Faculdade de Tecnologia de Americana
Curso Superior de Tecnologia em Análise de Sistemas e Tecnologia da Informação - Jogos Digitais

Análise do Processamento Gráfico Utilizando a Técnica de *Shaders*

RAFAEL BLUMLEIN CARVALHO

Americana, SP
2014

Faculdade de Tecnologia de Americana
Curso Superior de Tecnologia em Análise de Sistemas e Tecnologia da Informação - Jogos Digitais

Análise do Processamento Gráfico Utilizando a Técnica de *Shaders*

RAFAEL BLUMLEIN CARVALHO

rafab0@hotmail.com

Trabalho de Graduação desenvolvido em cumprimento à exigência curricular do Curso Superior de Tecnologia em Análise de Sistemas e Tecnologia da Informação, sob a orientação do Prof. Esp. Marcos Francisco Pereira da Silva.

Área: Jogos Digitais

Americana, SP
2014

Rafael Blumlein Carvalho

Análise do Processamento Gráfico Utilizando a Técnica de Shaders

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Jogos Digitais pelo CEETEPS/Faculdade de Tecnologia – FATEC/ Americana.
Área de concentração: Jogos Digitais.

Americana, 26 de Junho de 2014.

Banca Examinadora:

Marcos Francisco Pereira da Silva (Presidente)
Especialista
FATEC Americana

José Mário Frasson Scafi (Membro)
Mestre
FATEC Americana

Bruno Darros Lorençon (Membro)
Graduado
FATEC Americana

AGRADECIMENTOS

Em primeiro lugar, gostaria de agradecer aos meus pais que sempre apoiaram minhas escolhas dentro do meio acadêmico, me incentivando e motivando para melhorar cada dia mais. Em segundo lugar aos professores da Fatec, aos professores Vitor e Marcos que ajudaram muito com o desenvolvimento deste trabalho. Também quero agradecer minha namorada que esteve ao meu lado todo este tempo me ajudando com muitas coisas e mesmo nas horas mais difíceis foi paciente comigo. E por último, mas não menos importantes, ao restante dos meus familiares e amigos que estiveram presente ao longo de todos esses anos sempre me apoiando e me dando força para continuar.

DEDICATÓRIA

Dedico este trabalho aos profissionais e aspirantes ao desenvolvimento de jogos no Brasil.

RESUMO

O presente texto visa explorar os principais conceitos sobre os *shaders* e como influenciam no processamento gráfico de um jogo. Com uma abordagem ampla, o estudo contido nesta pesquisa faz uma análise sobre o uso dessas técnicas em jogos e de que maneira são executadas essas funcionalidades. Por fim, por meio de testes de desempenho, é realizada uma verificação do comportamento dessas configurações em diferentes microcomputadores e placas gráficas, demonstrando as métricas e os resultados obtidos.

Palavras Chave: *Shader*; Jogos Digitais; Computação Gráfica; Análise de Desempenho.

ABSTRACT

The present text aims to explore the main concepts about shaders and how they can influence on the graphic process of a game. With a wide approach, the study contained in this research analyzes these techniques on games and shows how these features are implemented. Finally, by means of performance tests, is realized a verification about the behavior of these configurations on different computers and graphic cards, demonstrating the metrics and the obtained results.

Keywords: *Shader; Games; Computer Graphics; Performance Analysis.*

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS E GRÁFICOS

LISTA DE ABREVIATURAS E SIGLAS

API – *Application Programming Interface*

CMYK – *Cyan, Magenta, Yellow, Black*

FPS – *Frames per Second*

MB – *Megabytes*

MHz – *Megahertz*

RGB – *Red, Green, Blue*

SSAO – *Screen Space Ambient Occlusion*

UPG – Unidade de Processamento Gráfico

1 INTRODUÇÃO

Partindo de uma definição estruturada por Salen e Zimmerman (2004, p.95), pode-se definir um jogo da seguinte maneira: “Um jogo é um sistema no qual os jogadores se envolvem em um conflito artificial, definido por regras, que implica um resultado quantificável”. Entende-se, desta maneira, que os jogos têm como intuito principal trazer desafios do cotidiano de uma forma mais interativa, e como resultado final o jogador ganha, perde ou recebe algum tipo de prêmio.

Atualmente os jogos são os principais conteúdos de interação e entretenimento digital, presentes em diversas plataformas como, por exemplo, *mobiles*, *consoles* e computadores. Os jogos digitais comportam grande tecnologia e são encontrados nos mais variados estilos e enredos, tanto 2D quanto 3D, *on-line* ou *off-line*, variavelmente de acordo com o perfil do usuário. Esta é uma tecnologia que vem crescendo, ganhando espaço e reconhecimento no mercado.

Esta pesquisa tem o objetivo de demonstrar os resultados do processamento gráfico de um jogo com o auxílio das técnicas de *shaders*, expor os diferentes tipos de *shaders* e analisar de que maneira os mesmos influenciam no processamento gráfico de um jogo.

Os *shaders* são programas específicos que rodam num certo momento dentro de um jogo digital. Eles se encaixam em um padrão mais avançado nas configurações do jogo, alterando qualidades de sombra, reflexo, água, curvaturas, entre outros que são apresentados no capítulo três. Durante o desenvolvimento do jogo é possível definir as configurações básicas, para que o mesmo possa operar adequadamente em uma grande variedade de equipamentos. No momento dessas configurações, deve-se escolher entre o maior ou menor desempenho e as definições gráficas que serão utilizadas no processamento do jogo. Silva e Scalco (2009) afirmam que, em muitas situações, o programador deve decidir entre qualidade visual ou desempenho quando aplica um determinado algoritmo, o que significa que no momento de aplicar os efeitos mencionados, deve-se identificar o que será melhor para o jogo e de que maneira será utilizado.

Durante a criação dos *shaders* é possível observar que o seu uso está diretamente ligado na alteração da composição de objetos previamente desenvolvidos, aplicando efeitos e tornando-os mais realistas. Considerando que quanto maior a realidade do objeto, maior será o processamento na Unidade de Processamento Gráfico (UPG), afetando diretamente no desempenho da máquina e na performance do jogo. *Consoles* e computadores com processadores e placas de vídeo avançados, conseguem executar jogos com suas configurações máximas sem muitos problemas. Já em equipamentos com menor capacidade de processamento, o ideal é diminuir a qualidade dos gráficos para melhorar o desempenho.

Para melhor entendimento do trabalho, o mesmo foi estruturado em seis capítulos, sendo que o primeiro conceitua os pontos chave de segmentação do trabalho, por meio de embasamentos do tema sugerido. O segundo, terceiro e quarto contêm as informações necessárias para o entendimento do que é abordado durante o estudo de caso, este que é elaborado no capítulo cinco. Mediante todas as informações conseguidas a partir dos estudos realizados, o sexto capítulo se reserva às considerações finais da pesquisa.

2 COMPUTAÇÃO GRÁFICA

A computação gráfica é a área da computação designada à geração de imagens, a mesma é em um conjunto de métodos e técnicas utilizadas que fazem com que a imagem seja reproduzida visualmente em um monitor.

Ela surgiu em meados das décadas de 40 e 50, em consequência de dois projetos militares norte-americanos que visavam a construção de um simulador de voo e um sistema de defesa aéreo contra ataques nucleares (COSTA, 2007, p.7).

> "Em 1960, o designer William Fetter tentava desenvolver um novo processo a fim de maximizar a eficiência dos *layouts* da parte interior dos aviões *Boing*. Seu produto final foi uma visão da forma humana gerada pelo computador em forma ortográfica. Fetter deu o nome de Computação Gráfica para descrever a criação, como ponto de partida a uma corrente de eventos que revolucionariam o mundo do entretenimento, da publicidade e da mídia" (SHKLYAR, 2004, tradução nossa).

A partir do marco inicial, foi possível utilizar a computação gráfica em diversas áreas como, por exemplo, a televisão, a arquitetura, a medicina, a publicidade e os jogos.

Um dos primeiros filmes a utilizar a computação gráfica foi o *Star Wars* de George Lucas em 1977. Foi apenas uma cena de aproximadamente quarenta segundos, porém, demorou diversas semanas para ser concluída.

Segundo Vianna (2002, p.2), por um ponto de vista informal, define-se que: "a computação gráfica é o veículo de comunicação homem/máquina mais adequado à percepção humana". A partir desta definição, é concebida uma representação do processo da computação gráfica seguido de quatro itens, a visão, a visualização e o processamento de imagens e de dados (Figura 1)[1].

[1] Sobre essa representação, ver Viana (2002, p.2).

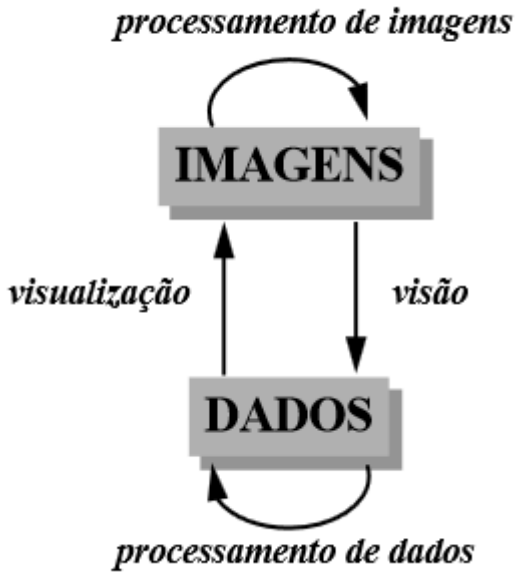

Figura 1 – Processos da computação gráfica (VIANNA, 2002, p.2)

2.1 Conceitos

Para Battaiola (2003, p.5), um aplicativo gráfico é um programa ou um sistema composto de vários programas que permitem a geração de uma determinada apresentação gráfica que pode ser composta de recursos 2D ou 3D.

