para a seleção de estudos e procedimentos

Os critérios de inclusão/exclusão auxiliam na definição de quais resultados são relevantes para a pesquisa, de acordo com as questões propostas. Esses critérios também evitam que o pesquisador realize um estudo tendencioso influenciado por suas próprias opiniões.

Critérios de Inclusão/Exclusão

- Artigos que apresentem alguma técnica ou metodologia sobre a avaliação de desempenho gráfico 3D.

- Artigos publicados em congressos ou periódicos das bases acessíveis na íntegra pela busca de periódicos da Capes.

- Artigos disponíveis gratuitamente.

- Artigos que possuam nível de relevância igual ou superior a três, na busca do portal da CAPES.

- Artigos publicados nos últimos 5 anos.

Estes critérios não foram escolhidos ao acaso. O primeiro critério define que o artigo primário em potencial deve possuir alguma técnica ou metodologia sobre como efetuar uma avaliação de desempenho gráfico 3D. O segundo e terceiro critérios definem artigos que não possuam limitações quanto a sua leitura para o autor deste trabalho. Definir um nível mínimo de relevância tem como propósito remover estudos primários que não possuam nenhuma afinidade com as questões de pesquisa. Por último, pelo fato do WebGL ter sua especificação finalizada somente no ano de 2011 e novas gerações de placas gráficas serem lançadas com média a cada 18 meses, a utilização de uma metodologia muito antiga pouco viria a contribuir para a análise de desempenho gráfico 3D das placas de vídeos mais recentes, as quais suportam esta nova API.

Processo Eliminatório

Após definidos os critérios para seleção dos resultados primários em potencial, é efetuada uma filtragem em duas etapas. Primeiramente é realizada uma pré-filtragem baseada nos critérios de inclusão já definidos:

- Primeiramente é aplicada uma estratégia de busca para identificar estudos primários em potencial.

- Em seguida é verificado se os estudos foram publicados nos últimos 5 anos e possuem nível de relevância igual ou superior a 3.

- Verifica-se os mesmos estão disponíveis gratuitamente.

- Finalmente o resumo do artigo é analisado para verificar se o mesmo se enquadra no objetivo da pesquisa em questão.

Após esta etapa de pré-filtragem realizada, os artigos selecionados são lidos na íntegra e avaliados com a relação às questões de pesquisas definidas no início do protocolo. Os artigos que passarem nesta filtragem final seguem para a fase de extração de dados.

2.2.4 Extração de dados

Dos artigos aprovados pelo processo de seleção serão extraídos os seguintes dados:

- Tipo do artigo: teórico, experimental ou ambos;
- Descrição da técnica, metodologia, processo ou ferramenta;

2.3 Pesquisa

Com as *strings* de busca e os critérios para a seleção dos estudos primários em potencial definidos, realizou-se a busca nas bases estabelecidas, através do portal da CAPES.

2.3.1 Resultados

Primeira pesquisa

Utilizou-se a primeira *string*, com um total de 60 resultados, sendo 4 selecionados após a etapa de pré-filtragem.

Tabela 2.1: Artigos selecionados na pré-filtragem da primeira pesquisa

Nº	**Nome do artigo**	**Ano**	**Relevância**	**Incluso**
01	*3D Graphics Performance Scaling and Workload Decomposition and Analysis*	2007	4	Não
O trabalho apresenta uma análise da ferramenta de *benchmarking* 3DMark 2005 e verifica se a mesma é capaz de escalar com o acréscimo de múltiplos processadores, mas não apresenta nenhuma metodologia ou técnica para a análise do desempenho.				
02	*Strategies of Enhancing the Performance of Embedded 3D Graphics Applications*	2007	3	Não
O trabalho apresenta estratégias de como otimizar a performance de geração de gráficos 3D em dispositivos embarcados, mas não demonstra nenhuma técnica ou metodologia de como avaliar esta performance				
03	*Power Analysis of Mobile 3D Graphics*	2006	3	Não
Este artigo descreve como os gráficos 3D em dispositivos móveis influenciam no consumo de energia dos mesmos.				
04	*GRAAL: A Framework for Low-Power 3D Graphics Accelerators*	2008	3	Não
Apresenta um *framework* que auxilia no desenvolvimento de uma metodologia para o desenvolvimento e simulação de aceleradores gráficos 3D embarcados.				

Segunda pesquisa

Utilizou-se a segunda *string*, com um total de 60 resultados, sendo 2 selecionados após a etapa de pré-filtragem.

Tabela 2.2: Artigos selecionados na pré-filtragem da segunda pesquisa

Nº	**Nome do artigo**	**Ano**	**Relevância**	**Incluso**
05	Signature-based workload estimation for mobile 3D graphics	2006	4	Não
Apresenta um estimador de desempenho baseado na assinatura de execução de gráficos 3D em dispositivos móveis				
06	A Ubiquitous 3D Graphics Modeler for Mobile Devices	2008	3	Não
Este trabalho apresenta um modelador na plataforma cliente-servidor que possibilita dispositivos móveis modelar gráficos 3D.				

Terceira pesquisa

Utilizou-se a terceira *string*, com um total de 59 resultados, com nenhum resultado selecionado após a etapa de pré-filtragem.

Quarta pesquisa

Utilizou-se a quarta *string*, com um total de 30 resultados, com nenhum resultado selecionado após a etapa de pré-filtragem.

Quinta pesquisa

Utilizou-se a quinta *string*, com um total de 60 resultados, sendo 3 selecionados após a etapa de pré-filtragem, mas com 2 resultados repetidos da primeira pesquisa.

Tabela 2.3: Artigos selecionados na pré-filtragem da quinta pesquisa

Nº	**Nome do artigo**	**Ano**	**Relevância**	**Incluso**
07	Towards HTML 5 and interactive 3D graphics	2010	3	Não
Apresenta as novas tecnologias para renderização de gráficos 3D nos navegadores.				

Sexta pesquisa

Utilizou-se a sexta *string*, com um total de 60 resultados, sendo 4 selecionados após a etapa de pré-filtragem, mas com 1 resultado repetido da segunda pesquisa e 1 resultado repetido da

quinta pesquisa.

Tabela 2.4: Artigos selecionados na pré-filtragem da sexta pesquisa

Nº	**Nome do artigo**	**Ano**	**Relevância**	**Incluso**
08	A 195mW, 9.1MVertices/s fully programmable 3D graphics processor for low power mobile devices	2007	3	Não
Demonstra um processador gráfico 3D totalmente programável de baixo consumo para dispositivos móveis de baixo consumo				
09	A Proposed Digital Rights Management System for 3D Graphics Using Biometric Watermarks	2010	3	Não
Artigo que demonstra o gerenciamento de direitos autorais para grafícos 3D utilizando marcas d'água biometricas				

Sétima pesquisa

Utilizou-se a sétima *string*, com um total de 60 resultados, com nenhum resultado selecionado após a etapa de pré-filtragem.

Oitava pesquisa

Utilizou-se a oitava *string*, com um total de 60 resultados, com nenhum resultado selecionado após a etapa de pré-filtragem.

Nona pesquisa

Utilizou-se a nona *string*, com um total de 60 resultados, com nenhum resultado selecionado após a etapa de pré-filtragem.

Décima pesquisa

Utilizou-se a décima *string*, com um total de 30 resultados, com nenhum resultado selecionado após a etapa de pré-filtragem.

2.3.2 Reavaliação da Revisão Sistemática

Após a coleta dos resultados da revisão sistemática, a mesma foi realizada novamente removendo o critério de eliminação por data, a fim de selecionar resultados que, mesmo não apresentando metologias capazes de avaliar o desempenho das arquiteturas de geração gráfica

3D atuais, auxiliassem na elaboração de uma nova metodologia para a avaliação do desempenho gráfico 3D das arquiteturas atuais.

Primeira pesquisa

Utilizou-se a primeira *string*, com um total de 60 resultados, sendo 1 adicionado após a etapa de pré-filtragem.

Tabela 2.5: Reavaliação: Artigos selecionados na pré-filtragem da primeira pesquisa

Nº	**Nome do artigo**	**Ano**	**Relevância**	**Incluso**
01	*Dynamic 3D graphics workload characterization and the architectural implications*	1999	3	Não
Faz uma análise da carga de trabalho de aplicações 3D e avalia as implicações que elas trazem sobre a arquitetura.				

Segunda pesquisa

Utilizou-se a segunda *string*, com um total de 60 resultados, sendo 1 adicionado após a etapa de pré-filtragem.

Tabela 2.6: Reavaliação: Artigos selecionados na pré-filtragem da segunda pesquisa

Nº	**Nome do artigo**	**Ano**	**Relevância**	**Incluso**
02	Low-power 3D graphics processors for mobile terminals	2005	4	Não
Realiza uma investigação sobre o *pipeline* de execução gráfica 3D e avalia as otimizações da arquitetura gráfica para satisfazer os requisitos de desempenho.				

