Anti-aliasing algorithm Midterm Report

for CS-534 Computational Photography

By Xiaochao Yan

University of Wisconsin Madison

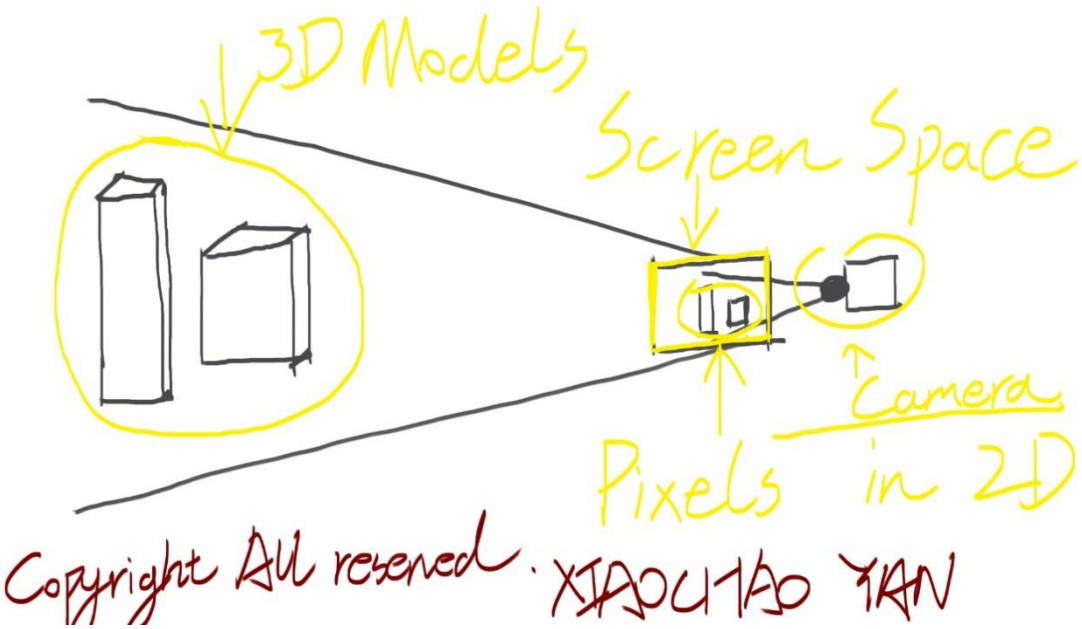
Introduction

In video game industry, there is also requirements to produce appealing graphics at real-life speed. Since the goal of modifying image is different, methods used to improve graphics quality are slightly different from what we have learned in lectures of CS 534 Computational Photographic taught by professor Mohit Gupta. This semester long research will be mainly about how video game industry improve graphics quality and how they managed to keep producing graphics at real life speed.

Why images are not perfect and have jagged lines?

While real-life objects are built of beautiful curves and smooth but varying surfaces, digital 3D models using vertices and lines connecting them to show 3D objects. (There is 3D vector model which has real curves but is beyond scope of this research). To save storage space and computational power, surface in 3D model will be stored as a group of flat surfaces which only represented by 3 or 4 vertices. To generate image from those 3D model, we use a process known as ***Rendering*** to project 3D objects into 2D screen space. There is a key process called ***Rasterization*** that project all models into screen space.

Fig1: Rasterization



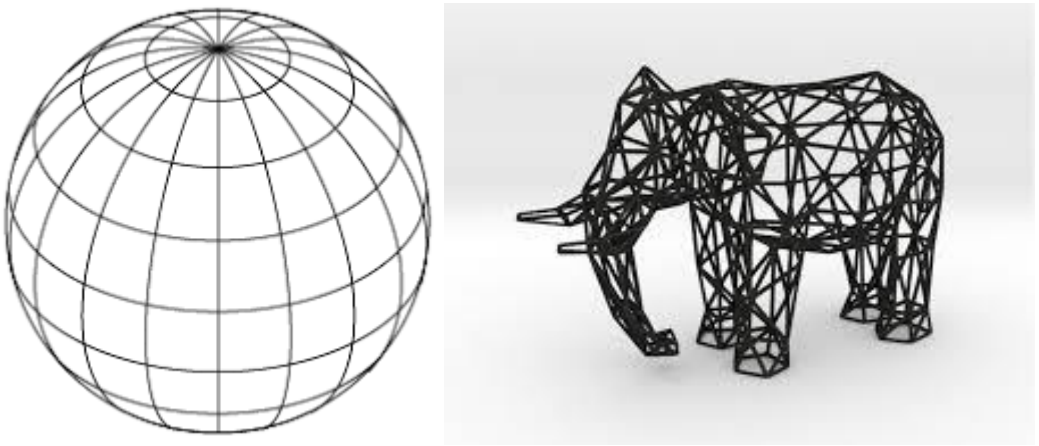
In rasterization, there are usually 4 kinds of coordinate system.