Entendendo os conceitos básicos de computação gráfica, observa-se a utilização de imagens em dois tipos, *bitmap* e vetorial. As imagens *bitmap* são formadas por informações de ponto a ponto, ou seja, por *pixels*. E as imagens vetoriais são formadas por descrições geométricas e expressões matemáticas, não representadas ponto a ponto (Figura 2).

Imagem Vetorial

Imagem *Bitmap*

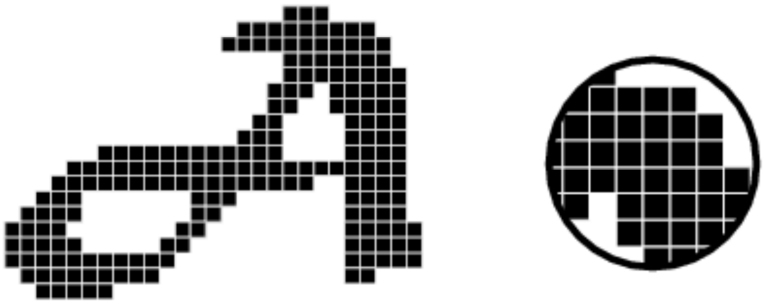

Figura 2 – Comparação entre imagem vetorial e imagem *bitmap* (ADOBE, 2008)

Nos sistemas de cores (Figura 3), os mais utilizados são o RGB (*Red, Green, Blue*, em português Vermelho, Verde, Azul) e o CMYK (*Cyan, Magenta, Yellow, Black*, em português Ciano, Magenta, Amarelo, Preto). O sistema RGB é conhecido como sistema aditivo que é formado pela luminosidade, encontra-se no desenvolvimento de *websites*, em monitores e TVs. Já o sistema CMYK funciona com a absorção de luz, ou seja, subtrativo e é aplicado em impressões como revistas e embalagens, pois reproduzem a maioria das cores do espectro visível.

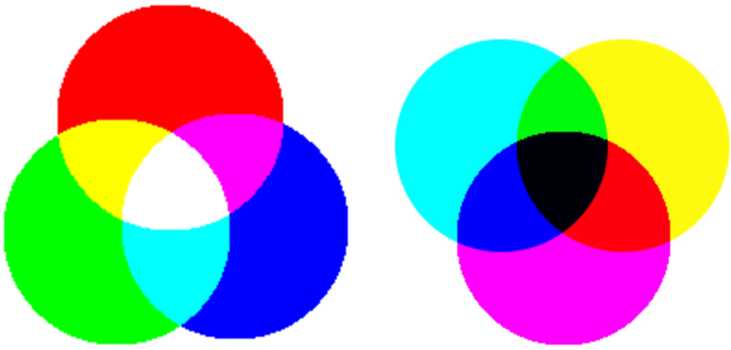

Figura 3 – Sistema de cores RGB e CMYK

No caso dos modelos 3D, apresentam-se como conceitos básicos os processos para construção do mesmo, passando pela modelagem, texturização, iluminação e posterior animação do objeto caso haja necessidade. Com a utilização de *software* específicos[2] para a criação destes objetos, é possível desenvolver qualquer tipo de modelo 3D.

2.2 Aplicação

Atualmente, a utilização da computação gráfica está presente em diversas áreas e em muitos dispositivos, podem-se citar alguns exemplos como simuladores de todos os tipos (voo, acidentes, carro, etc.), na arquitetura com maquetes eletrônicas, em filmes de todas as categorias que utilizam efeitos especiais, e principalmente nos jogos digitais.

A aplicação pode ser diretamente relacionada com a interação de formas primitivas geométricas, desenhos e até mesmo a realidade virtual, que estão presentes dentro de todos os jogos digitais. Tendo como marco inicial o primeiro jogo conhecido como Tênis Para Dois de 1958 do pesquisador William Higinbotham (Figura 4), vindo até os dias de hoje com *Just Cause* 2 publicado em 2010 pela empresa *Square Enix* (Figura 5).

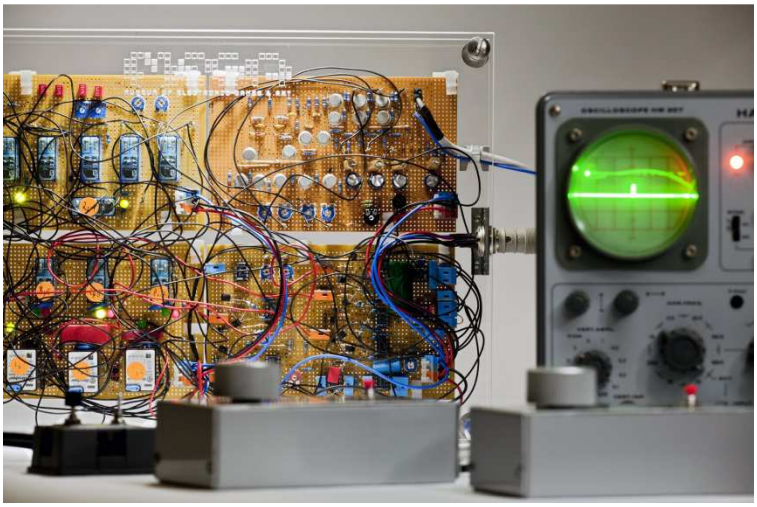

Figura 4 – Jogo Tênis Para Dois (MEGA, 2012)

[2] Alguns exemplos de *software* livres são: *Blender, Google SketchUp, BRL-CAD* e o *eDrawings*.

Figura 5 – Jogo *Just Cause* 2 (SQUARE ENIX, 20120)

2.3 Placas de Vídeo

A placa de vídeo é um componente do computador que inicialmente controla a saída de dados para o monitor, além de recursos de aceleração e de processos 3D.

Como descrito por Morimoto (2007a), originalmente as placas de vídeo eram apenas dispositivos simples que mostravam o conteúdo da memória de vídeo no monitor. Em seguida passaram a suportar recursos de aceleração à máquina, que permitem fazer coisas como mover janelas ou processar arquivos de vídeo de forma a aliviar o processador principal. Por fim, foi implementado o suporte para recursos 3D, capacitando a geração de processar e exibir imagens em 3D, que são formadas por uma grande quantidade de polígonos, posicionados no espaço tridimensional e texturas aplicadas sobre eles.

2.3.1 Placas 3D

As placas com suporte 3D são consideradas a última geração de placas de vídeo, e são as mais específicas para o estudo deste trabalho. Elas que influenciam diretamente no processamento gráfico, têm como objetivo acelerar e aperfeiçoar a experiência do jogador em jogos 3D, além de aplicativos profissionais e até mesmo nos sistemas operacionais.

De acordo com Morimoto (2007b), as placas 3D possuem processadores dedicados, cuja função é unicamente processar imagens, deixando o processador livre para executar outras tarefas.

O processo de criação de imagens 3D é dividido em três partes. Na primeira etapa, é criada uma descrição dos objetos que compõem a imagem. Depois, inicia-se a fase de geometria, onde a imagem é efetivamente criada e armazenada na memória da placa 3D. Por último o *rendering* ou representação da imagem que consiste em transformar as informações armazenadas na memória em uma imagem bidimensional que será exibida no monitor.

2.3.2 Placas *Onboard*

As placas de vídeo *onboard*, são aquelas que já estão acopladas na placa mãe do computador, facilitando na compra de um equipamento mais simples por um custo mais baixo. De acordo com Brito (2013), por este motivo estas placas se tornaram mais comuns e são reconhecidas mais facilmente pelo sistema operacional, além de consumirem menos energia. Porém, a capacidade dessas placas são inferiores se comparadas às o*ffboard* e não são passíveis para troca, pelo fato de serem incorporadas na placa mãe.

2.3.3 Placas *Offboard*

As placas de vídeo *offboard*, são aquelas compradas separadamente de um computador convencional, instaladas em um *slot* separado. Estas placas funcionam com um processador independente, o que ajuda no desempenho do computador, além de serem as responsáveis por recursos 3D e garantir mais desempenho do que uma placa o*nboard*.

Existem diferentes modelos e marcas de placas o*ffboard*, entretanto, a escolha depende apenas do perfil e da necessidade do comprador.

3 *SHADERS*

Os *shaders* são criados a partir de pequenos programas de baixo nível, ou seja, utilizando programações que abrangem as propriedades da arquitetura do computador, onde são compilados e executados em fases específicas do *pipeline*[3] gráfico (MICROSOFT, 2013a). Pode-se dizer também que um *shader* é um programa designado a rodar num certo momento pelo processador gráfico. Seu propósito é executar um dos estágios programáveis na fase de *rendering* dos *pipelines* (OPENGL, 2013).