Terceira à Décima pesquisa

Foram realizadas novamente as pesquisas utilizando-se as *strings* de número 2 à 10 sem nenhuma nova adição aos resultados.

2.4 Conclusão

O fato de nenhum trabalho primário em potencial ter passado pelos processos de filtragem indicam que, mesmo sendo esta uma área com vários produtos no mercado para a análise de desempenho gráfico 3D, não existe nenhuma metodologia ou técnica documentada ou publicada sobre o assunto no âmbito da pesquisa realizada.

2.5 Dificuldades Encontradas

Diversos problemas ocorreram durante as buscas. O Portal de Periódicos da CAPES por diversas vezes se mostrava indisponível para acesso. Outro problema foram as falhas durante as buscas. Constantemente pesquisas em bases de grande importância como ACM e IEEE eram canceladas, necessitando assim realizar novamente a busca.

Como as pesquisas são realizadas em diversas bases e algumas delas retornam grandes quantidades de resultados mesmo irrelevantes, o tempo de busca era de muitos minutos.

Além disso, o site do portal também apresentou, por vezes, falhas de renderização, impedido a escolha de uma busca avançada e a seleção das bases, fato que impossibilitava a realização das pesquisas.

Estes três fatores, combinados, fizeram com que o tempo consumido na Revisão Sistemática fosse superior ao estimado inicialmente.

Capítulo 3

Fundamentação Teórica

Este capítulo apresenta a fundamentação teórica necessária para o entendimento dos aspectos da metodologia que será desenvolvida posteriormente. A Seção 3.1 discute sobre tecnologias para utilização de gráficos na *web*. A Seção 3.2 apresenta a HTML5. A Seção 3.3 traz uma descrição do *OpenGL*.

3.1 Gráficos na *Web*

3.1.1 HTML

A HTML (HyperText Markup Language) uma linguagem de marcação baseada no SGML (*Standard Generalized Markup Language*) desenvolvida para elaborar páginas *web*, as quais são interpretadas pelos navegadores *web*. É atualmente a linguagem predominante neste tipo de desenvolvimento.

Seu desenvolvimento iniciou-se em 1989 como parte da uma pesquisa realizada por Tim Berners-Lee no CERN (Organização Europeia para a Pesquisa Nuclear) para a elaboração de um sistema de armazenamento e a vinculação de documentos estáticos em um sistema baseado em rede. Foi padronizado em 1993 e se tornou o principal responsável pela popularização da WWW (*World Wide Web*). Desde 1994, a especificação é mantida pelo W3C (*World Wide Web Consortium*), um consórcio internacional fundado pelo Tim Berners-Lee com o objetivo de maximizar o potencial da HTML, criando protocolos comuns que promovam sua evolução e garantam a interoperabilidade.[12]

Os elementos da HTML são as *tags*, blocos base para a construção das paginas *web*. Basicamente os documentos são formados pelo texto, pelas *tags* de formatação, responsáveis por

informar ao navegador como apresentar aquele texto e pelos *hyperlinks*, que conectam os documentos entre si. Dentre todas as *tags*, encontram-se elementos responsáveis por inserir conteúdo multimídia, como imagens, vídeos, áudio e conteúdos provenientes de *plugins*, como conteúdo desenvolvido em Adobe Flash.[3]

Desde o final de 1999 se encontra na versão 4.0.1, com uma errata publicada em 2001. Sua paralisação se deve ao fato de o W3C ter focado desde a versão 3.5 da HTML, e de 2002 à 2006 de forma exclusiva, na elaboração do XHTML, uma especificação da HTML baseada no XML (Extensible Markup Language), onde faz uso de uma sintaxe mais rigorosa e menos ambígua facilitando o processamento e extensão.[12]

Embora não tenha sido desenvolvida para a criação de paginas dinâmicas, com a introdução do DOM (*Document Object Model*, Modelo de Objeto de Documentos)[13] na HTML, um certo grau de dinamismo foi possível nas paginas HTML estáticas através de sua manipulação dinâmica com JavaScript, denominando esta técnica de DHTML (*Dynamic HyperText Markup Language*)[14]. O DOM é uma especificação da W3C para a criação de uma API que permite com que documentos eletrônicos, como HTML, XHTML e XML, tenham seu conteúdo, estrutura e estilo alterados de maneira dinâmica.

Com o advento do DHTML foi possível desenvolver sistemas mais complexos para navegadores *web*. Veja na Figura 3.1 um exemplo de uma réplica do classico jogo *Pong* desenvolvido em DHTML.

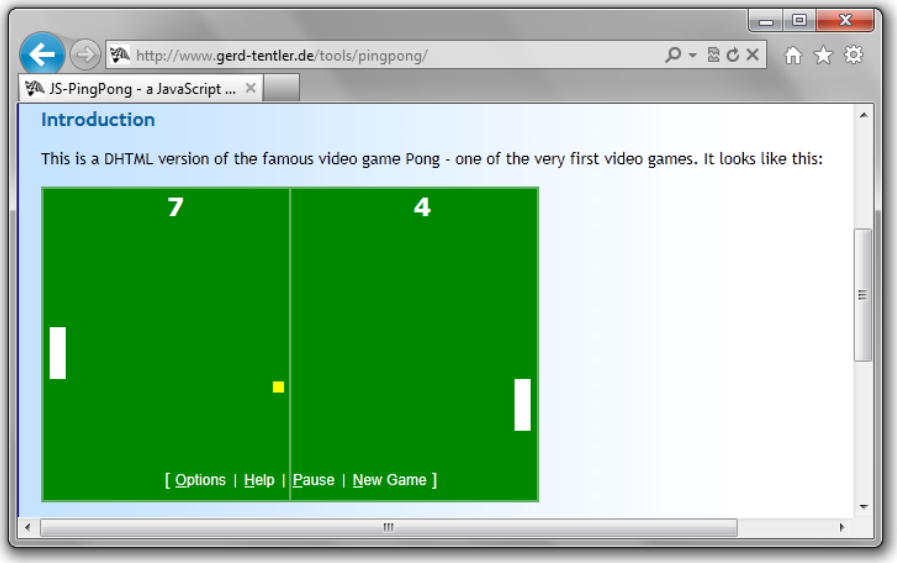

Figura 3.1: Exemplo de jogo desenvolvido em HTML

3.1.2 VRML

A VRML (*Virtual Reality Modeling Language*) é uma linguagem desenvolvida para descrever mundos virtuais conectados através da internet conectados por meio de *hyperlinks*. Esta linguagem executa sobre um visualizador VRML, que na maioria dos casos é um *plugin* para um navegador web.

Esta linguagem foi concebida em 1994 durante a Primeira Conferência Anual da *World Wide Web*. Os participantes da sessão sobre Realidade Virtual concordaram que era necessário haver uma linguagem comum para especificar a descrição de mundos 3D e os WWW *Hyperlinks*, um análogo da HTML para realidade virtual. Foi padronizada pela ISO (Organização Internacional para Padronização) em 1997.

A VRML permite, por meio de um arquivo de texto, que pode ser escrito utilizando um software específico ou diretamente em modo texto, modelar mundos virtuais utilizando objetos 3D através de formas básicas como esferas, cubos, cones, cilindros, etc., ou através de formas criadas pelo programador, como as extrusões. Nestes objetos, é possível definir cor, transparência, brilho e texturas, associando-os à um *bitmap*.[15]

No Código 3.1 temos um exemplo de uma cena composta com a VRML97, e na Figura 3.2 temos o resultado deste exemplo.

