CENTRO UNIVERSITARIO
DE TECNOLOGÍA Y ARTE DIGITAL

Estudio sobre sistemas de anti-aliasing e implmenetación de anti-aliasing temporal

Hugo Ferrando Seage

U-Tad

septiembre 2018

Tutor: Alberto Sánchez Campos

Resumen

Debido a la resolución limitada de las pantallas al rasterizar gráficos 3D se deforman ciertas lineas y curvas. Este fenómeno se llama aliasing. La demanda de gráficos cada vez más realistas ha propiciado la creación de diferentes técnicas y algoritmos para disimular estos defectos, sin tener que recurrir necesariemente a pantallas de mayor resolución.

El antialiasing se ha vuelto una técnica crucial para mejorar la calidad de imagen en el software con gráficos tridimensionales. Este campo de la computación gráfica lleva años desarrollandose, con nuevas técnicas publicadas continuamente.

Este proyecto tiene como objetivo el análisis de las distintas técnicas existentes en el estado del arte y la implementación de un algoritmo de temporal antialiasing, compatible con motores con deferred shading y comparable en calidad al multisampling, junto a otras propiedades.

Abstract

When rasterizing 3D graphics using computer screens, due to their limited resolution some lines and curves will become deformed. This phenomena is called aliasing. The ever increasing demand for more realistic graphics has driven the creation of new techniques and algorithms to hide these artifacts, without necessarily using higher resolution screens.

Antialiasing has become a crucial technique to advance image quality in graphics software. This field has been explored for years, with new approaches being developed continually.

This projects aims to analyze different antialiasing techniques developed and the implementation of a temporal antialiasing technique, compatible with deferred shading engines and with similar quality to mutlisampling, amongst other benefits.

Índice general

Glosario

OpenGL	API estándar para el desarollo de software con gráficos 2D o 3D con aceleración por hardware[10].
rasterizar	Conversión de una imagen descrita con alguna primitiva (curvas, vectores, triangulos) a un conjunto de pixels.

Capítulo 1

Introducción

1.1. ¿Qué es el aliasing?

El antialiasing es la deformación de ciertos elementos gráficos al ser rasterizados y plasmados en una pantalla con una resolución finita[1]. Normalmente los bordes de la geometría que tengan ángulos que no se alineen perfectamente con los pixels presentaran bordes de sierra abruptos, que no es fiel a la escena como se vería de forma natural.

Este proyecto se centra en aliasing dentro del campo de la computación gráfica, pero afecta a otras disciplinas, como el procesamiento de señales (por ejemplo al digitalizar señales de audio analógicas). En general, al representar valores continuos en algún medio discreto siempre habrá algún defecto de este tipo[1].

Existen otros tipos de aliasing, como en el interior de texturas o al mover la camera rápidamente.

1.2. ¿Qué es el antialiasing?

El antialiasing es el conjunto de técnicas cuyo objetivo es el de disimular o eliminar estas imperfecciones.

Capítulo 2

Planteamiento del problema

2.1. Aliasing en bordes de geometría

imagen

2.2. Aliasing de texturas

Posibles soluciones: Mip Maps, LoD

2.3. Specular Aliasing

2.4. Motion Aliasing

Explicar motion blur en cámaras

Accumulation motion blur (ps2+) Per pixel motion blur (ps3+) Per object motion blur (ps3+)

Capítulo 3

Objetivos

Este proyecto tiene dos objetivos principales:

1. Analizar el estado del arte en el campo de la computación gráfica en relación al anti-aliasing. Explorar los problemas que presenta el aliasing en gráficos 2D y 3D, la evolución de las soluciones y los desarrollos en curso.

2. El segundo objetivo es el de integrar una solución de anti-aliasing temporal a un motor gráfico 3D existente desarrollado durante la asignatura de ÄPIs tridimensionalesd̈el Máster en Computación Gráfica y Simulación cursado.

