

## 실시간 렌더링을 위한 효율적인 Anti-Aliasing

한태근\*, 안도랑, 이동욱  
동국대학교 전기공학과

### An Efficient Anti-Aliasing Algorithm for Real-Time Rendering

Tae-Kuen Han\*, Do-Rang Ahn, Dong-Wook Lee.  
Dept. of Electrical Engineering, Dongguk University

**Abstract** - In the field of computer graphics, it approached the investigation of outstanding performance and high speed. Although most introduced Anti-Aliasing method were to meet these, it was not to improve speed. Because Anti-Aliasing method was focus on only quality. Anti-Aliasing Effect is compensated from movement of object on the screen. Speed is important in the REAL-TIME application program like as 3D games. Cause Anti-Aliasing which needs great amount of time is not used in general.

This Paper suggest the Efficient Anti-Aliasing method which apply Two-Point Anti-Aliasing Method that informs brightness data of the screen and use adjacent brightness data for real-time rendering.

### 1. 서 론

임의의 물체를 렌더링하는 것은 연속적인 2차원 신호를 디스플레이 디바이스의 해상도로 샘플링 하는 과정이다. Sampling Theory[1]에 의하면 Nyquist Rate보다 더 많은 Sampling Rate이 보장될 때만이 연속적인 입력 신호가 이산적인 신호로 정확하게 재구성 될 수 있다. 하지만, 한정된 Display Device의 해상도로는 이러한 조건을 만족시킬 수 없기 때문에 필연적으로 정보의 손실에 의한 부정적인 효과가 나타나게 된다. 이러한 Aliasing 현상의 대표적인 예로 <그림 1>에서 볼 수 있는 것과 같은 Jagged Edge와 Texture에서 두드러지게 나타나는 Moire Pattern을 들을 수 있다.



<그림 1> 2차원 이미지에서의 Jagged Edge

Anti-Aliasing이란 이러한 부정적인 효과를 해소하기 위한 방법으로 크게 두 가지가 있다. 첫 번째 방법은 Sampling Rate을 늘리는 방법이고, 두 번째 방법은 필터링을 이용하여 Nyquist Rate 보다 높은 주파수를 갖는 부분을 입력신호에서 제거하는 방법이다.

본 논문에서는 비교적 새로운 개념인 Two-Point Anti-Aliasing 기법[2]을 이용하여 실시간의 렌더링이 요구되는 분야에서 유용하게 사용될 수 있는 효율적인 Anti-Aliasing 방법을 제시하였다.

### 2. 본 론

#### 2.1 선그리기 알고리즘

##### 2.1.1 브렌센핸(Bresenham) 알고리즘

단지 정수 계산만을 이용하여 실제 직선에 가까운 좌표를 찾아 그리는 알고리즘으로 더 정확하고 효율적인 래스터 선-생성 알고리즘이다.

1보다 적은 양의 기울기( $|m| < 1$ )를 갖는 직선에 대한 주사변환 처리를 고려해 보자. 먼저, 두 개의 선분 끝점을 입력, 왼쪽 끝점을  $(x_0, y_0)$ 로 저장한다. 그리고 프레임 버퍼에  $(x_0, y_0)$ 을 적재하여 첫 번째 점을 그린다. 다음과 같이 판단 매개 변수 값을 얻기 위해 상수  $\Delta x, \Delta y, 2\Delta y, 2\Delta y - 2\Delta x$ 를 계산한다.

$$p_0 = 2\Delta y - \Delta x \quad (1)$$

$k=0$ 에서 시작하여 직선을 따라 각  $x_k$ 에서 다음과 같은 검사를 수행한다.

만일  $p_k < 0$  이면 다음 점 좌표는  $(x_{k+1}, y_k)$ 이고

$$p_{k+1} = p_k + 2\Delta y \quad (2)$$

$p_k > 0$  이면, 다음 점의 좌표는  $(x_{k+1}, y_{k+1})$ 이다.

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x \quad (3)$$

위의 단계를  $\Delta x$  번 만큼 반복한다. 주사 변환될 각 직선에 대해 상수  $2\Delta y$ 와  $2\Delta y - 2\Delta x$ 은 한번만 계산되므로, 계산 시 오직 정수의 덧셈과 뺄셈만 쓰인다.

##### 2.1.2 Two-Point Anti-Aliasing 알고리즘

Two-Point Anti-Aliasing 기법이란 인접한 두 화소의 밝기정보를 이용하여 한 화소의 밝기 정보를 나타내는 방법이다. 즉 <그림 2>의 (b)와 같이 인접한 두 화소의 밝기 정보가 합해져서 한 개의 화소처럼 보이게 함으로써, 한 화소를 여러 개의 부분화소로 나누어 그 화소를 지나는 물체의 영역에 따라 그 화소의 밝기 정보를 구하는 방법과 같은 효과를 낼 수 있다.