Listing 3.1: Exemplo de uma cena em VRML97

```
1  #VRML V2.0 utf8
2  NavigationInfo { type [ "EXAMINE" "ANY" ] }
3  Group {
4    children [
5        Viewpoint {
6          centerOfRotation 0 -1 0
7          description "Hello world!"
8          position 0 -1 7
9        }
10       Transform {
11         rotation 0 1 0 3
12         children [
13             Shape {
14               geometry Sphere {
15               }
16               appearance Appearance {
17                 material DEF LightBlue Material {
18                   diffuseColor 0.1 0.5 1
19                 }
20                 texture ImageTexture {
21                   url [ "earth-topo.png"
22                         "earth-topo.jpg"
```

```
23                          "earth-topo-small.gif"
24                          "http://www.web3d.org/x3d/content/examples/
                               Basic/earth-topo.png"
25                          "http://www.web3d.org/x3d/content/examples/
                               Basic/earth-topo.jpg"
26                          "http://www.web3d.org/x3d/content/examples/
                               Basic/earth-topo-small.gif"
27                      ]
28                    }
29                  }
30                }
31            ]
32          }
33          Transform {
34            translation 0 -2 0
35            children [
36                Shape {
37                  geometry Text {
38                    string [ "Hello" "world!" ]
39                    fontStyle FontStyle {
40                      justify [ "MIDDLE" "MIDDLE"  ]
41                    }
42                  }
43                  appearance Appearance {
44                    material USE LightBlue
45                  }
46                }
47            ]
48          }
49      ]
50  }
```

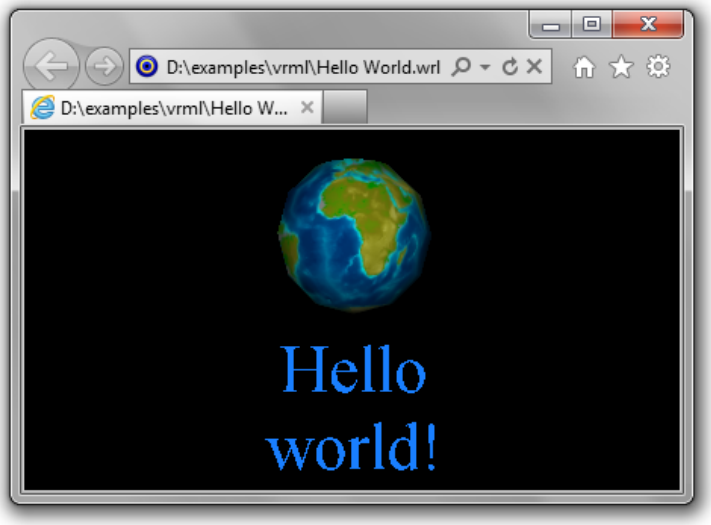

Figura 3.2: Resultado do exemplo em VRML

Apesar de ter sido desenvolvido para a elaboração de mundos virtuais, também foi utilizado

para o desenvolvimento de jogos, como no caso do jogo *The Castle*[16] que foi desenvolvido utilizando um motor de jogo baseado em VRML.

3.1.3 X3D

A partir de uma revisão da especificação da VRML, em 2005 foi criado o X3D, um novo padrão para substitui-lo. O X3D adiciona novas funcionalidades ao VRML, como formatos de codificação de dados adicionais, conformidade mais rigorosa e uma arquitetura modular que permite uma abordagem modular para apoiar o padrão.

Dentre as melhorias, podemos destacar a separação do *runtime* de arquitetura do *runtime* de codificação de dados, a adição de uma variedade de formatos de codificação, com destaque para o XML, adição de novos objetos gráficos, de comportamento e iterativos e a definição de perfis para diferentes necessidades de mercado.[17]

Apesar de ser uma nova especificação da VRML, o X3D, devido ao fato deste suportar múltiplos formatos de arquivos e linguagem de programação, foi dividido em 3 especificações ISO separadas[18], onde:

- **ISO 19775:200x** - Descreve, de maneira abstrata, todas as funcionalidades do sistema, isto é, os modelos estruturais e de execução. Funcionalidades de linguagens de programação externas também são expressadas em termos abstratos como grupos de serviços que podem ser requisitados e providos em um mundo ideal.

- **ISO 19776:200x** - Contém um conjunto de descrição de formatos de arquivos. A estrutura abstrata de três tipos de formatos de arquivos são descritos, sendo dois do tipo texto e um binário. Dentre os formatos de arquivo do tipo texto temos o clássico VRML e o XML. O formato de arquivo binário ainda se encontra em desenvolvimento.

- **ISO 19777:200x** - Descreve um conjunto de mapeamento para as linguagens de programação externa. Atualmente, as linguagens suportadas são o Java e o ECMAScript, também conhecido por JavaScript.

Contudo, o X3D não deixa de ser compatível com a VRML, sendo possível converter mundos descritos em VRML para X3D utilizando ferramentas adequadas, ou então, no caso de um

arquivo sem *scripts*, é possível converter o arquivo manualmente, bastando alterar o cabeçalho do arquivo e especificando o *profile*.

O Código de exemplo 3.2 é uma conversão para X3D do Código 3.1 descrito em VRML, gerando o mesmo resultado final observado na Figura 3.2.

Listing 3.2: Exemplo de uma cena em X3D

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.2//EN" "http://www.web3d.org/
    specifications/x3d-3.2.dtd">
3 <X3D profile='Immersive' version='3.2' xmlns:xsd='http://www.w3.org/2001/
    XMLSchema-instance' xsd:noNamespaceSchemaLocation='http://www.web3d.
    org/specifications/x3d-3.2.xsd'>
4   <Scene>
5     <Group>
6       <Viewpoint centerOfRotation='0 -1 0' description='Hello world!'
           position='0 -1 7'/>
7       <Transform rotation='0 1 0 3'>
8         <Shape>
9           <Sphere/>
10          <Appearance>
11            <Material DEF='LightBlue' diffuseColor='0.1 0.5 1'/>
12            <ImageTexture url='"earth-topo.png" "earth-topo.jpg" "earth-
                topo-small.gif" "http://www.web3d.org/x3d/content/
                examples/Basic/earth-topo.png" "http://www.web3d.org/x3d/
                content/examples/Basic/earth-topo.jpg" "http://www.web3d.
                org/x3d/content/examples/Basic/earth-topo-small.gif"'/>
13          </Appearance>
14        </Shape>
15      </Transform>
16      <Transform translation='0 -2 0'>
17        <Shape>
18          <Text string='"Hello" "world!"'>
19            <FontStyle justify='"MIDDLE" "MIDDLE"'/>
20          </Text>
21          <Appearance>
22            <Material USE='LightBlue'/>
23          </Appearance>
24        </Shape>
25      </Transform>
26    </Group>
27  </Scene>
28 </X3D>
```

X3D e a HTML5

O X3D Working Group, grupo responsável por manter a especificação do X3D, tem trabalhado para estabelecer uma sólida fundação para o suporte do X3D na HTML5. Para isso, três abordagens foram desenvolvidas para a exibição das cenas X3D dentro de uma página

HTML[19]:

- **Objeto X3D em uma página HTML como um objeto embutido**. Para isso, a pagina HTML inclui um elemento "object"que referencia para uma cena X3D, implementado via um *plugin* X3D. Dados podem ser trocados entre a página e a cena X3D utilizando eventos DOM.

- **Cena X3D inserida diretamente na HTML como XML**. O Código X3D é inserido diretamente na pagina HTML, renderizado por um *plugin* ou diretamente pelo navegador. Recurso em fase experimental, demonstrado pelo X3DOM.

- **Criação de um Canvas3D ou uma API de acesso especial**. A pagina HTML iria incluir algum tipo de elemento *canvas* que permita acesso programático à pagina, assim, o X3D poderia acessar este elemento astraves da API de acesso e desenhar o *bitmap*. Recurso deferido pela W3C pois ela ainda estuda a inclusão de novos recursos ao HTML5.

X3DOM

O X3DOM é um *framework* experimental que tem como objetivo a inserção do código X3D diretamente na pagina HTML sobre a arvore DOM utilizando um espaço de nomes XML específico. Com ele é possível com que o conteúdo 3D seja manipulado inserindo, editando ou removendo objetos da arvores DOM. Nenhum *plugin* específico é necessário para sua execução[20].

Uma cena X3DOM pode ser acelerada via hardware através do WebGL. No Código 3.3 temos o Código da cena X3D 3.2 inserido diretamente em uma página HTML utilizando um espaço de nomes XML com o X3DOM.

Listing 3.3: Exemplo de uma cena em X3DOM

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.
    org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml">
4   <head>
5     <link rel="stylesheet" type="text/css" href="http://www.x3dom.org/
        x3dom/src/x3dom.css" />
6     <script type="text/javascript" src="http://www.x3dom.org/x3dom/src/
        x3dom.js"></script>
7   </head>
8   <body>
9     <x3d xmlns="http://www.x3dom.org/x3dom" width="500" height="500">
10      <Scene>
```

```
11          <Group>
12            <Viewpoint centerOfRotation='0 -1 0' description='Hello world!'
                  position='0 -1 7'/>
13            <Transform rotation='0 1 0 3'>
14              <Shape>
15                <Sphere/>
16                <Appearance>
17                  <Material DEF='LightBlue' diffuseColor='0.1 0.5 1'/>
18                  <ImageTexture url='"earth-topo.png" "earth-topo.jpg" "
                        earth-topo-small.gif" "http://www.web3d.org/x3d/
                        content/examples/Basic/earth-topo.png" "http://www.
                        web3d.org/x3d/content/examples/Basic/earth-topo.jpg"
                        "http://www.web3d.org/x3d/content/examples/Basic/
                        earth-topo-small.gif"'/>
19                </Appearance>
20              </Shape>
21            </Transform>
22            <Transform translation='0 -2 0'>
23            <Shape>
24              <Text string='"Hello" "world!"'>
25                <FontStyle justify='"MIDDLE" "MIDDLE"'/>
26              </Text>
27              <Appearance>
28                <Material USE='LightBlue'/>
29              </Appearance>
30            </Shape>
31            </Transform>
32          </Group>
33        </Scene>
34      </x3d>
35    </body>
36 </html>
```

3.1.4 Flash

O Flash é um software de animação vetorial, desenvolvido pra criar animações para serem exibidas nas páginas *web*. Geralmente é utilizado para criar propagandas, jogos, animações e reproduzir conteúdo multimídia, como áudio e vídeo. Contudo, seu uso não fica restrito à essas aplicações, em alguns casos ele é usado para o desenvolvimento completo de páginas *web* em substituição da HTML[21]. Recentemente foi reposicionado como uma ferramenta para RIA (*Rich Internet Applications*).