O termo *shader* foi introduzido por Cook (1984) para designar um conjunto de procedimentos de modificações de propriedades de geometria e fragmentos para simulação de diferentes materiais em síntese de imagens.

Identifica-se desta maneira, que os *shaders* são utilizados na programação das UPGs. Com um conjunto de instruções enviadas à UPG, o objetivo é alcançar efeitos visuais, como por exemplo, um efeito mais real à água e alterações na iluminação (REIS, 2010, p.37). O mesmo autor aponta que na introdução das placas gráficas no mercado, estes equipamentos não eram programáveis, ou seja, não havia maneiras de se enviar um conjunto de instruções para realizar determinada operação, tinham o objetivo único de acelerar o processamento das máquinas. Porém, com o avanço da tecnologia destas placas, bibliotecas como *OpenGL* e *DirectX* passaram a introduzir recursos para programar as novas UPGs, adicionando funções especiais em suas APIs (*Application Programming Interface*, em português Interface de Programação de Aplicativos).

[3] O *pipeline* é o caminho responsável pela transição de informações associado aos *pixels* que constroem a imagem, ele oferece maior velocidade de processamento dependendo de seu tamanho.

De acordo com Ahearn (2008 apud GERARD, 2013, p.55) os *shaders* permitem um nível de realismo em jogos que estão ficando cada vez melhores. Observando paisagens e até mesmo as características humanas, podendo citar como exemplo os jogos de futebol, que a cada lançamento estão aprimorando estes detalhes. É possível verificar esta comparação entre jogos da mesma franquia em certo intervalo de tempo entre os lançamentos, como exemplo FIFA *WorldCup* 2006 (Figura 6) e FIFA 2014 (Figura 7).

Figura 6 – Exemplo de gráficos no jogo FIFA *WorldCup* 2006 para PC (EA, 2009)

Figura 7 – Exemplo de gráficos no jogo FIFA 2014 para PC (EA, 2013)

Existem ainda interfaces que facilitam na criação e visualização de efeitos de *shader*, como o RenderMonkey (AMD, 2006), ferramenta dinâmica que possibilita que programadores e artistas possam trabalhar de maneira colaborativa com este tipo de aplicação.

3.1 Vantagens e Desvantagens

A vantagem na utilização dos *shaders* está diretamente ligada à visualização dos ambientes e personagens dentro de um jogo, por meio de códigos, sem a necessidade de modificação da malha tridimensional dos objetos.

Com a utilização de efeitos aos materiais como cabelos, cicatrizes, grama, luzes, sombras, água e ondulações, torna-se possível construir cenas e personagens mais realísticos (NVIDIA, 2001a).

As desvantagens perante os *shaders* foram encontradas nas placas gráficas mais antigas, por não suportarem processadores gráficos programáveis, sendo assim, não conseguem processar os *shaders*.

Na utilização de máquinas mais recentes, mas com placas de baixo desempenho ou placas *onboard* sem suporte a 3D, alguns jogos podem apresentar instabilidade ou lentidão caso certas configurações que utilizam *shaders* estiverem habilitadas, tornando ineficaz a experiência que estes pretendem passar ao jogador (Figura 8).

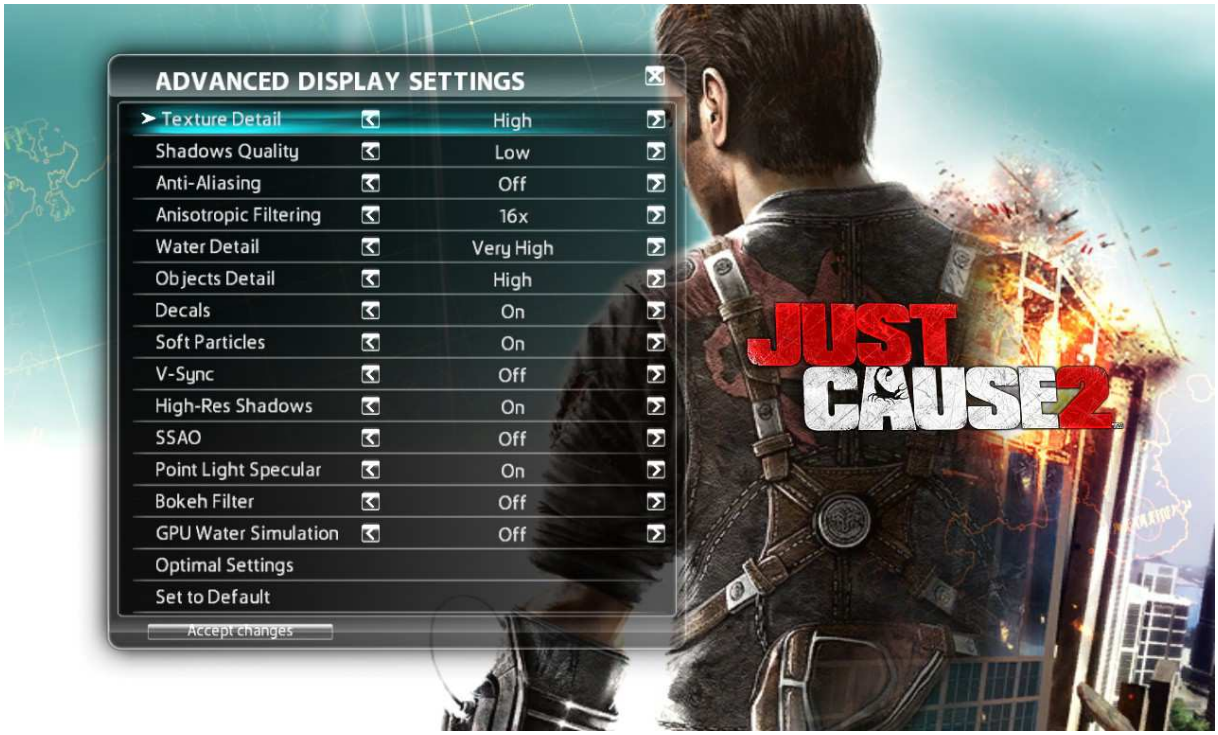

Figura 8 – Menu de configurações do jogo *Just Cause* 2 (SQUARE ENIX, 2010)

3.2 Tipos de *Shaders*

Existem diversos tipos de *shaders*, cada um com sua função única, porém, dependente das outras para que o objetivo final do projeto seja o mais adequado possível. A seguir serão listados alguns dos tipos de *shaders*, o *vertex shader*, o *geometry shader* e o *fragment shader*.

Durante a pesquisa foram encontrados outros tipos *shaders* que são utilizados na *tesselation*[4] (em português, tesselação), nova tecnologia disponível para *DirectX* 11 e *OpenGL* 4.0 APIs, são o *hull shader* e o *domain shader*[5] (NVIDIA, 2010).

Abaixo segue uma representação do *pipeline* gráfico (Figura 9).

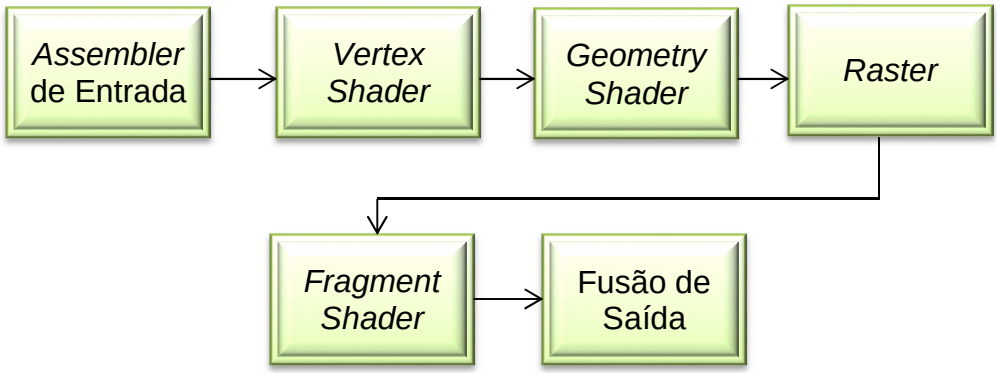

Figura 9 – Representação do *pipeline* gráfico

O início do *pipeline* gráfico indica ao programa a posição, o índice e as coordenadas que serão utilizadas, chamado de *assembler* de entrada. Segue então para o *vertex shader*, que é executado uma vez para cada vértice aplicando constantes de projeção, seguido do *geometry shader*, responsável por replicar as primitivas previamente adicionadas pelo *vertex shader*. Após esta etapa, inicia-se o que é conhecido por *rastering* onde transforma a cena 3D, descrita como polígonos e transforma-a em uma superfície 2D. Logo em seguida, o *fragment shader* é responsável por adicionar as cores e texturas aos objetos, executando em cada *pixel* gerado pela imagem. Por último, é executada a combinação de todos os resultados obtidos e exibe na tela todo o processo combinado, etapa conhecida como fusão de saída.

3.2.1 *Vertex Shader*

Conforme Löwgren (2010, p.10, tradução nossa) “O *vertex shader* é o primeiro *shader* programável e é chamado uma vez para cada ponto de vértice na malha.” Por exemplo, um triângulo consiste em três vértices, e cada um desses vértices descreve onde duas arestas do polígono se encontram. Além da posição dos pontos,

[4] “A tesselação é um método que quebra polígonos em peças mais finas” (NVIDIA, 2010).
[5] Mais sobre *hull* e *domain shader* na referência (MICROSOFT, 2013b).

os vértices possuem outras informações pertinentes, como cores e as coordenadas de textura.