1. World coordinate, position comparing to origin (x=0,y=0,z=0)
2. Camera coordinate, position comparing to camera position, camera position is (0,0,0)
3. Film coordinate, position of pixels on screen space in the form of (x,y), could be equal to Pixel coordinate.
4. Pixel coordinate, the final position of pixel on screen in the form of (x,y).

What rasterization do is multiple the coordinate of all 3D objects by a sequence of pre-calculated matrix to convert all 3D objects in to pixels position. This process is usually done in the 4 stages, corresponding to the 4 types described above. The final product, pixel coordinate is information needed to display an image on screen.

When one looked closely on images produced using these inaccurate 3D models, he or she will notice that what appears to be curve are a sequence of lines connected. Intersect of all lines seem to be sharp (discontinued) instead of the smooth curve we see in real life. Though those lines are clearly defined, we as human will perceive those sharp turning as imperfect representation of curves and the overall graphics unappealing.

Fig 1 : wire frames of 3D models, showing how 3D objects represented in digital world



What exactly is Anti-aliasing/Super sampling?

To minimize the unpleasant presence of jagged pixels, techniques called anti-aliasing is employed. Anti-aliasing is NOT any single technique or algorithm used to improve the graphics quality; it is the overall name of all sort of techniques that are used to smooth jagged lines. These techniques enable more accurate approximation of curves in screen space.

Fig2: Effect of anti-aliasing, the jagged edge became smother.

2 main types of Anti-aliasing method.

1. Super Sampling

Calculate the output graphics at higher resolution than actual output resolution. For each pixel on screen space, the output color is a combination of This method has a computational cost between Spatial anti-aliasing and Temporal Sample anti-aliasing. Examples are Multiple Sampling Anti-Aliasing (MSAA), full-scene anti-aliasing (FSAA).

2. Fast Approximate Anti-Aliasing (FXAA) like anti-aliasing

Only use information (pixel position, color, etc.) in screen space to smooth out jagged pixels. Gaussian filter could be used for this type of anti-aliasing. This type of anti-aliasing has the lowest computational cost but could blur the image and worsen the final output graphics quality.

Some anti-aliasing methods cannot be fitted into above categories:

MFAA Multi-Frame Anti-Aliasing

Since video games are not static objects but moving objects, so the output is a sequence of similar images, which seen by us as video. Because for most of the time, any object will NOT move by a huge difference in screen space in video games, between frames the output images are like each to each

other. Due to the similarity of output images, it is possible to use output from previous frames to sample the output of current frame. This algorithm uses similiar logic as that of ***Super-Sampling***.

MLAA Morphological Antialiasing

This algorithm makes use of only information from screen space, which resembling many other ***Temporal Sample Anti-aliasing*** method. However, what interesting is that this algorithm makes use of feature detection. Techniques like feature detection are usually used in computer vision and not seen in most of anti-aliasing algorithm.

The algorithm consists of the following steps:

1.Noticeably different pixels are identified. Figure 3 shows a sample image with solid axis-aligned lines separating different pixels.

2. Piecewise-linear silhouette lines are derived from the separation lines. This is illustrated in Figure 3 with a Z-shape formed by b-c-d lines and a U-shape defined by d-e-f lines. In MLAA, silhouette segments originate at the edges of pixels, that have both horizontal and vertical separation lines (all such pixels are shown with stripped shading). Not all potential end-points are used. The exact placement of end-points is somewhat arbitrary, with half-edge points producing satisfactory results.