Capítulo 4

Estado del Arte

4.1. Filtrado

En gráficos raster (bidimensionales) el anti-aliasing se realiza mediante filtros.Para entender el filtrado en imágenes es necesario pensar en pixels no como pequeños cuadrados en una pantalla, si no en muestras de una función (normalmente 3 muestras, una por cada canal RGB)[2]. Los filtros (o kernels) transforman las funciones para intentar suavizar los cambios bruscos de color.

En la figura 4.1 se muestra una comparación de las diferentes formas en las que cada filtro interpola valores de una imagen con un solo canal. Cada punto representa la muestra de la función que se visualizará en la pantalla.

Figura 4.1: Comparación de interpolación usando diferentes filtros[8]

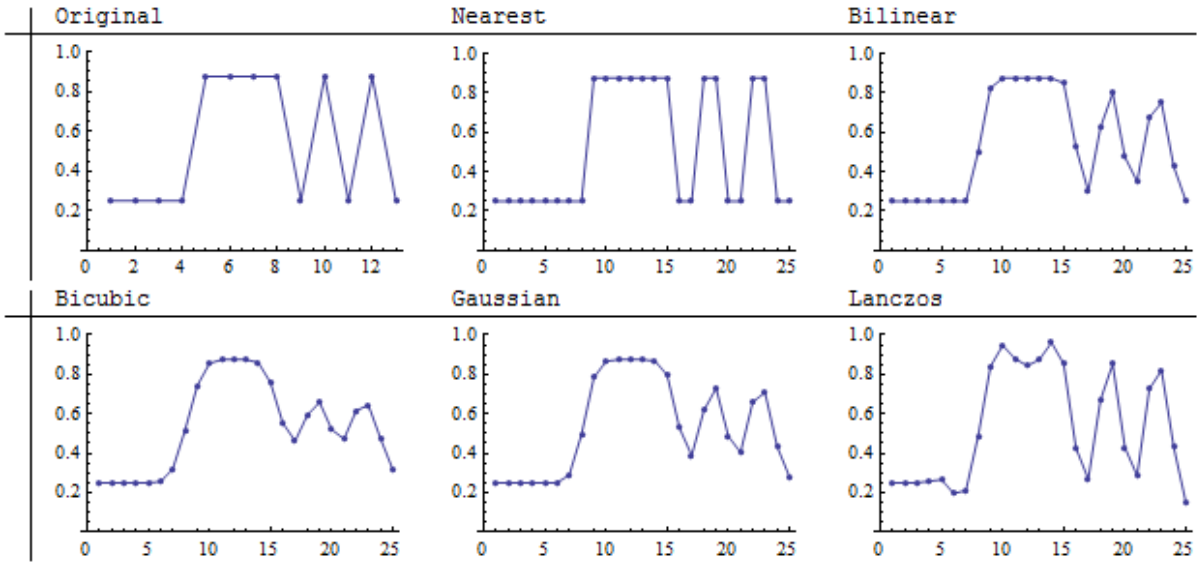
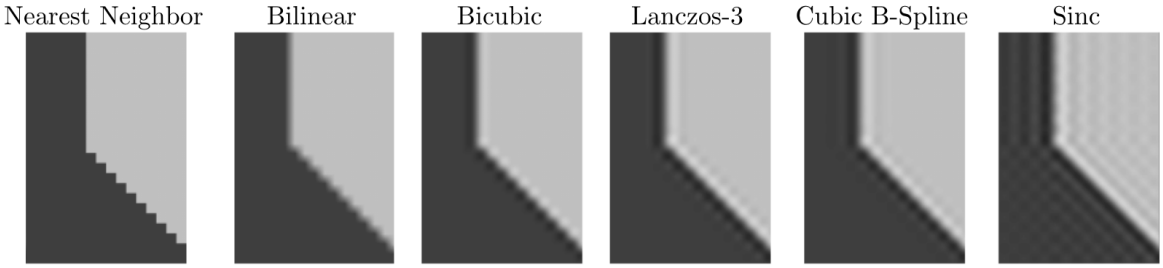


Figura 4.2: Comparación de resultados de diferentes filtros[7]



4.2. Supersampling Antialiasing

El supersampling consiste en minimizar el aliasing usando imágenes con una resolución mayor a la usada para su visualización (antialiasing espacial). Cada pixel en la pantalla será representado por más de un pixel en la imagen. Para el color final normalmente se hace la media del valor de los colores de los múltiples pixels.