Eles não costumam mudar o tipo de dados, simplesmente mudam valores dentro de cada vértice para alterar efeitos. A partir destes pontos é possível adicionar vários efeitos como transformações do objeto, luz, névoa, ondas de calor, aspecto de movimento e coloração para cada vértice e enviá-los para o *pipeline*.

Os *vertex shaders* permitem realizar também a deformação de superfícies, tecidos (Figura 10) e alteração nas matrizes de pele, oferecendo aos desenvolvedores uma gama de possibilidades para criar personagens e cenários, intensificando a experiência gráfica (NVIDIA, 2001b).

Figura 10 – Simulação da utilização do *vertex shader* com tecidos (BROOKER, 2000)

3.2.2 *Geometry Shader*

O segundo *shader* a ser executado no *rendering* do *pipeline* é o *geometry shader*, ele recebe os vértices diretamente do *vertex shader* caso não existam *shaders* de *tesselation* ativos.

> "Uma das características do *geometry shader* é a possibilidade de enviar uma primitiva para a placa de vídeo e receber várias outras primitivas. Na prática, isso permite a elaboração de um único vértice, representando uma cena triangular e escrevendo diversos valores para diferentes posições no *framebuffer*." (SANTOS, 2009, p.46, tradução nossa).

De acordo com Löwgren (2010, p.10), uma habilidade única do *geometry shader* é fornecer ao programador a possibilidade de criar ou destruir objetos na malha.

“Os *geometry shaders* podem trazer também os dados de vértice para as primitivas de ponta-adjacente como entrada (um adicional de dois vértices de uma linha, mais três por um triângulo)” (MICROSOFT, 2013a, tradução nossa).

A ilustração abaixo mostra um exemplo de entrada das primitivas que são utilizadas no *geometry shader*, representadas por um triângulo e uma linha com seus vértices adjacentes (Figura 11).

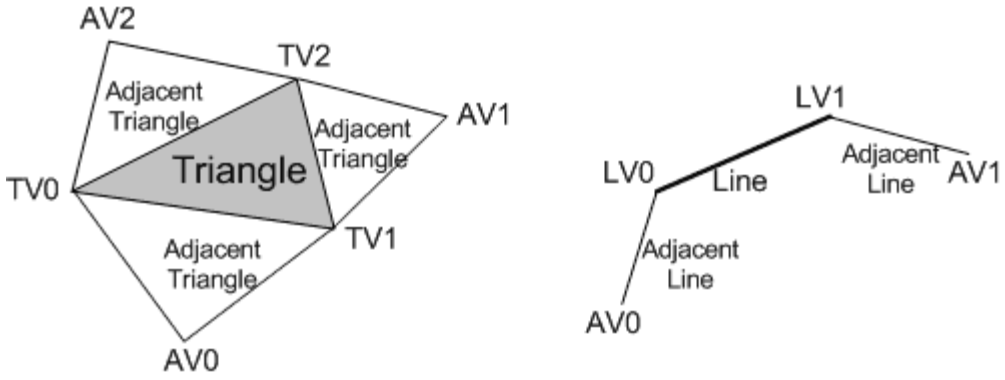

Figura 11 – Triângulo e linha com vértices adjacentes (MICROSOFT, 2013a)

Habitualmente, para construir um objeto adicional, seriam necessários mais arquivos sendo enviados para o *pipeline*, aumentando o processamento da máquina. Utilizando o *geometry shader* é possível gerar novas primitivas a partir das que já foram enviadas, economizando os esforços da máquina e compondo novas imagens somente com o uso da UPG.

A utilização do *geometry shader* pode ser vista em pelos, cabelos (Figura 12), deformações na pele no corpo de um objeto, gramas, nuvens, entre outros.

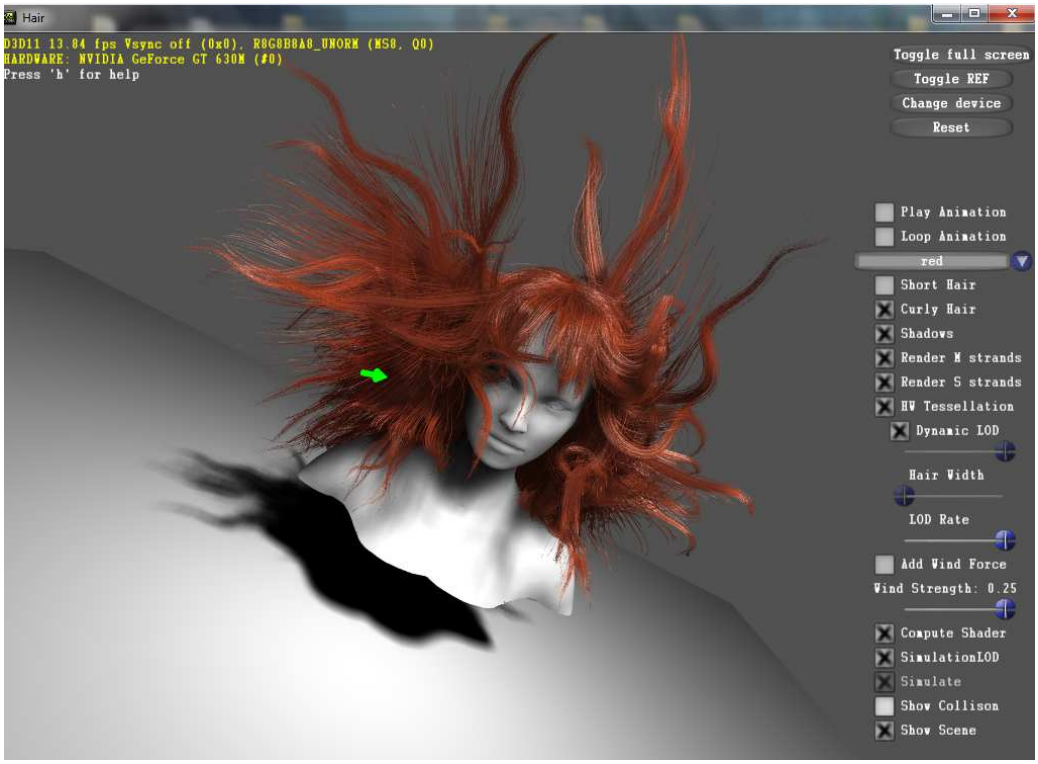

Figura 12 – Exemplo do *geometry shader* com cabelos (NVIDIA SDK, 2011)

E uma comparação na utilização de primitivos com os *geometry shaders* em malha (Figura 13) e logo após com os efeitos de *shaders* habilitados (Figura 14), também utilizando o NVIDIA Direct3D SDK 11 para demonstração.

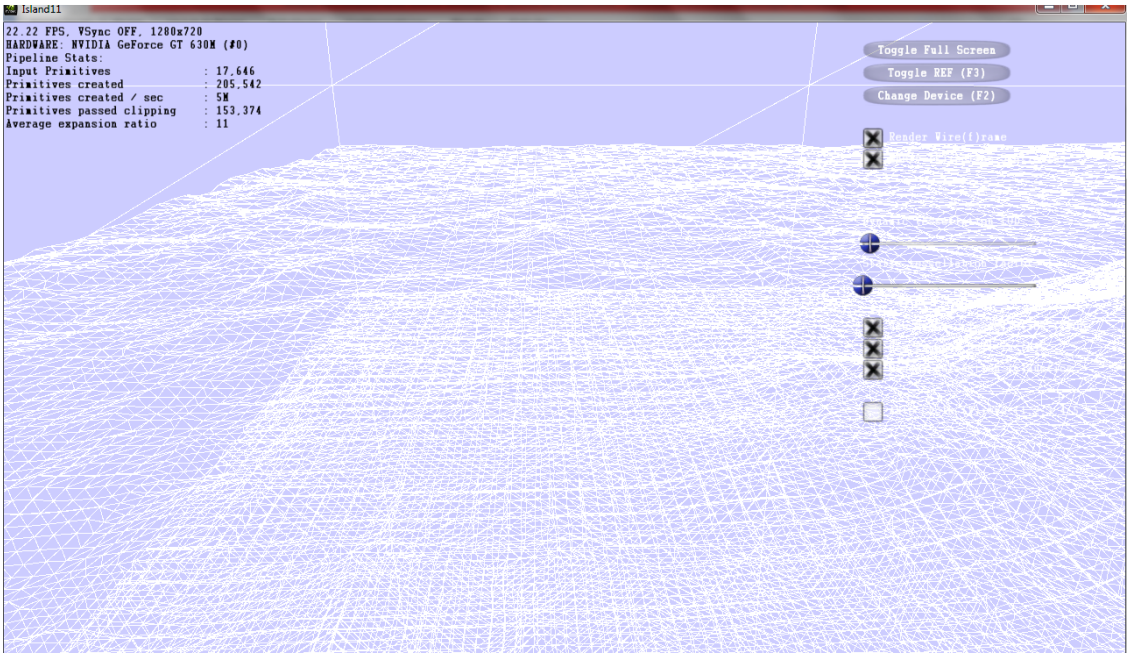

Figura 13 – Malha demonstrando a utilização do *geometry shader* (NVIDIA SDK, 2011)