3.Color filtering is performed for all pixels intersected by silhouette lines. Essentially, this is done by propagating colors on opposite sides of separation lines into polygons formed by silhouettes and separation lines, as illustrated at the bottom of Figure 3. The areas of these polygons are used for color mixing.

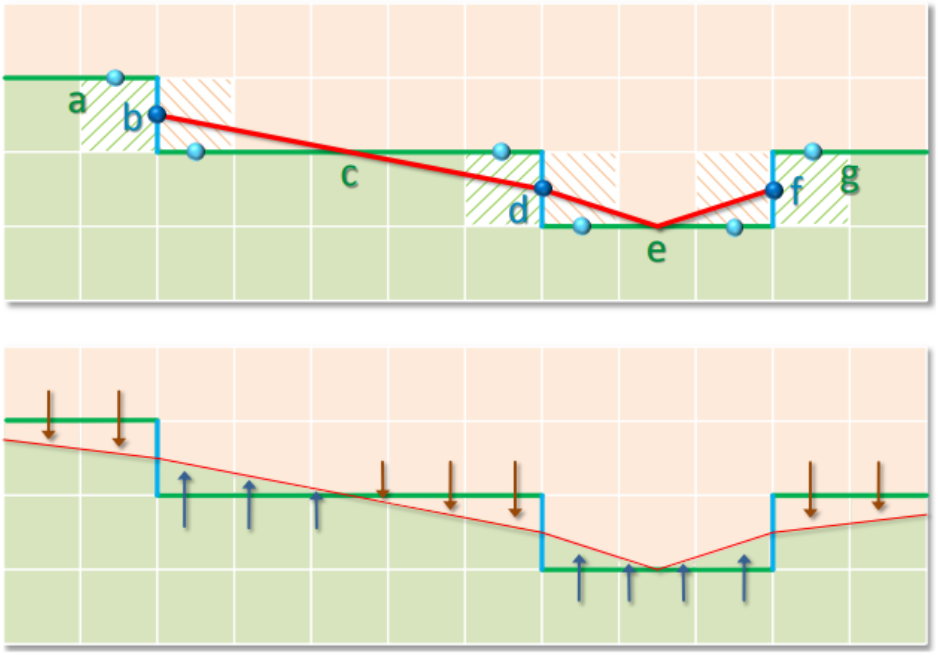


Figure 3: *A sample picture illustrates the main MLAA concepts. Solid axis-aligned lines separate different pixels. Top: reconstructed silhouettes are shown as red lines. Bottom: filtering through color propagation.*

Reference:

Jason Gregory, Jeff Lander (2009). Game Engine Architecture

Alexander Reshetov, (2009). Morphological Antialiasing

Jorge Jimenez, Diego Gutierrez, Jason Yang, Alexander Reshetov, Pete Demoreuille, Tobias Berghoff, Cedric Perthuis, Henry Yu, Morgan McGuire, Timothy Lottes, Hugh Malan, Emil Persson, Dmitry Andreev, Tiago Sousa (2011). Filtering Approaches for Real-Time Anti-Aliasing, http://www.iryoku.com/aacourse/

NVIDIA, (2014). GeForce Tech Demo: MFAA, https://www.youtube.com/watch?v=Nef6yWYu0-I

Timothy Lottes,(2009). FXAA, http://developer.download.nvidia.com/assets/gamedev/files/sdk/11/FXAA_WhitePaper.pdf

What to do next?

Since MATLAB provides simple 3D rendering and performance testing mechanic. I will try to implement some algorithms, such as super sampling, FXAA to test the exact difference in computational cost.

I was not able to access as many of anti-aliasing algorithm are developed by entrepreneurs and papers about those algorithms are kept confidential. However, there are released algorithm such as super sampling and TSAA. I will try to use MATLAB to implement 2-3 anti-aliasing algorithm and do performance test on those algorithms.

The test will be mainly on following parameters.

1. Avg time used on processing one frame of image
2. The average and variance of processing time when processing a long sequence of images
3. Memory used.