Estas técnicas consumen más ancho de banda y memoria, ya que los buffers tienen que guardar más información por pixel que la imagen original.

La calidad del supersampling viene dado por la cantidad de samples por pixel y la técnica para saber que pixels usar en la imagen con mayor resolución para cada pixel de la pantalla.

En comparación a otras técnicas de antialiasing esta corrige aliasing de texturas y especular, además de aliasing de geometría. Esto se debe a que los subsamples los coge de en toda la escena.

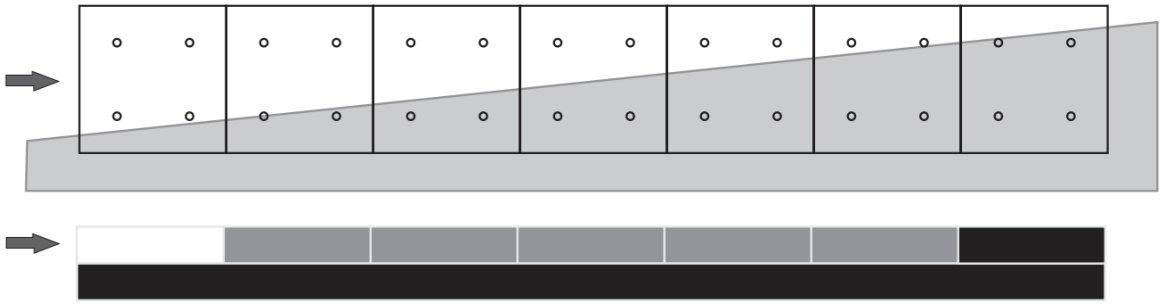
4.2.1. Ordered Grid Supersampling

En esta técnica de supersampling la distribución de los samples es uniforme. Debido a la naturaleza regular de este patrón, los subpixels se encuentran situados en forma de dos columnas y dos filas. Para lineas muy planas, tanto en horizontal como vertical el antialiasing no será muy efectivo, ya que muy rápidamente tocará dos subpixels en fila o columna y hasta el final no tocará los otros subpixels4.3.

Este efecto se puede reducir usando más samples, pero el impacto en el rendimiento será todavía mayor.

Desde 2014 Nvidia ha publicado una técnica llamada Dynamic Super Resolution[9], que consiste en renderizar software a mayor resolución y usar OGSS con un filtro de blur gaussiano. En 2015 AMD implementó una técnica similar llamada Virtual Super Resolution.

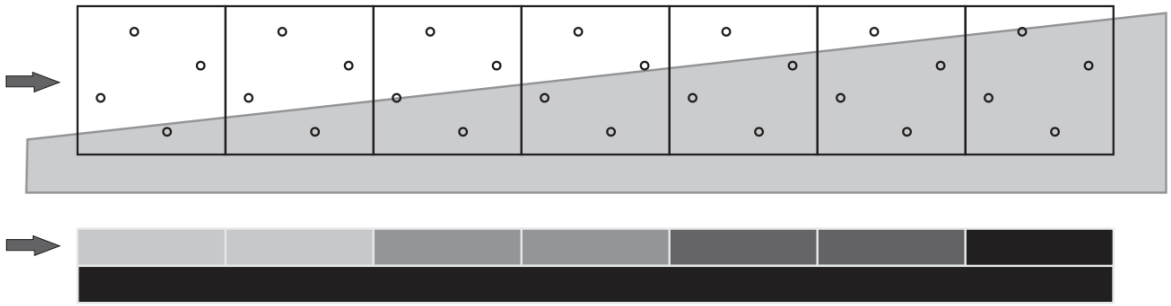
Figura 4.3: Ordered Grid Super-Sampling[3]



4.2.2. Rotated Grid Supersampling

Aquí la idea es cambiar el lugar de los subsamples para minimizar la cantidad de columnas y filas. La distribución sigue siendo uniforme, pero el problema del aliasing en lineas muy planas se soluciona4.4 sin tener que recurrir a aumentar de manera significativa el numero de subsamples.