Dentre suas funcionalidades, o Flash tem como destaque a manipulação de gráficos vetoriais. Contudo, ele pode prover muito mais funcionalidades, como a manipulação de gráficos rasterizados, *streaming* bidirecional de áudio e vídeo e a captura e manipulação de informações provindas do usuário por meio do *mouse*, teclado, microfone e câmera. Para sua manipulação,

o Flash contém o ActionScript[22], uma linguagem orientada a objetos derivada do ECMAScript, que foi inicialmente desenvolvida para a criação de sites com a utilização da plataforma Flash. Primeiramente desenhada para o controle de animações vetoriais 2D simples, em suas versões mais recentes oferece recursos de interatividade, permitindo a criação de Jogos e aplicações ricas para a internet (RIA's), como *streaming* de áudio e vídeo. A partir de sua versão 10, o Flash Player ganhou suporte ao desenvolvimento de aplicações 3D[4][23], inclusive com aceleração via *hardware* por parte a API Stage3D desenvolvida pela Adobe[24]. Dentre as vantagens do Flash perante ao HTML5 com WebGL, podemos destacar o suporte para aplicações em tela cheia e a possibilidade de captura do ponteiro do *mouse*, recursos fundamentais no desenvolvimento de jogos modernos como jogos de tiro em primeira pessoa. Contudo, apresenta algumas desvantagens, como o fato de ser necessária a instalação do *plugin* e os problemas que isto acarreta, como travamentos e lentidão.

O conteúdo Flash pode ser exibido em diversos sistemas computacionais através do Flash Player, que é disponibilizado de maneira gratuita como um *plugin* para navegadores *web* comuns e para outros dispositivos móveis em embarcados, como *smartphones* através do Flash Lite.

A plataforma Flash entrou em processo de declínio com a criação da HTML5, onde é possível recriar a maioria dos conteúdos hoje criados em Flash, sem a utilização de *plugins* de terceiros por parte do usuário. O declínio do Flash foi comemorado por alguns usuário, devido aos constantes problemas por parte do Flash Player, como travamentos e falhas de segurança. Em Novembro de 2011, foi anunciado que a Adobe, atual proprietaria desta tecnologia, não tinha mais planos para o desenvolvimento de sua versão Lite para plataformas móveis e, com a constante evolução e expansão da HTML5, tem reposicionado o Flash como uma plataforma para aplicações RIA. A partir da versão CS5 do ambiente de desenvolvimento Flash, é possivel exportar o conteúdo gerado para sua versão equivalente da HTML5 *Canvas*. Outras ferramentas tem surgido com o propósito de converter conteúdo Flash para HTML5, como no caso do Google Swiffy[25].

Atualmente é a plataforma mais utilizada para a criação de jogos para *web*, com dezenas de sites exclusivos de jogos em Flash. Um exemplo de um jogo desenvolvido para esta plataforma é o Tanki Online, desenvolvido em Flash 3D, que pode ser visto na Figura 3.3.

Figura 3.3: Exemplo de jogo desenvolvido em Flash 3D

3.1.5 Silverlight

O Silverlight é um *framework* criado pela Microsoft para o desenvolvimento e execução de aplicações ricas para a internet, com recursos e propostas similares ao Adobe Flash. Seu ambiente de execução está disponível por meio de um *plugin* para navegadores *web* que executam sob o sistema operacional Windows e Mac OS X[26]. Há também uma implementação livre, denominada Moonlight, desenvolvida pelo Mono Project, financiado pela Novell em cooperação com a Microsoft, que tem como objetivo trazer as funcionalidades do Silverlight para os sistemas operacionais Linux, FreeBSD e outras plataformas livres[27]. Assim como o Flash, permite a criação de jogos 3D acelerados via hardware.

3.2 HTML5

A HTML5 é a mais nova versão da linguagem de estruturação e apresentação de conteúdo para *web*. Sua especificação ainda se encontra em fase de esboço, porém diversos navegadores *web* já implementam diversas de suas funcionalidades[6].

Esta quinta versão da HTML traz consigo diversas funcionalidades novas, como semântica e acessibilidade. O seu maior destaque é a inclusão de novos recursos que antes sé eram possíveis com a utilização de tecnologias de terceiros, por meio de *plugins*, como o suporte a reprodução de áudio e vídeo. Dentre os novos recursos disponibilisados, podemos destacar a inclusão do componente *canvas* para desenhos, componentes para reprodução de àudio e vídeo, uma API para manipulação de aquivos, armazenamento local de arquivos, o *WebSockets* para a realização de conexões persistentes com outras aplicações e o WebGL para gráficos 3D[6]. Este conjunto de recursos possibilita a criação de aplicações ricas para *web*, mais epecificamente a elaboração de jogos complexos diretamente no navegador, alcançando o nível de jogos desenvolvidos para as plataformas disponíveis hoje no mercado.

3.2.1 *Canvas*

O *canvas* é o equivalente em inglês para lona, muito utilizada pelos artistas como superfície de pintura presa à um quadro de madeira [28]. Na HTML5, o *canvas* é uma área *bitmap* de modo imediato que pode ser manipulada pelo JavaScrip. Este modo imediato se refere ao modo

com que o *canvas* renderiza os *pixels* na tela. A HTML5 *Canvas* repinta completamente a área *bitmap* a cada *frame* através das chamadas da API *Canvas* do JavaScript.[29]

Na Código 3.4 temos um pequeno exemplo da HTML5 *Canvas*, onde é desenhado um quadrado vermelho e um texto escrito "Hello World"em azul, como pode ser visto na Figura 3.4.

