# Análise e Projeto de Filtros Digitais para Correção do Efeito de *Aliasing* Aplicados na Computação Gráfica

Daniel Centeno Einloft
Faculdade de Engenharia de Computação
Pontifícia Universidade Católica do Rio Grande do Sul
Disciplina de Aplicação de Processamento Digital de Sinais
Porto Alegre, Rio Grande do Sul
*daniel.einloft@acad.pucrs.br*

Jean Larrocca
Faculdade de Engenharia de Computação
Pontifícia Universidade Católica do Rio Grande do Sul
Disciplina de Aplicação de Processamento Digital de Sinais
Porto Alegre, Rio Grande do Sul
*jean.larrocca@acad.pucrs.br*

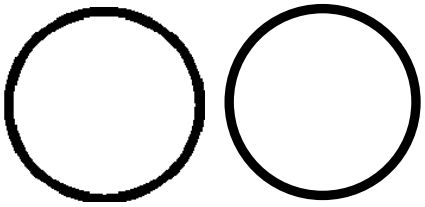
***Abstract*—Com os avanços da computação gráfica dos últimos 30 anos, a necessidade da geração de modelos gráficos de alta qualidade está cada vez mais maior. Porém, a modelagem e texturização de objetos 3D ainda possui grandes dificuldades em apresentar um objeto de forma fotorealista em tempo real. Isso se dá, não só pela capacidade de processamento atual, mas pela imprecisão de um computador ao representar um objeto real num ambiente virtual, devido à quantidade limitada de pontos de representação deste objeto, comparado com "infinitos" pontos presente na nossa realidade. Neste contexto, se utiliza uma série de filtros digitais para que um objeto gerado digitalmente seja mais semelhante possível a um objeto real. Esta artigo serve como apresentação do tema escolhido pelos alunos Daniel Centeno Einloft e Jean Larrocca para a disciplina de Aplicações de Processamento Digital de Sinais, sobre filtros de serrilhamento e filtros de textura devido à Aliasing na geração de gráficos.**

***Keywords*—Computação Gráfica, Filtros de Serrilhamento, Anti-Aliasing, Filtros de Texturas.**

## I. Introdução

A computação gráfica tem muita importância no mundo de hoje, tanto na indústria do entreterimento, em jogos digitais, quanto no campo da pesquisa, em aplicações CAD e simulações. Mesmo com a alta tecnologia dos motores gráficos da atualidade, a representação digital de um objeto real ainda permanece um desafio na área da computação gráfica. Isso porque computadores possuem um número finito de pontos para representar um objeto, em quanto na nossa realidade existe uma infinidade. Este problema foi demonstrado no teorema de amostragem de Nyquist-Shannon, onde é mostrado que um sinal analógico periódico pode sofrer alterações no seu comportamento se for amostrado incorretamente. De acordo com o teorema, para que não haja perda de informação, a frequência de amostragem precisa ser pelo menos duas vezes o valor da frequência máxima do sinal original [1].

Este problema é bem comum na computação gráfica. Se não existem *pixels* o sufiente para amostrar uma imagem de um objeto real, a imagem digital gerada fica cortada (*jagged*)[2]. Este efeito chama-se serrilhamento, ou *Aliasing*. Para solucionar este problema, existem duas alternativas: aumentar a frequência de amostragem (aumentando a resolução da imagem) ou criando algum filtro que minimize este efeito. A Figura 1 demonstra este problema, onde as bordas de um círculo estão com o problema de *Aliasing* (Figura 1(a)), e resolvendo-o com um filtro *Anti-Aliasing* (Figura 1(b)).



(a) Com *Aliasing* (b) Com *Anti-Aliasing*

Fig. 1: Circulos com e sem problema de serrilhamento.