Figura 4.4: Rotated Grid Super-Sampling[3]

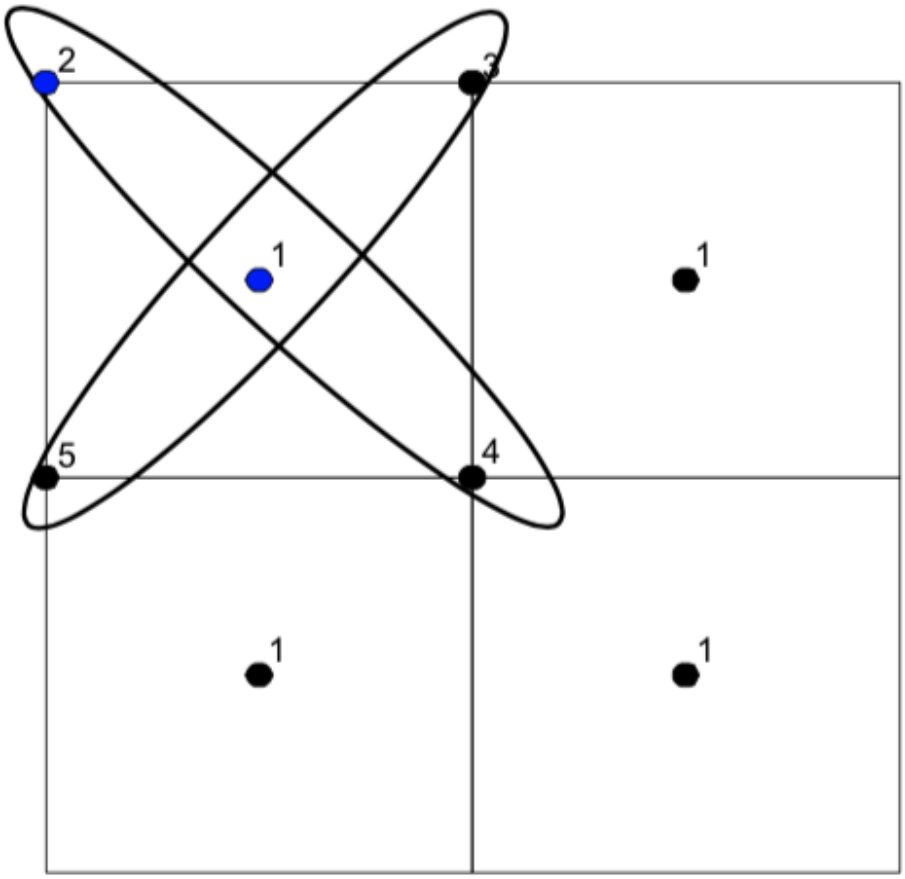


4.2.3. Quincunx Supersampling

Este patrón es interesante como técnica de supersampling ya que usa subsamples localizados en las esquinas de los pixels, lo que permite reutilizar varios subsamples entre pixels4.5. Esto permite una calidad similar a la de un antialiasing 4x pero con un coste computacional de un supersampling 2x. Aún así al reutilizar subsamples produce resultados algo más borrosos que otros métodos.

Aunque este patrón se pueda usar para supersampling Nvidia lo implementó en hardware desde la GeForce 3[4] para acelerar el calculo de su multisampling. Sony al utilizar una GPU de Nvidia en la PS3 también disponía de esta tecnología en la consola.