Listing 3.4: Código de um exemplo da HTML5 Canvas

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>Canvas Test</title>
5      <meta http-equiv="content-type" content="text/html; charset=ISO
           -8859-1">
6    </head>
7    <bod>
8      <canvas id="canvas" width="200" height="200">
9        Se você estiver vendo isso, o seu navegador não suporta WebGL.
10     </canvas>
11
12     <script type="text/javascript">
13       var canvas = document.getElementById('canvas');
14       var context = canvas.getContext('2d');
15       context.fillStyle = "rgb(255, 0, 0)";
16       context.fillRect(30, 30, 50, 50);
17
18       context.font = "20px serif";
19       context.fillStyle = "rgb(0, 0, 255)";
20       context.fillText("Hello World", 100, 100);
21     </script>
22   </body>
23 </html>
```

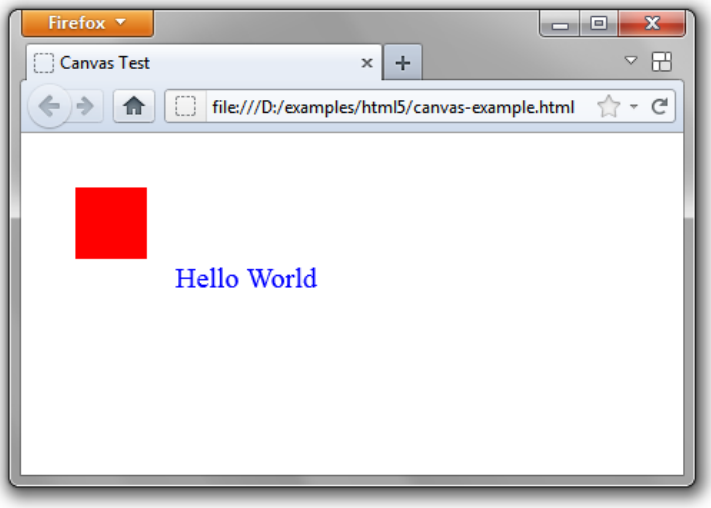

Figura 3.4: Resultado final do exemplo da HTML5 Canvas

3.3 OpenGL

O OpenGL (*Open Graphics Library*) é uma API livre, multiplataforma, multilinguagem utilizada na computação gráfica 2D e 3D para a criação de aplicativos gráficos, jogos, ambientes 3D, visualização científica, simulação de voo, dentre outros.

Desenvolvida inicialmente pela SGI (*Silicon Graphics Inc.*) em 1992, passou a ser gerenciada pelo Khronos Group, um consórcio sem fundos lucrativos formado pelas principais empresas de tecnologias com foco em multimidia, como a AMD, Nvidia, Intel Corporation, Apple Inc., ARM Holdings e a própria SGI. Possui atualmente mais de 100 companhias.

Ele é formado por um conjunto de funções, responsáveis por fornecer acesso a praticamente todos os recursos do *hardware* de vídeo. Inicialmente fornecia um conjunto de funções responsáveis por exibir os objetos da cena, calcular iluminação e sombreamento, realizar operações sobre os vértices, como rotação e translação, dentre outras funções. Porém, após o lançamento da sua versão 2.0, passou a utilizar os *Vertex Programs*, pequenos programas escritos em GLSL (*OpenGL Shading Language*), uma linguagem de *shading* baseada na sintaxe da linguagem C, que são compilados e executados nos processadores gráficos. Estes *Vertex Programs* são formados por dois pequenos *shaders*, o *Vertex Shader* e o *Fragment Shader*. O *Vertex Shader* é responsável por fazer os cálculos baseados nos vértices dos objetos 3D da cena, como o calculo de perspectiva, rotação, translação e iluminação. Em contra partida, os *Frament Shaders*, também conhecidos como *Pixel Shaders*, são responsáveis pela rasterização do objeto e o cálculo individual por *pixel* do preenchimento, texturas, sombreamento e iluminação. Desta forma, o OpenGL passou a fornecer um ambiente para que o programador tenha completo acesso ao *hardware*, porém fica responsável por qualquer tipo de cálculo necessário.

É uma API gráfica ubíqua, presente em alguma de suas variações em praticamente todos os dispositivos eletrônicos, desde computadores pessoais à *smartphones*, *tablets*, televisores e *video games* e dispositivos embarcados.

Muito utilizado no desenvolvimento de jogos eletrônicos, como o recente *Rage* desenvolvido pra *id Software*, que tem como programador John D. Carmak, um dos principais evangelistas da biblioteca.

No Código 3.5 temos um pequeno exemplo de uma aplicação desenvolvida na linguagem C, com a utilização do OpenGL e a *toolkit* GLUT (*OpenGL Utility Toolkit*), um conjunto de

funções que auxiliam no desenvolvimento de aplicações em OpenGL. Neste exemplo é criada apenas uma janela sem nenhum desenho e o título "Hello World".

Listing 3.5: Exemplo básico da utilização do OpenGL

```
#ifndef WIN32 //if using windows then do windows specific stuff.
#define WIN32_LEAN_AND_MEAN //remove MFC overhead from windows.h witch
    can cause slowness
#define WIN32_EXTRA_LEAN

#include <windows.h>
#endif

#include <GL/gl.h>
#include <GL/glut.h>
#include <GL/glu.h>

void setup() { /* empty function  nothing to setup yet */ }
void display() { /* empty function   required as of glut 3.0 */ }

int main(int argc, char *argv[])
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_RGB | GLUT_DEPTH | GLUT_DOUBLE);
        glutInitWindowSize(800,600);
        glutCreateWindow("Hello World");

        setup();
        glutDisplayFunc(display);

        glutMainLoop();