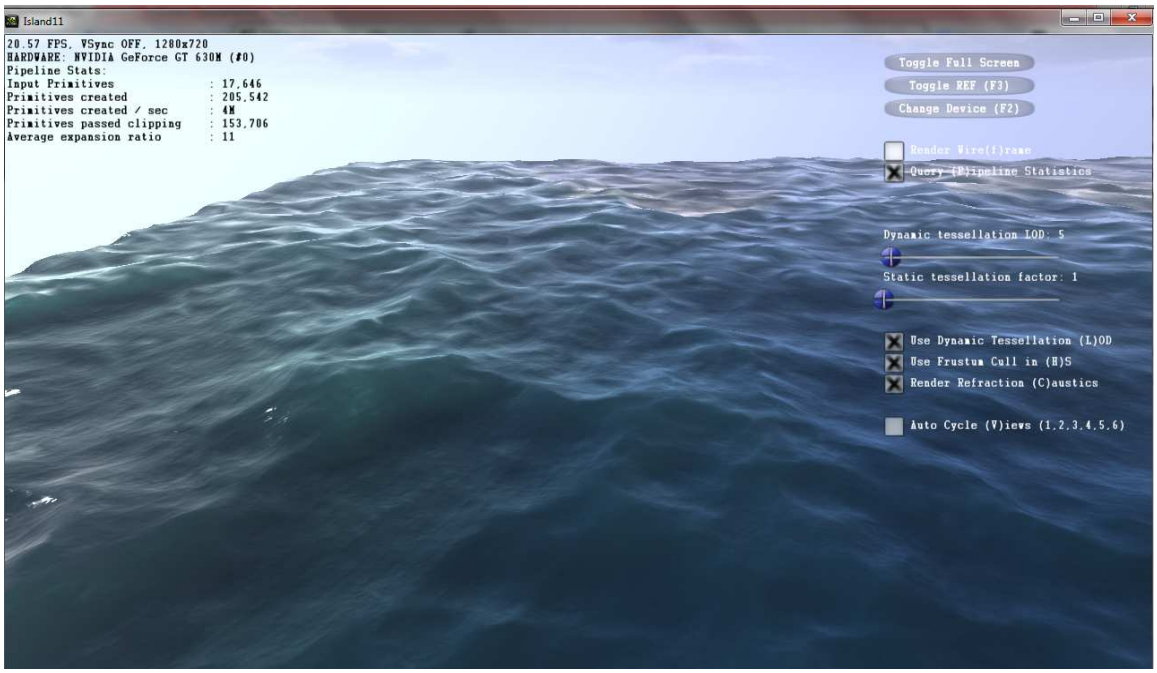

Figura 14 – Efeito dos *shaders* aplicados à malha (NVIDIA SDK, 2011)

3.2.3 *Fragment Shader*

Para entendimento desta etapa do *pipeline*, será utilizado o nome *fragment shader*, como base de referência o livro de Benstead, Astle e Hawkins (2009, p.125), no qual dizem que são conhecidos também como *pixel shaders*. Este que opera por nível de *pixel* e é responsável por calcular a cor de um *pixel* (LÖWGREN, 2010, p.10), além de atuar também na textura de um objeto. Quando a palavra textura é utilizada, entende-se como uma imagem por cima do objeto que o represente como na vida real, atualmente utiliza-se este termo para realçar algo que seja sentido. Ele tenta simular a verdadeira realidade dos objetos dentro dos jogos, com isso concede ao jogador uma qualidade visual notável e agradável.

Por exemplo, uma rocha com grãos de areia por cima dela ou na sombra de um barco sobre a água (Figura 15).

Esses efeitos incluem recursos como o *bump mapping* (Figura 16) ou *parallax mapping*, técnicas utilizadas para fazer com que texturas planas apareçam com efeito de profundidade e adicionando efeitos de luz ou sombra sobre a superfície (CHRISTIAN, 2010).

Figura 15 – *Fragment shader* em sombras no jogo *Age of Empire* 3 (MUNDO DIGITAL, 2012)

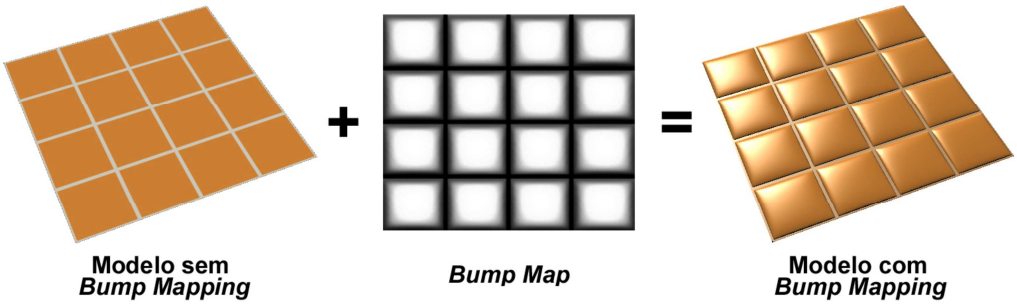

Figura 16 – *Bump mapping* (REIS, 2010, p.41)

4 ANÁLISE DE DESEMPENHO

Análises de desempenho são técnicas utilizadas mediante uma proposta inicial e um objeto de estudo. Servem para analisar o comportamento e a execução de uma atividade específica, colocando como pontos chave características previamente apontadas, examinando e coletando dados para posterior comparação entre outros objetos do mesmo nicho.

Estas análises apresentam o comportamento da aplicação, de acordo com os fatores que se deseja analisar. Como descrito por Jain (1991 apud RAEDER, 2007, p.29), existem diversas maneiras e ferramentas utilizadas para realizar análises de desempenho, como por exemplo, monitores de *software* ou de *hardware*. Ainda pontua que para escolher entre um dos dois tipos de monitoramento, faz-se necessário verificar o que será analisado, o valor a ser gasto e à velocidade que deseja observar os eventos.

4.1 *Benchmark*

O *benchmark* é um dos modelos mais utilizados para a análise de desempenho em diversas áreas atualmente. Desde a década de 70 quando surgiu com o intuito de aumentar a competitividade de mercado, passou por mudanças e aprimoramentos, dando oportunidade de crescimento para empresas como a *Xerox*, pioneira na utilização desta técnica.

Pode-se definir que o “*Benchmarking* é o processo contínuo de medição de produtos, serviços e práticas em relação aos mais fortes concorrentes, ou às empresas reconhecidas como líderes em suas indústrias [...] ” (CAMP, 1998, p.8 e10 apud ZAGO et al, 2008, p.3). O mesmo autor ainda pontua que o modelo também é plausível para utilizar como padrão para a comparação de outros objetos ou atividades, no caso da área da tecnologia da informação, para medir o desempenho de *hardware* e *software*.

4.2 Ferramentas

Como ferramenta de estudo, foi utilizado o sistema de *benchmark* existente no jogo *Just Cause* 2 (SQUARE ENIX, 2010), em dois mapas diferentes. Esta ferramenta foi escolhida pela sua praticidade, facilidade de uso para os testes e por conter uma gama variada de configurações que podem ser ajustadas. Ao utilizar a ferramenta é verificado o FPS (*frames per second*, em português quadros por segundo) da cena utilizada durante o teste, mediante as configurações previamente estabelecidas.

O FPS é uma unidade de medida que quantifica a frequência que um dispositivo produz em quadros por segundo, ou seja, quanto maior for esta frequência melhor será a gravação ou visualização de toda sequência de quadros na tela a cada segundo. Segundo Machado (2011), como referência para o melhor desempenho dos jogos, utilizam-se as taxas de 30 a 60 FPS, mas isto varia dependendo do *hardware* e do *software* que são utilizados. Sendo assim, o FPS é considerado um fator chave para o estudo de caso apresentado no próximo capítulo.

Para melhor entendimento das configurações disponíveis utilizadas, foi desenvolvida uma breve análise sobre cada uma delas.

- O **desfoque de movimento** é uma técnica utilizada por fotógrafos há bastante tempo, na captura de imagens em grande velocidade na qual ela se torna desfocada ou distorcida (PRADA, 2008). Nos jogos é percebido em cenas de ação e grande velocidade (Figura 17);

Figura 17 – Desfoque de movimento no jogo *Just Cause* 2 (SQUARE ENIX, 2010)

- O **detalhe de textura** basicamente determina as resoluções das texturas usadas no jogo;

- A **qualidade das sombras** é determinada mediante a sua suavidade;

- ***Anti-aliasing*** (em português anti-cisalhamento), é um filtro utilizado para suavizar as bordas de qualquer superfície, gerando uma imagem diferente do tamanho da tela, aumentando assim a quantidade de *pixels* da imagem e disponibilizando o efeito (Figura 18). É possível transformar esses objetos em versões maiores de até 32x, dependendo também da placa de vídeo e suporte do jogo. Quanto maior o nível do efeito mais próximo da perfeição o objeto estará, porém, faz-se necessário atenção ao alterar esta opção, pois é possível ocorrer uma baixa considerável no FPS durante o jogo.