Figura 4.5: Quincunx Pattern[4]



4.3. Multisampling

Multisampling es una optimización del supersampling. Se sigue usando subpixels, pero en vez de resolver el color para cada subpixel el color se resuelve una única vez por pixel. Después de resolver el color se calcula el stencil buffer y depth buffer con subpixels, y dependiendo de la cantidad de subpixels que estén contenidos dentro del triangulo se usará un porcentaje del valor del color calculado. Esto reduce drásticamente el impacto en el rendimiento de la aplicación, ya que solo hay que acceder una vez por pixel a la textura,

pero solo soluciona aliasing en los bordes de la geometría.

En el centro de un polígono todos los subpixels van a estar contenidos dentro, pero el color solo se calcula una vez (por ejemplo en el centro del pixel), por lo que se usará el 100 % del color, pero puede contener aliasing. En cambio, en los bordes, algún subpixel estará fuera de la geometría, por lo que se usará un porcentaje del color calculado, suavizando el borde.

Otra de las desventajas del multisampling anti-aliasing es que, tradicionalmente, no se puede usar en motores con deferred shading. Debido a que el calculo del color final de cada pixel se realiza en un pixel shader posterior al calculo del g-buffer, al hacer el resolve del mutisampling no esta lista toda la información necesaria. Desde OpenGL 3.2 y Direct3D 10.1 esto ha cambiado gracias al explicit multisampling[11]. La creación del g-buffer ahora se puede realizar usando texturas mltisampled y al calcular la luz se puede acceder a los subsamples mediante la función texelfetch. Si se realiza de esta forma hay que sacrificar la creación automática de mipmaps y filtrado de texturas, aunque siempre se pueden implementar manualmente en sahders. Aún así muchas desarrolladores han optado por usar post-processing antialiasing al usar deferred shading.

4.4. Post Processing Antialiasing

Este tipo de antialiasing ha ganado popularidad durante los últimos años, debido en parte al uso del deferred shading en los últimos motores gráficos. Los algoritmos presentados a continuación se pueden considerar filtros de rasters, como los de la sección4.1, pero están pensados para ser usados en motores gráficos. Además, en muchos casos, usan información que no esta presente en la textura final, como pueden ser texturas de profundidad o vectores de movimiento,

4.4.1. Fast Approximate Anti-aliasing

El FXAA[6] es un sistema de anti-aliasing de post procesado desarrollado por NVIDIA.

4.4.2. Morphological Antialiasing

mlaa

4.4.2.1. Enhanced Subpixel Morphological Antialiasing

smaa

4.4.3. Temporal Antialiasing

El temporal antialiasing es un tipo de antialiasing de post procesado, pero que amortiza el coste computacional usando múltiples samples en múltiples frames. Esto presenta unas complicaciones a la hora de escoger samples en frames anteriores en imágenes no estáticas.

Capítulo 5

Desarrollo

Para este proyecto se ha integrado un sistema de anti-aliasing temporal en un motor de gráficos 3D que usa OpenGL como API gráfica.