        return 0;
}
```

Na Figura 3.5 temos uma imagem do jogo de tiro em primeira pessoa *Rage*, desenvolvido em OpenGL.

3.3.1 OpenGL ES

O OpenGL ES (*OpenGL for Embedded Systems*), é um subconjunto da API OpenGL, desenvolvida para ser utilizada em sistemas embarcados, como *smartphones*, *tablets* e consoles de *video game*. Foi criado e é mantido pelo Khronos Group.

Atualmente existem três versões do OpenGL ES, a 1.0, 1.1 e 2.0, baseadas nas especificações *desktop* do OpenGL 1.3, 1.5 e 2.0, respectivamente. Por ser um subconjunto da especificação para *desktop*, é possivel reduzir um código em OpenGL 1.5 para OpenGL ES 1.1, mas o contrário não necessariamente é possível, pois depende dos recursos utilizados.

Figura 3.5: *Rage*, exemplo de um jogo desenvolvido em OpenGL

A maioria das funcionalidades introduzidas no OpenGL 2.0 e posteriores derivaram das funcionalidades desenvolvidas no OpenGL ES, como a remoção do "glBegin ... glEnd"para o desenho de primivas, em favorecimento dos *arrays* de vértices e a utilização dos *Vertex* e *Fragment Shaders*.

3.3.2 WebGL

O WebGL é a especificação de uma biblioteca que estende a capacidade do JavaScript, permitindo com que seja possível gerar gráficos 3D interativos em navegadores *web* compatíveis, renderizando imagens *bitmap* através do contexto do componente HTML5 *Canvas*. Por ser apenas uma especificação, cada navegador *web* deve implementar a sua versão da biblioteca, sem a necessidade da utilização de *plugins*[8].

Seu desenvolvimento iniciou com experimentos sobre o *Canvas3D* realizados pelo Vladimir Vukicevic da Mozilla Foundadion[7]. O primeiro protótipo foi disponibilizado em 2006 e ao final do ano de 2007, ambos os navegadores Mozilla Firefox e o Opera fizeram suas próprias implementações individuais. Em 2009, em parceria com a Mozilla, o Khronos Group iniciou o

grupo de trabalho para o desenvolvimento do WebGL, tendo sua especificação final, sob versão 1.0, lançada em Março de 2011[8].

É uma biblioteca baseada no OpenGL ES 2.0, permitindo assim, a sua implementação em navegadores *web* de dispositivos embarcados. Existem diversas implementações, como a do navegador Google Chrome, do WebKit (utilizada pelo Safari), do Opera e do Mozilla Firefox. As implementações do Google Chrome e do Mozilla Firefox permitem total aceleração via *hardware*, porém, somente em computadores com os sistemas operacionais Windows Vista e Windows 7[30]. Além disso, ambas utilizam o ANGLE (*Almost Native Graphics Layer Engine*), uma camada de abstração que, em caso do computador em execução não possuir *drivers* para o OpenGL 2.0 ou superior para que possa ser feita a tradução das chamadas WebGL para OpenGL, realiza uma tradução para as chamadas da API do DirectX 9. Já na implementação do Opera, é realizada uma aceleração via *hardware* em qualquer sistema operacional, desde que o *hardware* possua os requisitos mínimos requiridos, permitindo assim que o WebGL rode com mais eficiência em diversos tipos de dispositivos como televisores e *smartphones*[30].

O WebGL ainda possui muitas restrições, como a falta de possibilidade de captura do ponteiro do mouse e a execução de aplicativos em tela cheia real[8]. Porém, mesmo com estas limitações, vem se expandindo muito, com muitos desenvolvedores criando jogos para a plataforma. Esta popularidade se deve ao fato da crescente expansão de dispositivos portáteis como *smartphones* e *tablets*, que utilizam o OpenGL ES como bilioteca gráfica, permitindo uma migração simples para o WebGL. Outro fato que contribui para esta expansão é a própria popularidade dos navegadores *web* como plataformas computacionais, centralizando muito das atividades realizadas diariamente pelos usuários.

Na Figura 3.6 podemos observar um *port* do jogo Quake II para WebGL utilizando o Google Web Toolkit.

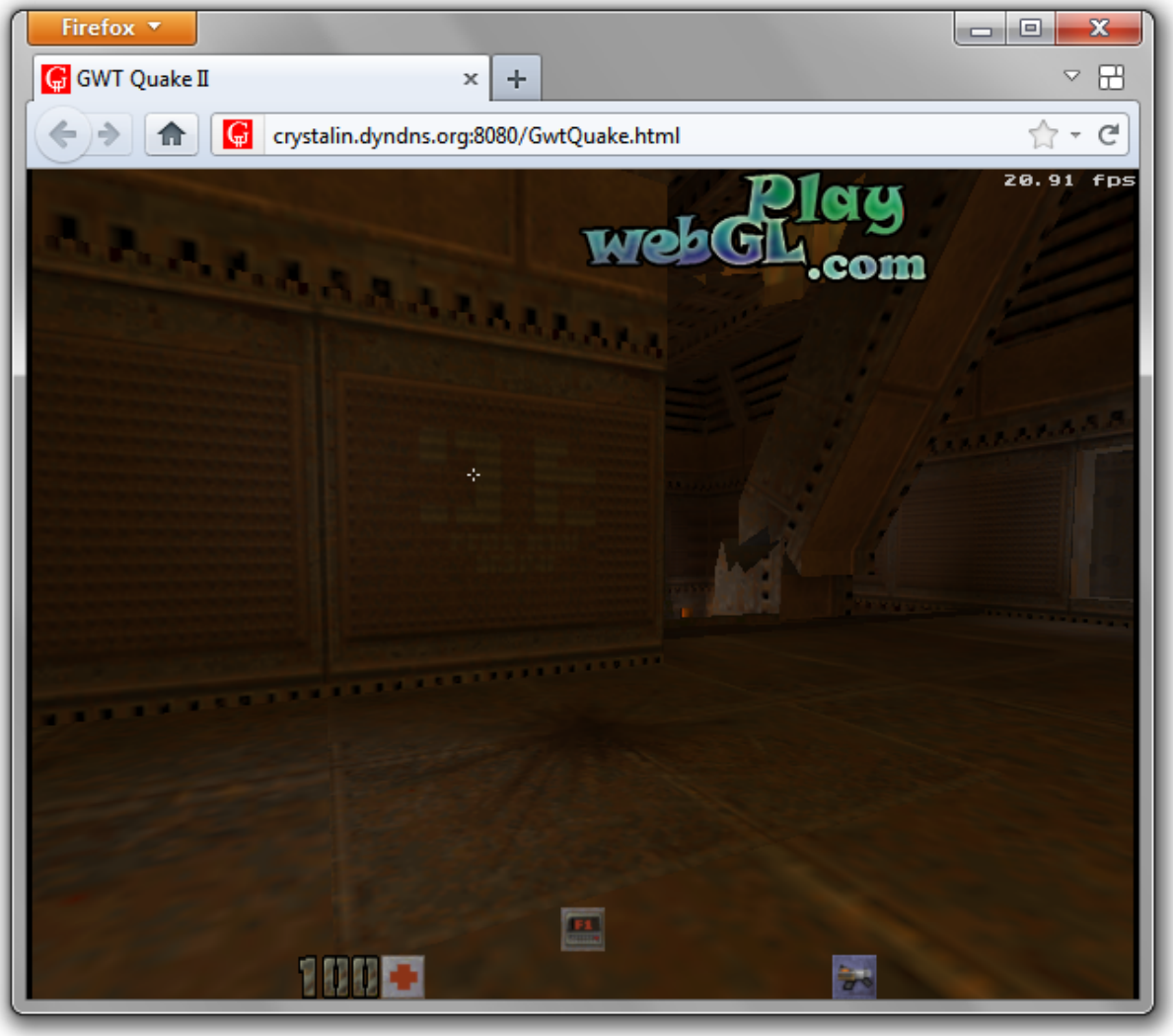

Figura 3.6: *Port* do jogo Quake II para WebGL

Capítulo 4

Metodologia

Este capítulo tem como objetivo apresentar uma metodologia para avaliação da performance gráfica em plataformas com suporte ao WebGL.

4.1 Introdução

No capítulo 2, a revisão sistemática apresentada teve como objetivo encontrar e identificar trabalhos que apresentem técnicas e metodologias que auxiliem no desenvolvimento de uma nova metodologia capaz de analisar a performance gráfica do WebGL em plataformas que o suportem. Porém, não foram encontrados resultados relevantes neste contexto. Isso demonstra que a abordagem aqui proposta é inovadora, além de tratar de tecnologias recentes e pouco documentadas. Dessa maneira, a abordagem foi definida considerando-se os conhecimentos obtidos com a leitura da documentação oficial das tecnologias utilizadas e com a utilização de ferramentas de avaliação de desempenho já existentes, como o 3DMark[31] e o Unigine Heaven Benchmark[32].

Nas seções abaixo será apresentada uma metodologia para ser utilizada na implementação de uma ferramenta para avaliação da performance gráfica de plataformas com suporte ao WebGL.

4.2 Sobre a metodologia

Esta metodologia foi desenvolvida de maneira que diferentes plataformas sejam avaliadas e comparadas através de um mesmo índice, tornando possível sua comparação quanto ao resultado final. Dentre os fatores de variação na performance gráfica de plataformas WebGL, podemos destacar:

- Arquitetura do *hardware*, pois a maneira com que uma aplicação será executada depende de como a arquitetura foi desenvolvida, sendo assim, uma mesma aplicação pode obter resultados diferentes em arquiteturas diferentes que executem à uma mesma velocidade;

- Implementação dos *drivers*, que variam de acordo com o sistema operacional em execução, devido à arquitetura e políticas sobre *drivers* nele implantados;

- Implementação do suporte ao WebGL no navegador *web*, onde cada um tem a liberdade de realizar sua própria implementação do WebGL, podendo ou não, acelerar sua execução via *hardware*.

Devido ao fato desta metodologia permitir avaliar diferentes plataformas e das variáveis acima descritas, a ideia inicial de se estabelecer uma única pontuação padrão que indicaria a quantidade de recursos que a plataforma conseguiria executar, de maneira satisfatória, se torna inviável, pois como descrito no capítulo 3, o WebGL é o padrão de uma API que por sua vez possui uma implementação própria em cada navegador *web*, podendo ou não ser acelerado via *hardware*. Aceleração que, por sua vez, também varia de navegador para navegador, comprometendo a confiabilidade da métrica para analisar os recursos gráficos possíveis de serem executados em um aplicativo 3D.

Sendo assim, a metodologia proposta retorna não uma, mas várias pontuações ao final dos testes, possibilitando ao desenvolvedor analisar cada uma independentemente e averiguar se dada plataforma tem capacidade de executar sua aplicação 3D sobre WebGL. Assim o programador pode adaptar a configuração gráfica do aplicativo a fim de maximizar a performance sem redução significativa da qualidade gráfica. O cálculo destas pontuações tem como base o FPS (*Frames Per Second*) ou Quadros Por Segundo, que é a unidade de medida da cadência de um dispositivo audiovisual qualquer, como uma câmara de cinema ou vídeo, uma webcam, um projetor cinematográfico ou de vídeo, etc. Significa o número de imagens que tal dispositivo registra, processa ou exibe por unidade de tempo, também denominado Hertz (Hz) pelo Sistema Internacional de medidas, que significa "ciclos por segundo". Estes cálculos são realizados baseando-se no quanto um determinado recurso gráfico incide sobre o desempenho geral da aplicação relacionado ao FPS médio que este recurso consegue alcançar durante os testes.