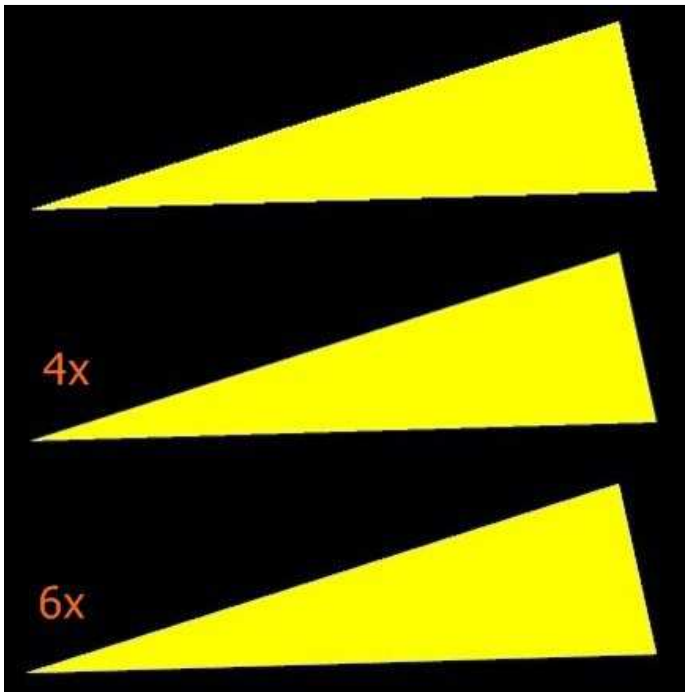

Figura 18 – *Anti-aliasing* (CHAN, 2004)

- O **filtro anisotrópico** é responsável por deixar mais claro e nítido as imagens tridimensionais (NVIDIA, 2012a), melhorando a qualidade das texturas em superfícies que estejam a certa distância e num ângulo diferente em relação ao ponto de vista do jogador. Este filtro foi criado para substituir os antigos filtros bilineares e trilineares (Figura 19). A alteração desta opção pode chegar até 16x e por ter uma pequena penalidade no desempenho, é recomendável utilizá-la no máximo;

Figura 19 – Filtro anisotrópico (NVIDIA, 2012a)

- Nas opções de **detalhes da água** ocorre o controle da aparência da água, incluindo a resolução da textura e as partículas de densidade;

- Nos **detalhes de objetos**, há influência direta em menores detalhes na malha de alguns objetos;

- Os ***decals*** (em português decalque) são objetos que atingem uma superfície já existente e exibem algo diferente sobre ela. Andrade (2011, p.11) afirma que estes decalques podem ser implementados por meio de uma pequena textura e aplicados em tempo real, exibindo efeitos de buracos de bala numa parede, pegadas, respingos de sangue e marcas de queimadura, entre outros (Figura 20);

Figura 20 – *Decals* (SQUARE ENIX, 2010)

- Na opção de **partículas suaves**, cria-se uma suavização na borda das partículas de efeito quando estas se cruzam com as geometrias, eliminando a borda irregular;

- ***V-sync*** (em português sincronização vertical) é a opção que visa sincronizar a atualização de quadros gerados pela placa de vídeo com a frequência definida para o monitor, como por exemplo, 60, 75 ou 80 *Hertz* (PANKIEWICZ, 2009). O objetivo do *V-sync* é não deixar que aconteça o *screen tearing* (em português tela rasgada), problema na qual a imagem é recortada durante a execução da cena. É uma difícil decisão entre habilitar ou não está opção, visto que quando em operação pode diminuir consideravelmente o FPS do jogo, restando escolher entre o desempenho ou pelo *screen tearing*;

- A opção de **sombras de alta resolução** aumenta a qualidade do efeito e fornece um cálculo mais preciso na projeção da sombra;

- **SSAO** (*Screen Space Ambient Occlusion*, em português Oclusão Ambiente de Espaço na Tela) é um modelo de luz que calcula o brilho de um *pixel* em relação a objetos próximos da cena (NVIDIA, 2012b). Estes efeitos são apresentados em tempo real assim que ocorre o pós-processamento da imagem. É possível analisar a comparação da funcionalidade do SSAO com o

exemplo em uma cena (Figura 21), à esquerda adicionando o efeito e à direita sem o efeito;

Figura 21 – SSAO ativado e desativado (UNITY, 2013)

- A opção de **ponto de luz especular** permite apontar luzes nos objetos, realçando aspectos dos mesmos, contribuindo principalmente no brilho e no reflexo da imagem;

- O **filtro de *bokeh*** é uma técnica de profundidade de campo, que produz um efeito de desfoque em luzes distantes (Figura 22). Esta opção é disponível apenas para placas gráficas NVIDIA e influencia somente nas cinemáticas do jogo;

Figura 22 – *Bokeh* ativado e desativado (SOLIDLYSTATED, 2010)

- A opção de **simulação de água da UPG** também é exclusiva para placas gráficas da NVIDIA, na qual deforma a superfície da água simulando o realismo do movimento da água do mar.

A utilização da ferramenta GPU-Z (2007) também foi necessária para a composição dos dados analisados, esta que fornece informações da placa gráfica utilizada pela máquina. Foram recolhidos apenas os seguintes parâmetros de teste dentre todos que o GPU-Z fornece: o *Core Clock* da UPG (MHz) e o uso de memória (MB). Na qual o *Core Clock* realiza a verificação do desempenho e velocidade da UPG enquanto os processos estão em análise, e o uso da memória, refere-se à quantidade total de memória utilizada. Optou-se por estes dois fatores, pois são os mais específicos e que influenciam diretamente nos testes realizados.

5 ESTUDO DE CASO

O estudo de caso se baseia na comparação do desempenho de *hardware* de diversas máquinas, utilizando as ferramentas e parâmetros citados no capítulo anterior.

Para realizar os testes, foi definido que as máquinas necessitavam conter diferentes configurações de *hardware* e placas gráficas, porém, como único requisito do jogo que utilizassem o sistema operacional *Windows Vista* ou *Seven*. Segue abaixo as especificações de cada uma delas (Tabela 1).

Tabela 1 – Configurações das máquinas

Máquina	Processador	Sistema Operacional	Memória RAM	Placa Gráfica	Resolução
M1	Intel(R) Core(TM) i5-3210M CPU @ 2.50GHz	Windows 7 64-bit build 7601 [Service Pack 1]	6013 MB	NVIDIA GeForce GT 630M | 1024MB	1366x768
M2	Intel(R) Core(TM) 2Duo CPU E4600 @ 2.40GHz	Windows 7 32-bit build 7601 [Service Pack 1]	4096 MB	NVIDIA GeForce GT 430 | 1024MB	1280x1024
M3	Intel(R) Core(TM) i7-3630QM CPU @ 2.40GHz	Windows 7 64-bit build 7601 [Service Pack 1]	16384 MB	NVIDIA GeForce GT 650M | 2048MB	1366x768
M4	Intel(R) Core(TM) 2Duo CPU E7500 @ 2.93GHz	Windows 7 64-bit build 7601 [Service Pack 1]	4096 MB	NVIDIA GeForce 210 | 1024MB	1280x1024

Fonte: Autoria própria

Para a primeira fase de testes (Tabela 2) (Tabela 3), foram utilizadas as configurações maximizadas para alta resolução: Desfoque de movimento ligado; Detalhe de textura alto; Qualidade das sombras alto; *Anti-aliasing* 32x CSAA; Filtro anisotrópico 16x; Detalhes da água muito alto; Detalhes de objeto muito alto; *Decals* ligado; Partículas suaves ligado; *V-sync* ligado; Sombras de alta resolução ligado; SSAO alto; Ponto de luz especular ligado; Filtro de *bokeh* ligado; Simulação de água da UPG ligado.

Tabela 2 – Resolução alta primeiro teste

Máquina	**Média do FPS**	**Média da UPG *Core Clock* (MHz)**	**Média do Uso de Memória (MB)**	**Tempo Decorrido (seg)**
M1	13.26	797.3	440.85	140.27
M2	9.45	699.5	492.60	191.04
M3	19.39	683	581.70	123.56
M4	1.15	580	524.89	1559.99

Fonte: Autoria própria

Tabela 3 – Resolução alta segundo teste

Máquina	Média do FPS	Média da UPG *Core Clock* (MHz)	Média do Uso de Memória (MB)	Tempo Decorrido (seg)
M1	17.88	797.3	455	118.94
M2	13.30	699.5	507.61	134.67
M3	18.96	520.16	602.58	121.54
M4	-	-	-	-

Fonte: Autoria própria

Após a realização dos testes em alta qualidade, verificou-se que houve grande diferença no FPS, principalmente no primeiro teste, pontuando a máquina três que contém o melhor processamento, atingindo 19.39 de FPS. Já a máquina quatro, por conter uma placa gráfica inferior, processou o primeiro teste com muita lentidão e diversos erros de *rendering*, e houve falha na aplicação na tentativa de realizar o segundo teste. A partir disso, foi analisado que quanto melhor o processamento mais rápido será realizado o teste, tendo em vista que o ambiente aplica as configurações em tempo real, e que sem uma placa gráfica adequada não é possível processar o jogo com excelência.