5.1. Jitter

5.2. Motion Vectors

5.3. Combinación de texturas

5.4. Neighborhood Clamping

5.5. Sharpening kernel

Capítulo 6

Resultados

Figura 6.1: Anti-aliasing off

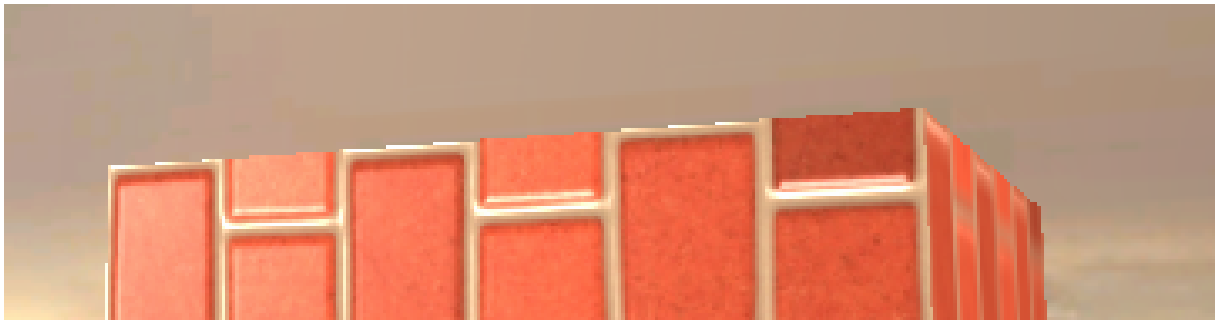


Figura 6.2: Temporal anti-aliasing con 8 samples

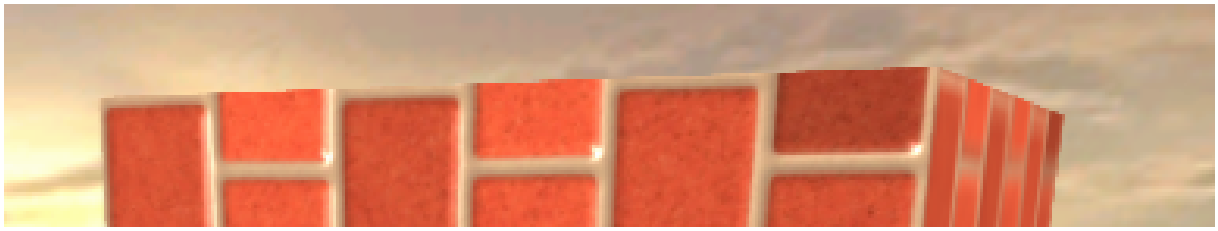


Figura 6.3: Temporal anti-aliasing con 16 samples

Capítulo 7

Conclusión

Bibliografía

[1] D. P. Mitchell y A. N. Netravali, «Reconstruction Filters in Computer-graphics», *SIGGRAPH Comput. Graph.*, vol. 22, n.º 4, págs. 221-228, jun. de 1988, ISSN: 0097-8930. DOI: 10.1145/378456.378514. dirección: http://doi.acm.org/10.1145/378456.378514.

[2] A. R. Smith, «A Pixel Is Not A Little Square, A Pixel Is Not A Little Square, A Pixel Is Not A Little Square! (And a Voxel is Not a Little Cube)», Technical Memo 6, Microsoft Research, inf. téc., 1995.

[3] K. Beets y D. Barron, «Super-sampling Anti-aliasing Analyzed», 2000.

[4] Nvidia, «HRAA: High-Resolution Antialiasing through Multisampling», inf. téc., 2001. dirección: https://www.nvidia.com/object/feature_hraa.html.

[5] J. Sachs, «Image Resampling», inf. téc., 2001.

[6] T. Lottes, «FXAA. NVIDIA White Paper», inf. téc., 2009.

[7] P. Getreuer, «Linear Methods for Image Interpolation», vol. 1, sep. de 2011.

[8] *What is Lanczos resampling useful for in a spatial context?*, 2011. dirección: https://gis.stackexchange.com/questions/10931/what-is-lanczos-resampling-useful-for-in-a-spatial-context.

[9] Nvidia, «GeForce GTX 980 Whitepaper», inf. téc., 2014. dirección: https://international.download.nvidia.com/geforce-com/international/pdfs/GeForce_GTX_980_Whitepaper_FINAL.PDF.

[10] J. Kessenich, G. Sellers y D. Shreiner, *OpenGL®Programming Guide: The Official Guide to Learning OpenGL®, Version 4.5 with SPIR-V*, 9.ª ed. Addison-Wesley Professional, 2016, ISBN: 9780134495491.

[11] K. Nvidia, «OpenGL ARB texture multisample», inf. téc., 2009-2014. dirección: https://www.khronos.org/registry/OpenGL/extensions/ARB/ARB_texture_multisample.txt.

Índice de figuras

Índice de cuadros