Os passos para a implementação e a descrição da metodologia se encontram na sessão seguinte.

4.3 Fases e Etapas

Para a implementação da ferramenta de avaliação gráfica 3D sobre WebGL, esta abordagem foi dividida em fases e etapas, seguindo uma sequência lógica que resulta, ao seu fim, na ferramenta finalizada.

Fase 01: Definir o domínio de teste.

Isso consiste em definir quais tipos de aplicações se pretende mensurar o desempenho, como por exemplo, especificar o domínio de testes para testar o dispositivo quanto do seu desempenho na execução de jogos 3D. Outros tipos de domínio de testes possíveis seriam aplicativos CAD (*Computer-Aided Design*, ou Desenho Assistido por Computador) ou aplicativos para modelagem e animação 3D.

Fase 02: Definir o conjunto de recursos gráficos à serem utilizados.

Nesta fase é necessário definir o conjunto de recursos gráficos que se queira avaliar, como texturas e os filtros que se possam aplicar à elas, os métodos de iluminação a serem testados, *Depth of Field* (profundidade de campo) e *Anti-Aliasing* (Remoção de Serrilhado), de acordo com o domínio de testes especificado.

Fase 03: Compor uma cena que utilize todos os recursos selecionados, de modo que represente o domínio definido.

Nesta fase, se deve elaborar uma cena, ou seja, uma composição gráfica que contenham objetos que representem uma estrutura comum de uma aplicação referente ao domínio em estudo. Por exemplo, se o domínio for definido como jogos 3D com a temática de tiro em primeira pessoa, se deve elaborar uma cena com elementos típicos destes jogos, como um ambiente de guerra em meio à uma cidade ou ao meio de uma mata fechada.

Fase 04: Implementar a ferramenta de avaliação, utilizando a cena composta, seguindo as seguintes etapas:

- **Etapa 01:** Para cada recurso definido, executar o teste utilizando a cena composta aplicando-se o recurso à todos os elementos da cena. Ao final de cada teste, calcular o FPS médio.

- **Etapa 02:** Para cada recurso definido, executar o teste utilizando a cena composta, porém desabilitando apenas o recurso selecionado. Este teste permitirá calcular o impacto deste recurso na execução do teste. Ao final de cada teste, calcular o FPS médio.

- **Etapa 03:** Por fim, executar um teste com todos os recursos habilitados, à fim de mensurar a capacidade real de execução da plataforma.

Na Figura 4.1, é apresentado um fluxograma das etapas para a execução do aplicativo que implementa a metodologia aqui descrita para a avaliação gráfica de plataformas com suporte ao WebGL.

Fase 05: Calcular as pontuações com base nos conjuntos de resultados.

Para isso, deve se aplicar as métricas descritas na próxima Seção.

4.4 Métricas da Metodologia

Para o cálculo das pontuações, é utilizado os resultados dos testes que foram obtidos na Fase 4. A partir destes resultados é possível realizar o cálculo das pontuações da seguinte maneira:

- Primeiramente se calcula a o quanto cada recurso gráfico incide sobre a redução da performance quando utilizado, conforme a formula da Equação 4.1, onde temos o FPS médio de cada recurso calculado durante a Etapa 02 divido pela diferença da soma de todos os FPS médio pelo FPS total da aplicação calculado na Etapa 03. Isso retorna, para cada recurso gráfico, a porcentagem de quanto ele influência na queda de desempenho geral.

$$PERC_i = \frac{AVG\_FPS\_ETAPA\_02_i}{\left(\sum_{i=1}^{n} AVG\_FPS\_ETAPA\_02_i\right) - AVG\_FPS\_ETAPA\_03} \quad (4.1)$$

- Calcula-se então a pontuação parcial para cada um dos recursos gráficos testados, tanto para a pontuação efetiva quanto para a pontuação bruta, conforme as formulas das Equações 4.2 e 4.4 respectivamente. Para o cálculo da pontuação efetiva, leva-se apenas em consideração o FPS médio que o recurso gráfico alcançou individualmente, ponderado pela porcentagem que o mesmo incide sobre a queda de desempenho geral, calculada na

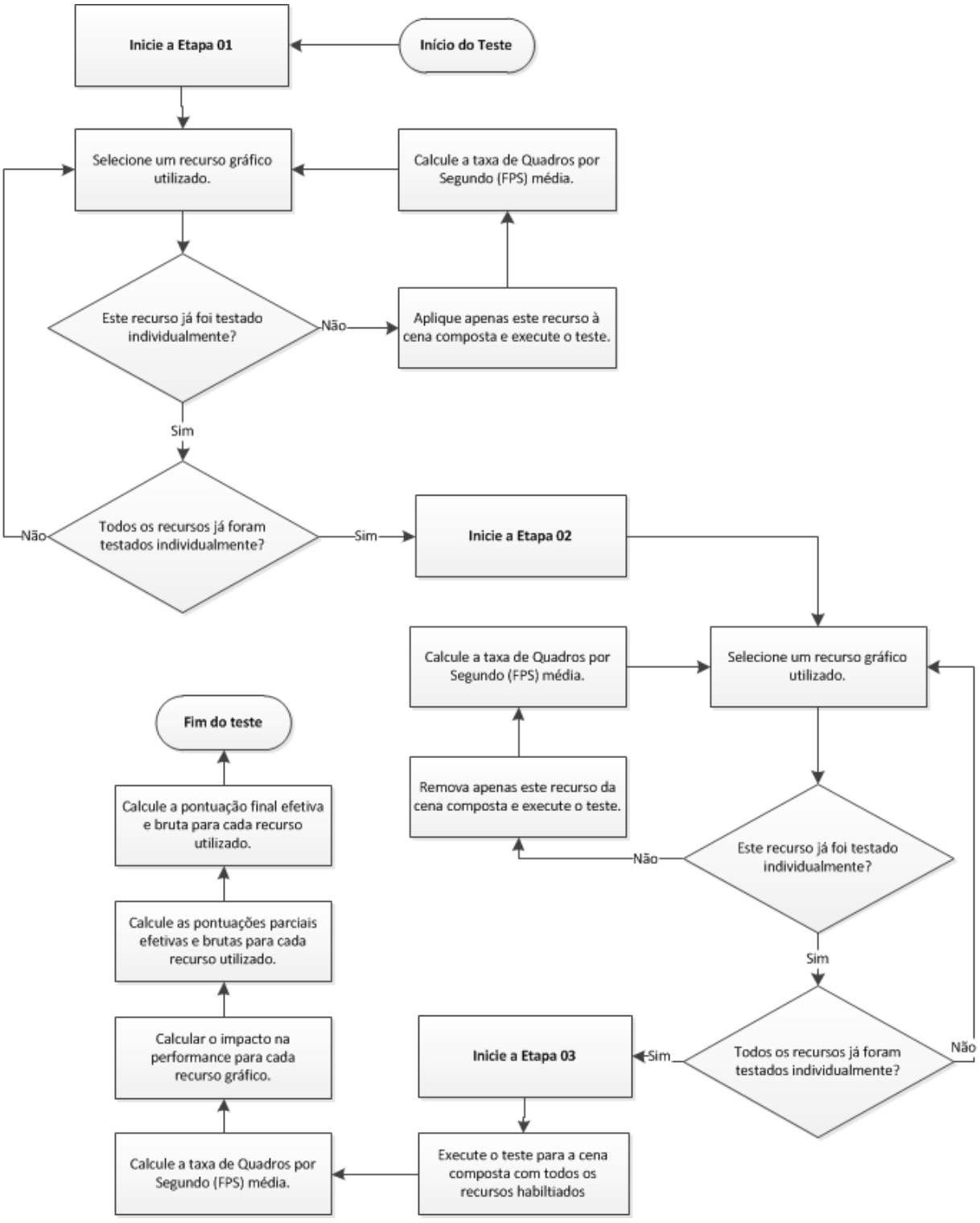

Figura 4.1: Fluxograma que representa as etapas da execução do conjunto de testes.

Equação 4.1. Contudo, para o calculo da pontuação efetiva do dispositivo, leva-se em consideração também o tamanho da *viewport*, que é a dimensão da porção da tela utilizada para a exibição dos testes, denominados por VP_WIDTH para a largura e VP_HEIGHT para a altura. Nota-se também, em ambas as equações, que o FPS médio calculado na Etapa 01 é dividido pelo valor 30 acrescido de uma ponderação do valor 30 pela diferença de 1 e o percentual que o recurso gráfico incide sobre a queda de desempenho geral. Isso se deve ao fato de que 30FPS se aproximar do valor mínimo que a visão humana consegue visualizar uma animação constante em uma troca sequencial de imagens [33], assim, espera-se que o recurso gráfico atinja no mínimo este valor específico.

$$POINTS_i = \frac{AVG\_FPS\_ETAPA01_i}{30 + (30 \times (1 - PERC_i))} \times (1000 \times PERC_i) \tag{4.2}$$

$$RP\_TEMP_i = \frac{AVG\_FPS\_ETAPA01_i}{30 + (30 \times (1 - PERC_i))} \times (1000 \times PERC_i) \tag{4.3}$$

$$RAW\_POINTS_i = RP\_TEMP_i \times \frac{VP\_WIDTH \times VP\_HEIGHT}{1280 \times 720} \tag{4.4}$$

- É calculada a pontuação bruta geral e a pontuação efetiva geral somando-se os resultados parciais respectivos de cada um dos conjuntos, seguindo as Equações 4.5 e 4.6, que consistem no somatório das pontuações intermediarias calculadas para cada recurso gráfico.

$$\sum_{i=1}^{n} POINTS_i \tag{4.5}$$

$$\sum_{i=1}^{n} RAW_POINTS_i \tag{4.6}$$

A escolha de uma pontuação satisfatória de 1000 pontos se deve apenas pelo apelo visual do número, que poderia ser 1, 10, 100, ou qualquer valor que o desenvolvedor que implementar a metodologia queria, atentando apenas pela modificação do mesmo valor nas equações. Já a escolha do tamanho de referência para a *viewport* de 1280 *pixels* de largura por 720 *pixels* de altura se deve ao fato de ser uma resolução de tela que se aproxima da maioria dos *notebooks*

hoje vendidos no mercado e dos monitores de entrada de linha, bem como da resolução atual dos *tablets* mais populares e dos *smartphones* de alto desempenho previstos para chegarem ao final do ano de 2011 e início de 2012, tornando-se assim uma resolução satisfatória que engloba diversos dispositivos que podem ser avaliados com esta metodologia.

Capítulo 5

Conclusão e Trabalhos Futuros

Neste trabalho foram apresentados estudos e o desenvolvimento de uma metodologia para avaliação de desempenho gráfico 3D em uma plataforma com suporte ao WebGL.

5.1 Dificuldades Encontradas

Durante o desenvolvimento deste trabalho, diversas dificuldades foram encontradas. A falta de experiência ao se realizar uma revisão sistemática, aliada aos problemas encontrados durante esta etapa contribuíram para o aumento no tempo de sua realização, acarretando num atraso geral do cronograma do trabalho. O conhecimento apenas superficial das tecnologias e métodos aqui aplicados foram responsáveis por uma extensa revisão bibliográfica, necessária, devido ao enorme grau técnico envolvido. A ausência de resultados por parte da revisão sistemática também foi uma grande dificuldade, pois, pretendia-se ao início deste trabalho, adaptar de metodologias e técnicas para a análise de performance gráfica 3D para o contexto do WebGL. Sendo assim, de maneira inovadora foi necessária a elaboração de uma metodologia para tais análises.