O problema de amostragem possui duas consequencias: bordas de modelos em 3D com serrilhados e texturas cortadas. Para o primeiro caso pode se resolvido com filtros de *Anti-Aliasing*. Já o segundo, prefere-se aplicar algum tipo de filtro de textura, pois *Anti-Aliasing* normalmente causa efeitos de borramento (*blur*) em texturas [3], [4], [5]. Este artigo irá apresentar alguns filtros já desenvolvidos pela academia e desenvolvedores de jogos digitais (Seção II), apresentar a proposta de trabalho final da disciplina de Aplicações de Processamento Digital de Sinais (Seção III) e os resultados esperados (Seção IV).

## II. Trabalhos Relacionados

As técnicas de *Anti-Aliasing* usadas em jogos digitais e ambientes CAD podem ser divididas em duas categorias: pré-processamento (durante a renderização) e pós-processamento (depois da renderização). As primeiras tecnologias baseavam-se em pré-processamento, focando no teorema de Nyqyuist-Shannon para resolver o problema. A *Super Sampling Anti-Aliasing* (SSAA) e *Multi Sampling Anti-Aliasing* (MSAAA) aumentam a frequência de amostragem, ou seja, a resolução dos polígonos durante a renderização, para depois comprimir para a resolução correta. Esta compressão retira o efeito de serrilhamento com bastante eficácia, porém essas técnicas sofrem com a utilização de muita memória e possuem alta necessidade de uma elevada largura de banda [5], [6]. Além

destes modelos, a NVIDIA e AMD desenvolveram, respectivamente, o *Coverage Sampling Anti-Aliasing* (CSAA) e *Enhanced Quality Anti-Aliasing* (EQAA), baseadas no MSAA. Estas técnicas aumentaram a performance significativamente, mas ainda permanecem pesadas para aplicações de tempo real [3].

Por causa destas características, técnicas de pós-processamento começaram a ser cada vez mais implementadas tanto por desenvolvedores de motores gráficos quanto acadêmicos [3]. Desta forma, começou-se uma nova geração de filtros de *Anti-Aliasing*, mais rápidos e leves, baseados na análise de *pixels*, mais indicados para aplicações de tempo real. Em 2009, a NVIDIA lançou o *Fast Approximate Anti-Aliasing* conseguindo altas perfomances [7]. Já em 2011, foi apresentado o *Morphological Anti-Aliasing* (MLAA) [8], que em um ano, foi melhorado e substituído pelo *Enhanced Subpixel Morphological Anti-Aliasing* (SMAA) [4]. Além destas técnicas, foi desenvolvido uma grande variedade algoritmos de *Anti-Aliasing*, como *Adaptive Appoximate Anti-Aliasing* [5], *Accumulative Anti-Aliasing* (ACAA) [9], *Subpixel Reconstruction Anti-Aliasing* (SRAA) [10], entre outros.

Porém, a maioria destas técnicas não resolvem problemas de texturas, podendo até piorar a qualidade de texturas em superfícies [3], [4], [5]. Desta forma, para a filtragem de texturas, pode-se usar técnicas como *Nearest-Neighbor Interpolation*, *Linear Mipmap Filtering with Minimap* [11], *Bilinear Filtering* , *Trilinear Filtering* [12] e *Anisotropic Filtering* [13].

## III. Descrição do Trabalho

O trabalho escolhido pelos alunos consiste numa análise mais profunda sobre o efeito de *Aliasing* em gráficos de computador e nas diversas formas que este problema pode ser suavizado. Após um estudo detalhado das tecnologias de *Anti-Aliasing* e filtros de texturas, será escolhido alguns dos filtros estudados para implementa-los no *software* Matlab. Pretende-se focar nos algoritmos de pós-processamento, pois ambos os alunos não possuem experiência com renderização com OpenGL no Matlab. Desta forma, focar em detalhes de renderização demandaria muito tempo, sendo que este assunto não está relacionado diretamente com o problema.