Para a segunda fase de testes (Tabela 4) (Tabela 5), foram utilizadas as configurações otimizadas para média resolução: Desfoque de movimento ligado; Detalhe de textura alto; Qualidade das sombras médio; *Anti-aliasing* 4x; Filtro anisotrópico 16x; Detalhes da água médio; Detalhes de objeto médio; *Decals* ligado; Partículas suaves ligado; *V-sync* ligado; Sombras de alta resolução desligado; SSAO baixo; Ponto de luz especular desligado; Filtro de *bokeh* desligado; Simulação de água da UPG desligado.

Tabela 4 – Resolução média primeiro teste

Máquina	Média do FPS	Média da UPG *Core Clock* (MHz)	Média do Uso de Memória (MB)	Tempo Decorrido (seg)
M1	25.62	712.42	277.41	120.32
M2	21.24	699.5	399.80	120.13
M3	41.10	677.85	430.41	120.07
M4	2.94	580	354.57	610.08

Fonte: Autoria própria

Tabela 5 – Resolução média segundo teste

Máquina	Média do FPS	Média da UPG *Core Clock* (MHz)	Média do Uso de Memória (MB)	Tempo Decorrido (seg)
M1	35.68	661.5	291.5	118.27
M2	31.21	699.5	397.83	118.23
M3	48.90	587.92	442.87	118.22
M4	3.94	580	358.21	449.15

Fonte: Autoria própria

Com os testes em resolução média, verificou-se que as máquinas um, dois e três executaram os testes dentro dos padrões para um bom desempenho, mantendo o FPS acima de 30, que é considerado o valor ideal mínimo para que não ocorram instabilidades. Porém, assim como no teste em alta resolução, a máquina quatro apresentou dificuldades no processamento, desta vez conseguindo efetuar o segundo teste, mas com muita lentidão se comparada com as outras máquinas.

Para a terceira fase de testes (Tabela 6) (Tabela 7), foram utilizadas as configurações minimizadas para baixa resolução: Desfoque de movimento desligado; Detalhe de textura baixo; Qualidade das sombras baixo; *Anti-aliasing* desligado; Filtro anisotrópico 2x; Detalhes da água baixo; Detalhes de objeto baixo; *Decals* desligado; Partículas suaves desligado; *V-sync* desligado; Sombras de alta resolução desligado; SSAO desligado; Ponto de luz especular desligado; Filtro de *bokeh* desligado; Simulação de água da UPG desligado.

Tabela 6 – Resolução baixa primeiro teste

Máquina	**Média do FPS**	**Média da UPG *Core Clock* (MHz)**	**Média do Uso de Memória (MB)**	**Tempo Decorrido (seg)**
M1	56.81	797.3	224.92	120.04
M2	42.56	699.5	260.90	120.01
M3	87.33	835.3	326.64	123.83
M4	6.10	580	274.49	294.47

Fonte: Autoria própria

Tabela 7 – Resolução baixa segundo teste

Máquina	Média do FPS	Média da UPG *Core Clock* (MHz)	Média do Uso de Memória (MB)	Tempo Decorrido (seg)
M1	80.75	757.67	238.92	118.20
M2	65.49	699.5	277.48	118.21
M3	109.30	683.89	339.62	118.20
M4	9.17	580	290.09	193.11

Fonte: Autoria própria

Nesta fase, identifica-se uma grande melhora no desempenho dos testes, atingindo mais de 60 FPS para as três primeiras máquinas, possibilitando uma experiência de jogo estável e sem nenhum tipo de oscilação. A máquina quatro, mesmo com as configurações mínimas, não conseguiu atingir um FPS maior do que 10, não sendo possível jogar com estabilidade, gerando bastante lentidão. Apesar de não conter uma visualidade cativante como acontece com a resolução máxima, é recompensado em performance.

Para uma melhor visualização de todos os testes, foram compactados os dados de FPS adquiridos em um gráfico de barras, partindo dos testes em máxima resolução até os de mínima (Gráfico 1).

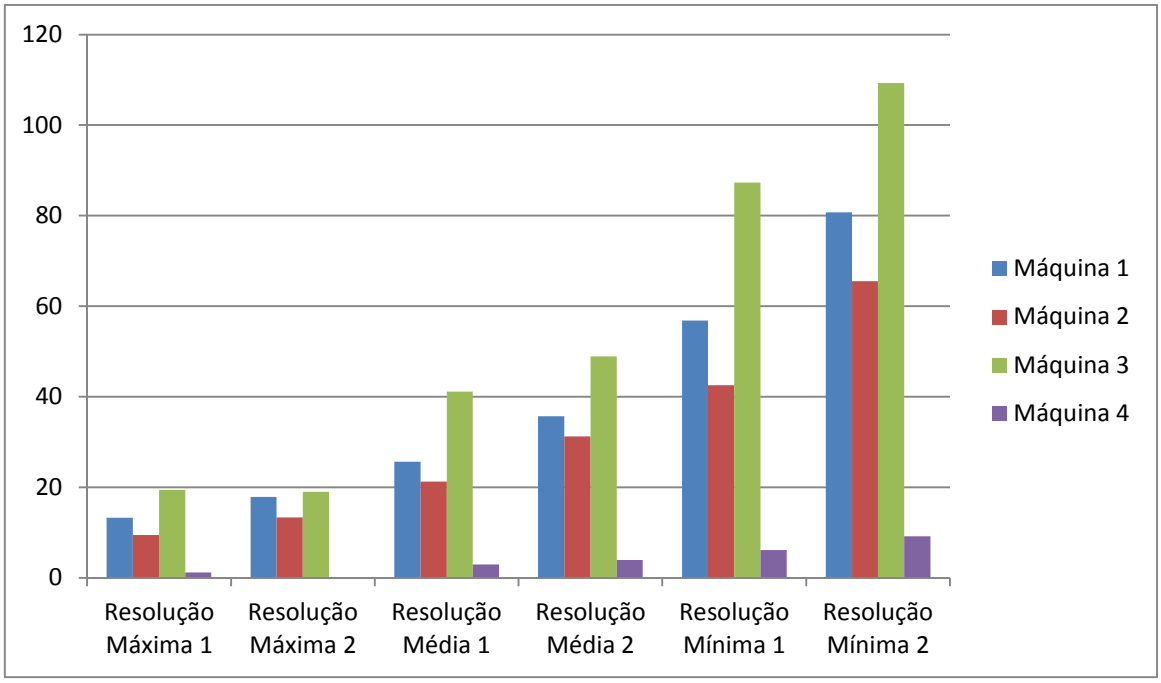

Gráfico 1 – Comparação do FPS

6 CONCLUSÃO

A partir da apresentação e análise dos dados, observa-se que, a utilização dos *shaders* nos jogos influencia diretamente no processamento gráfico de uma máquina, e que quanto melhor o *hardware* utilizado, melhor será o desempenho dos efeitos, trazendo assim, uma melhor experiência ao jogador. A questão mais importante no momento de efetuar os testes, diz respeito ao FPS, podendo ser considerado como ponto chave na avaliação do desempenho das máquinas.

Desta forma, é possível verificar todos os testes, podendo ressaltar individualmente o desempenho de cada máquina. A primeira máquina, considerada como a de médio desempenho, conseguiu atingir um bom processamento em todos os testes, oferecendo experiência de jogo adequada sem muitas oscilações. A segunda máquina, que contém uma placa gráfica mais antiga, alcançou um desempenho razoável, mesmo com as configurações máximas, gerou um ambiente com possibilidade de jogo, considerando uma melhora nos testes de média resolução. A terceira máquina, classificada como a melhor entre as quatro, obteve um bom desempenho na maioria dos testes, principalmente com as configurações mínimas que ultrapassou dos 100 FPS, porém, nas configurações máximas não alcançou 30 FPS, o que seria o desejável para melhor processamento. Por último, a quarta máquina, não obteve um desempenho adequado para considerar uma experiência de jogo efetiva, na qual o FPS não passou de 10 em nenhum dos testes, salientando os de resolução máxima, que não conseguiu executar com primor.

Sabe-se que a cada ano as placas gráficas e máquinas investem em melhorias, disponibilizando novos aprimoramentos e funcionalidades, e em algum momento as configurações estudadas não afetarão mais tanto no desempenho e no processamento dessas atividades.

Como proposta para trabalhos futuros, destaca-se o desenvolvimento de *shaders* aplicados a algum jogo previamente produzido, verificando a otimização dos mesmos e efetuando também testes de *hardware*.

7 REFERÊNCIAS BIBLIOGRÁFICAS

ADOBE *Systems Incorporated*. *Bitmaps*. 2008.
Disponível em:
<http://help.adobe.com/en_US/Director/11.0/help.html?content=06_bitmaps_01.html>
Acesso em: maio de 2014.

AMD *Inc*. *RenderMonkey*. 2006.
Disponível em: <http://developer.amd.com/tools-and-sdks/archive/legacy-cpu-gpu-tools/rendermonkey-toolsuite/>
Acesso em: maio de 2014.

ANDRADE, Felipe Araújo de. Aplicação de *decals* sobre superfícies geométricas arbitrárias. 2011.
Disponível em:
<http://www.lume.ufrgs.br/bitstream/handle/10183/36855/000819126.pdf?sequence=1>
Acesso em: junho de 2014

BATTAIOLA, André Luiz. Apostila do Curso de Computação Gráfica. 2003.
Disponível em: <http://www2.dc.ufscar.br/~saito/download/comp-grafica/apostila.pdf>
Acesso em: novembro de 2013.