5.2 Revisão Sistemática

A revisão sistemática apresentada neste trabalho teve como objetivo pesquisar estudos que apresentassem algum tipo de metodologia para avaliação gráfica 3D. Porém, ao seu final constatou-se que atualmente não existe nenhum tipo de metodologia desenvolvida para esse fim, ou não existe nenhuma metodologia documentada sobre o assunto. Sendo assim, o foco deste trabalho que era de desenvolvimento de um software para avaliação de desempenho de

dispositivos com suporte ao WebGL mudou para o desenvolvimento de uma metodologia para avaliação de plataformas com suporte ao WebGL.

5.3 Metodologia Desenvolvida

A metodologia desenvolvida teve por objetivo servir de auxílio para programadores que desejam implementar uma ferramenta para análise da performance gráfica 3D em plataformas que suportem o WebGL. Para tanto foi criado um processo que consiste em criar métricas para avaliar se a plataforma em estudo possui os requisitos mínimos para a execução de um dado aplicativo.

A metodologia proposta neste trabalho se trata de uma abordagem inovadora, visto que não foi encontrada documentação sobre nenhum processo parecido. A falta de documentação teórica sobre as novas tecnologias utilizadas dificultou o processo de desenvolvimento da metodologia. Porém, espera-se, que esta seja útil para programadores que desejem implementar um sistema de avaliação gráfica 3D de plataformas com suporte ao WebGL.

5.4 Trabalhos Futuros

Como trabalhos futuros podemos citar a implementação da metodologia aqui proposta para validar seus resultados e verificar se a mesma atinge o seu objetivo. Também propomos a elaboração de um estudo do domínio dos jogos 3D, a fim de determinar o conjunto de recursos gráficos mais utilizados. Podendo-se assim utilizar esta ferramenta para criar uma base de dados que contenha os requisitos mínimos de diversos jogos, validados de maneira com que o usuário possa executar o teste de avaliação aqui proposto e prontamente tenha acesso à um acervo de informações. Este acervo informa de maneira mais realista, se comparadas as informações de requisitos mínimos e recomendados de *hardware* oferecidos pelas desenvolvedoras de jogos, para determinar se o testado dispositivo tem capacidade suficiente para executar um jogo em específico. Outro trabalho futuro é a utilização dos resultados da métrica aqui desenvolvida para auxilar o desenvolvedor à analisar o desempenho da plataforma em estudo para reconfigurar os parâmetros gráficos do jogo ou aplicativo de forma com que se obtenha uma melhora na performance sem comprometer a qualidade gráfica.

Referências Bibliográficas

[1] GROUP, M. M. *Internet World Stats*. 2010. Disponível em: <http://www.internetworldstats.com/stats.htm>.

[2] BROADBANDINFO. *International Broadband Usage Statistics*. Disponível em: <http://www.broadbandinfo.com/high-speed-internet/travel/international-broadband-usage-statistics.html>.

[3] W3C. *HTML 4.01 Specification*. dec. 1999. Disponível em: <http://www.w3.org/TR/html401/>.

[4] SHANKLAND, S. *Adobe: Flash to take 3D graphics plunge*. 2010. Disponível em: <http://news.cnet.com/8301-306853-20009940-264.html>.

[5] JIANPING, Y.; JIE, Z. Towards html 5 and interactive 3d graphics. In: *Educational and Information Technology (ICEIT), 2010 International Conference on*. [S.l.: s.n.], 2010. v. 1, p. V1–522 –V1–527.

[6] W3C. *HTML5 Working Draft*. may. 2011. Disponível em: <http://www.w3.org/TR/html5/>.

[7] VUKIC'EVIC', V. *Canvas 3D: GL power, web-style*. Disponível em: <http://blog.vlad1.com/2007/11/26/canvas-3d-gl-power-web-style/>.

[8] GROUP, K. *WebGL Specification*. feb. 2011. Disponível em: <https://www.khronos.org/registry/webgl/specs/1.0/>.

[9] GOOGLE. *Quake II GWT Port*. Disponível em: <http://code.google.com/p/quake2-gwt-port/>.

[10] KITCHENHAM, B. et al. Systematic literature reviews in software engineering - a systematic literature review. *Information and Software Technology*, Butterworth-Heinemann Newton, MA, USA, v. 51, n. 1, p. 7–15, January 2009.

[11] CONTE, T. U.; MENDES, E.; TRAVASSOS, G. H. *Revisão Sistemática sobre Processos de Desenvolvimento para Aplicações Web*. [S.l.], 2004.

[12] RAGGETT, D. et al. *Raggett on HTML 4*. [S.l.]: Addison-Wesley Professional, 1998.

[13] W3C. *Document Object Model (DOM)*. jan. 2009. Disponível em: <http://www.w3.org/DOM/>.

[14] W3C. *Web Style Sheets - Dynamic HTML*. fev. 2011. Disponível em: <http://www.w3.org/Style/dynamic>.

[15] CONSORTIUM, W. *The Virtual Reality Modeling Language - Version 1.0 Specification*. may 1995. Disponível em: <http://www.web3d.org/x3d/specifications/vrml/VRML1.0/index.html>.

[16] KAMBURELIS, M. *The Castle*. Disponível em: <http://castle-engine.sourceforge.net/castle.php>.

[17] CONSORTIUM, W. *X3D Architecture and base components Edition 2*. jul. 2008. Disponível em: <http://web3d.org/x3d/specifications/ISO-IEC-19775-1.2-X3D-AbstractSpecification/index.html>.

[18] CONSORTIUM, W. *X3D and Related Specifications*. Disponível em: <http://www.web3d.org/x3d/specifications/>.

[19] CONSORTIUM, W. *X3D and HTML5*. Disponível em: <http://www.web3d.org/x3d/wiki/index.php/X3D_and_HTML5>.

[20] IGD, F. *X3DOM*. Disponível em: <http://www.x3dom.org/?page_id=2>.

[21] INCORPORATED, A. S. *Adobe Flash Player 11 / Features*. Disponível em: <http://www.adobe.com/br/products/flashplayer/features.html>.

[22] INCORPORATED, A. S. *ActionScript reference and documentation*. Disponível em: <http://www.adobe.com/devnet/actionscript/documentation.html>.

[23] MAGAZINE, F. *Flash Player 10 feature: 3D support*. may. 2008. Disponível em: <http://www.flashmagazine.com/news/detail/flash_player_10_feature_3d_support/>.

[24] INCORPORATED, A. S. *Stage 3D*. Disponível em: <http://www.adobe.com/devnet/flashplayer/stage3d.html>.

[25] GOOGLE. *Google Swiffy*. Disponível em: <http://www.google.com/doubleclick/studio/swiffy/>.

[26] MICROSOFT. *Silverlight Features*. Disponível em: <http://www.microsoft.com/silverlight/features/>.

[27] PROJECT, M. *Moonlight*. Disponível em: <http://www.mono-project.com/Moonlight>.

[28] ANTONOV, A. L. *Classical Oil Painting Technique*. Disponível em: <http://www.cartage.org.lb/en/themes/arts/painting/principl-tech/paint-instruct/class-oil-paint/classoilpaint.htm>.

[29] FULTON, S.; FULTON, J. *HTML5 Canvas*. O'Reilly Media, 2011. (O'Reilly Series). ISBN 9781449393908. Disponível em: <http://books.google.com.br/books?id=zFvRqdL_pUAC>.

[30] JOHANSSON, T. *WebGL and Hardware Acceleration*. fev. 2011. Disponível em: <http://my.opera.com/core/blog/2011/02/28/webgl-and-hardware-acceleration-2>.

[31] FUTUREMARK. *3D Mark Video card benchmark*. 2011. Disponível em: <http://www.3dmark.com>.

[32] UNIGINE. *Unigine Heaven Benchmark*. 2011. Disponível em: <http://unigine.com/products/heaven>.

[33] CORPORATION, M. *Temporal Rate Conversion*. dec. 2001. Disponível em: <http://msdn.microsoft.com/en-us/windows/hardware/gg463407.aspx>.