Após a implementação de alguns filtros, pretende-se fazer uma análise de perfomance dos algoritmos desenvolvidos e testes em várias telas de jogos digitais, para assim analisar qual modelo proporciona uma qualidade de imagem melhor. Estes testes serão feitos com diversos jogos que tenham estilos gráficos distintos, podendo assim mostrar qual filtro se comporta melhor com estilos específicos. Como a qualidade da imagem pode ser um assunto muito subjetivo, será feita uma pesquisa com pessoas que estão acostumadas com este tipo de problema (ou seja, pessoas que jogam jogos de computador/consoles com mais frequência) e pessoas que não estão acostumadas (ou seja, que não jogam ou jogam de forma mais casual). Maiores detalhes da pesquisa ainda está em discussão entre a dupla.

## IV. Resultados Esperados

Os resultados esperados pela dupla consiste em entender as principais soluções de filtros de *Anti-Aliasing* e de texturas, explicando sucintamente as semelhanças/diferenças e lados positivos/negativos de cada tecnologia. Grande parte do foco será direcionado para este assunto, pois a dupla acredita que não existe muita informação direta e sucinta nos manuais de jogos digitais. Da breve pesquisa feita em alguns jogos de computadores (pertencentes à grandes franquias de videogame, como Final Fantasy, Half-Life, Borderlands, etc), nenhum deles explicava do que se tratava os filtros utilizados.

Com a finalização do desenvolvimento das técnicas escolhidas, pretende-se investigar até onde o olho humano identifica as falhas apresentadas e qual o limite entre a penalidade de perfomance e qualidade gráfica dos jogos.

## Agradecimentos

Os autores gostariam de agradecer ao professor Doutor Denis Fernandes, por proporcionar a possibilidade de estudo na área de Computação Gráfica, e à professora Doutora Isabel Harb Manssour, pelo auxílio e indicação de materiais para a pesquisa do tema escolhido.

## References

[1] B. P. Lathi, *Sinais e Sistemas Lineares*, 2nd ed. Bookman, 2007.

[2] B. M. P. Hearn, Donald, *Computer Graphics: C Version*, 2nd ed. Prentice Hall, Inc., 1997.

[3] S. W.-y. L. W. L. L.-z. M. Xu-dong, Jiang Bin, "Image anti-aliasing techniques for internet visual media processing: a review," *Journal of Zhejiang University SCIENCE C*, 2014.

[4] J. Jimenez, J. I. Echevarria, T. Sousa, and D. Gutierrez, "Smaa: Enhanced morphological antialiasing," *Computer Graphics Forum (Proc. EUROGRAPHICS 2012)*, vol. 31, no. 2, 2012.

[5] Y. J. J. P. C. S. Jae-Ho Nah, Sunho Ki, "Axaa: Adaptive approximate anti-aliasing," *SIGGRAPH '16 ACM SIGGRAPH 2016*, 2016.

[6] *Design and Analysis of Anti-Aliasing Filters for Rendering of Graphical Entities*. ICTCS '16 Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies, 2016.

[7] T. Lottes, "Fxaa," NVIDIA, Tech. Rep., 2009, white Paper.

[8] J. Jimenez, B. Masia, J. I. Echevarria, F. Navarro, and D. Gutierrez, *GPU Pro 2*. AK Peters Ltd., 2011, ch. Practical Morphological Anti-Aliasing.

[9] C. R. O. K. Eric Enderton, Eric Lum, "Accumulative anti-aliasing," NVIDIA, Tech. Rep., 2015.

[10] D. L. Matthaus G. Chajdas, Morgan Mcguire, "Subpixel reconstruction antialiasing for deferred shading," in *I3D '11 Symposium on Interactive 3D Graphics and Games*, 2011, pp. 15–22.

[11] *The Official Guide to Learning OpenGL, Version 1.1*, http://www.glprogramming.com/red/chapter09.htmlname3.

[12] K. I. F. Joel McCormack, Ronald Perry and N. P. Jouppi, "Feline: Fast elliptical lines for anisotropic texture mapping," in *SIGGRAPH '99 Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, 1999, pp. 243–250.

[13] J.-A. L. Hyun-Chul Shin and L.-S. Kim, "Spaf: sub-texel precision anisotropic filtering," in *HWWS '01 Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pp. 99–108.