BENSTEAD, Luke; ASTLE, Dave; HAWKINS, Kevin. *Beginning OpenGL game programming, Second Edition*. *Boston*: *Course Technology*, *a part of* Cengage *Learning*, 2009.

BRITO, Edivaldo. O que é placa de vídeo *onboard* e *offboard* e qual a diferença entre elas?. 2013.
Disponível em: <http://www.techtudo.com.br/noticias/noticia/2013/08/o-que-e-placa-de-video-onboard-e-offboard-e-qual-diferenca-entre-elas.html>
Acesso em: abril de 2014.

BROOKER, Wayne; RICHARDS, Martyn M. Radeon 8500/7500 *Review*. 2000.
Disponível em: <www.3dvelocity.com/reviews/R200/smartshader_2.htm>
Acesso em: maio de 2014.

CHAN, Eric. *Fast Antialiasing Using Prefiltered Lines on Graphics Hardware*. 2004.
Disponível em: <http://people.csail.mit.edu/ericchan/articles/prefilter/>
Acesso em: junho de 2014

CHRISTIAN, Matthew. *Understanding Pixel and Vertex Shaders*. 2010.
Disponível em:
<http://www.insidegamer.org/documents/Understanding%20Pixel%20and%20Vertex%20Shaders.pdf>.
Acesso em: março de 2014

COOK, Robert L. *Shade Trees*. *In Proceedings of* SIGGRAPH. 1984, p. 223–231, Nova Iorque, 1984. ACM Press. ISBN 0-89791-138-5 doi: 10.1145/800031.808602

COSTA, António Cardoso. História da Computação Gráfica. 2007.
Disponível em:
<http://www.dei.isep.ipp.pt/~jpp/sgrai/Historia.pdf>
Acesso em: novembro de 2013.

EA Inc. 2006 *FIFA WorldCup PC Screenshot*. 2009.
Disponível em:
<http://www.ea.com/uk/2006-fifa-world-cup/images/b8684fb444cd2210EASiteClonerad65140aRCRD>
Acesso em: maio de 2014.

EA Inc. *FIFA* 14 - *Teammate Intelligence*. 2013.
Disponível em:
<www.easports.com/fifa/news-updates-gameplay/article/fifa-14-teammate-intelligence>
Acesso em: maio de 2014.

GERARD, Vinicius Neitzel. Desenvolvimento e construção de jogos em 3D (*triple A games*) com ênfase na modelagem e texturização. 2013.
Disponível em:
<http://ca.ufpel.edu.br/design/digital/tcc/acervo/2012_2/vinicius_gerard_teoria.pdf>
Acesso em: novembro de 2013.

GPU-Z – *the Graphics Card Information Utility*, *version* 0.7.8. TechPowerUp, 2007.

LÖWGREN, Martin; OLIN, Niklas. *PN-triangle tessellation using Geometry shaders: The effect on rendering speed compared to the fixed function tessellator*. 2010.
Disponível em:
<http://www.bth.se/fou/cuppsats.nsf/all/821d8d514d579984c1257737004bde5c/$file/Bachelor_Thesis.pdf>
Acesso em: novembro de 2013.

MACHADO, Jonathan D. O que são *Frames* por Segundo?. 2011.
Disponível em: < http://www.tecmundo.com.br/video/10926-o-que-sao-frames-por-segundo-.htm>
Acesso em: junho de 2014.

MEGA, *Museum of Electronic Games & Art*. 2012.
Disponível em: <http://www.m-e-g-a.org/research-education/research/t42-tennis-for-two/>
Acesso em: maio de 2014.

MICROSOFT *Corporation*. *Shader Stages*. 2013a.
Disponível em:
<http://msdn.microsoft.com/en-us/library/windows/desktop/bb205146%28v=vs.85%29.aspx>
Acesso em: novembro de 2013.

MICROSOFT *Corporation*. *How to use Direct3D 11*. 2013b.
Disponível em:
<http://msdn.microsoft.com/en-us/library/windows/desktop/hh404569%28v=vs.85%29.aspx>
Acesso em: maio de 2014.

MORIMOTO, Carlos. Placa de Vídeo. 2007a.
Disponível em: <http://www.hardware.com.br/termos/placa-de-video>
Acesso em: abril de 2014.

MORIMOTO, Carlos. Placa 3D. 2007b.
Disponível em: <http://www.hardware.com.br/termos/placa-3d>
Acesso em: abril de 2014.

MUNDO DIGITAL. O que é *pixel shader*. 2012.
Disponível em: <http://mundodigital.blogs.sapo.ao/34997.html>
Acesso em: maio de 2014.

NVIDIA *Corporation*. *Pixel Shaders*. 2001a.
Disponível em: <http://www.nvidia.com/object/feature_pixelshader.html>
Acesso em: novembro de 2013.

NVIDIA *Corporation*. *Vertex Shaders*. 2001b.
Disponível em: <http://www.nvidia.com/object/feature_vertexshader.html>
Acesso em: novembro de 2013.

NVIDIA *Corporation*. *Tesselation*. 2010.
Disponível em: <http://www.nvidia.com/object/tessellation.html>
Acesso em: novembro de 2013.

NVIDIA SDK Browser, *version* 11.0.328.2105. NVIDIA *Corporation,* 2011.

NVIDIA *Corporation*. *Anisotropic Filtering*. 2012a.
Disponível em: <http://www.geforce.com/whats-new/guides/aa-af-guide#1>
Acesso em: junho de 2014.

NVIDIA *Corporation*. *Enabling Ambient Occlusion in Games*. 2012b.
Disponível em: <http://www.geforce.com/whats-new/guides/ambient-occlusion#1>
Acesso em: junho de 2014.

OPENGL Khronos *Group*. *Shader*. 2013.
Disponível em: <http://www.opengl.org/wiki/Shader>
Acesso em: outubro de 2013.

PANKIEWICZ, Igor. O que é *V-Sync*?. 2009.
Disponível em: <http://www.tecmundo.com.br/jogos/1828-o-que-e-v-sync-vertical-synchronization-.htm>
Acesso em: junho de 2014.

PRADA, Rodrigo. O que é *Motion Blur*?. 2008.
Disponível em: <http://www.tecmundo.com.br/video-game/724-o-que-e-motion-blur-.htm>
Acesso em: junho de 2014.

RAEDER, Mateus. Um Estudo Sobre Técnicas de Avaliação de Desempenho e Modelos de Complexidade para Computação Paralela. 2007.
Disponível em: <http://www.inf.pucrs.br/gmap/pdfs/Mateus/TI-I%20Mateus%20Raeder.pdf >
Acesso em: maio de 2014.

REIS, Clausius Duque Gonçalves. Animação em tempo real de rugas faciais explorando as modernas GPUs. 2010.
Disponível em:
<http://www.dca.fee.unicamp.br/~martino/mestrados/Reis,ClausiusDuqueGoncalves_M.pdf>
Acesso em: outubro de 2013.

SALEN, Katie; ZIMMERMAN, Eric. Regras do jogo: fundamentos do design de jogos. Volume 1: Principais Conceitos. [tradução Edson Furmankiewicz]. São Paulo: Blucher, 2012. Título original: Rules of play: game design fundamentals.

SANTOS, Paulo Ivson Netto. Ray Tracing Dynamic Scenes on the GPU. 2009.
Disponível em: <http://www.tecgraf.puc-rio.br/~psantos/dissertacao/Dissertation.pdf>
Acesso em: fevereiro de 2014.

SHKLYAR, Dmitry. 3D *Rendering History*. 2004.
Disponível em:
<http://www.cgsociety.org/index.php/CGSFeatures/CGSFeatureSpecial/3d_rendering_history_part_1._humble_beginnings>
Acesso em: novembro de 2013.

SILVA, Tiago Sanches da; SCALCO, Roberto. Uso de *shaders* como ferramenta de auxílio ao processamento de algoritmos para iluminação e efeitos visuais. 2009.
Disponível em:
<www.academia.edu/1807975/Uso_de_shaders_como_ferramenta_de_auxilio_ao_processamento_de_algoritmos_para_iluminacao_e_efeitos_visuais>
Acesso em: setembro de 2013.

SOLIDLYSTATED. *Just Cause* 2 *Bokeh Filter*. 2010.
Disponível em: <http://solidlystated.com/software/just-cause-2-bokeh-filter/>
Acesso em: junho de 2014

SQUARE ENIX CO., LTD. *Just Cause* 2. 2010.
Disponível em: <http://www.justcause.com>
Acesso em: maio de 2014.

UNITY *Technologies*. *Screen Space Ambient Occlusion* (SSAO). 2013.
Disponível em: <http://docs.unity3d.com/Manual/script-SSAOEffect.html>
Acesso em: junho de 2014

VIANNA, Arlindo Cardarett. Computação Gráfica. 2002.
Disponível em:
<http://www.inf.ufes.br/~thomas/graphics/www/apostilas/CIV2801AcvCompGraf.pdf>
Acesso em: novembro de 2013.

ZAGO, Camila Avozani; RODRIGUEZ, Carlos Manuel Taboada; COELHO, Leandro Callegari; FOLLMANN, Neimar; SILVA, Vanina Macowski Durski. 2008.
Disponível em:
<http://www.aedb.br/seget/artigos08/516_516_benchmarking_logistico_seget.pdf>
Acesso em: maio de 2014.