(a)Aliased      (b)Anti-Aliased  
<그림 2> Aliased And Anti-Aliased Lines

$I[i, j]$ 를 화소  $(i, j)$ 의 밝기 정보라 하고  $I_0$ 를 그리려는 곡선  $f(x)$ 의 밝기 정보라고 할 때, 가상의 화소  $(i, f(i))$ 의 밝기 정보는 식(4)과 같이 두 화소를 이용하여 표현될 수 있다.

$$I[i, f(i)] = I_0(f(i) - f(i)) \quad (4)$$

$$I[i, f(i)] = I_0 - f(i)$$

또한, 한 화소  $(i, j)$ 를 중심이  $(i, j)$ 에 있고, 밝기 정보  $I[i, j]$ 를 갖는 크기 1의 정사각형이라고 하면, 식(5)와 같이 점  $P=(i, f(i))$ 는 두 점  $P0=(i, f(i))$ 의 중심이 된다.

$$IoP = I[i, f(i)]p0 + I[i, f(i)]p1 \quad (5)$$

그러므로 식(4)의 효과는 그리려는 곡선  $f(x)$ 상의 점  $P=(i, f(i))$ 에 초점을 갖는 빛의 영역이 되어 결과적으로 사람의 눈에는 정확히 곡선  $f(x)$ 위의 점으로 인식이 되게 된다. 따라서, 위의 샘플링에 의해 생기는 오차는 0이 되어 정보의 손실을 없앨 수 있게 된다.

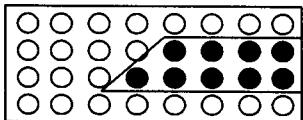
식(4)가 바로 Two-Point Anti-Aliasing 기법의 핵심이다. 이 방법의 가장 큰 잊점은 그 단순성으로 어떤 응용 프로그램에도 적용이 가능하다. 참고문헌[2]에서는 이 방법을 이용하여 Bresenham방법[3] 보다 빠른 직선과 원의 렌더링 방법을 제시하고 있다.

### 2.1.3 A-Buffer 알고리즘

A-Buffer Algorithm은 1974년에 Carpenter가 제안한 방법으로 한 화소를 32개의 영역으로 나누어 물체가 차지하는 부분을  $4\times 8$ 마스크를 이용하여 계산하고 Box Filtering을 이용하여 그 화소의 값을 결정한다.

이 방법은 우선 다각형들을 Scan Convert하고 그 결과 화소의 경계면으로 Clipping 되는 화소들의 리스트를 구한 후, 각 화소들에서  $4\times 8$ 마스크를 이용하여 다각형이 그 화소에서 차지하는 영역을 구한다. 이 때, 다각형의 변에 해당하는 부분과 그 오른쪽에 있는 마스크의 bit값은 1을 가지며, 겹쳐지는 마스크들은 서로 xorring 되어 그 화소에 해당하는 마스크의 값이 결정된다. 그 후, 결정된 마스크의 값을 이용하여 화소의 색을 구하게 된다.

<그림 3>에서 알고리즘에 사용될 마스크를 제시하였다.



<그림 3> Polygon Edge Mask

이 방법은 Lucas Film의 REYES 3-D 렌더링 시스템에 적용되었으며, Star Trek II에서 성공적으로 사용되었다고 한다. 하지만, 위에서 기술한 바와 같이 한 화소의 값을 결정하기 위해서는 복잡한 화소의 자료 구조와 함께 마스크의 계산과 Filtering이 필요하기 때문에 실시간 응용 프로그램에 사용하기에는 무리가 따른다고 할 것이다.

## 2.2 Two-Point Anti-Aliasing 알고리즘의 적용

본 시스템에서의 기본적인 렌더링 단위는 삼각형이다. 따라서 곡선이나 곡면에 대한 고려는 필요없다. 하지만, 그려지려는 물체의 색깔과 배경의 색깔은 항상 변하므로, 그에 따른 새로운 식의 고안이 필요하며, 또한 투명한 물체나 서로 교차하는 물체의 정확한 렌더링이 보장되어야 한다.

### 2.2.1 개선된 Two-Point Anti-Aliasing 알고리즘

본 알고리듬은 Aliasing 효과를 없애기 위해서는 샘플링이 일어나기 전에 High Frequency를 갖는 부분이 제거되어야 하며, 이 부분은 렌더링 되려는 물체의 경계

면이라는 주안점에서 시작된다. 즉 Anti-Aliasing은 물체의 경계면에서만 이루어지면 되며, 본 시스템과 같이 렌더링의 단위가 삼각형인 경우에는 삼각형의 모서리에만 Anti-Aliasing 처리가 필요하고 이 모서리는 직선이기 때문에 결론적으로 직선에 대한 빠른 Anti-Aliasing 방법의 개발이 필요한 Anti-Aliasing의 전부이다.

따라서, 본 시스템에서는 모든 물체들이 렌더링 된 후에 그 물체들의 경계면에서의 Anti-Aliasing이 처리된다. 하지만, 앞에서 언급한대로 본 시스템에서는 그리고자 하는 물체와 배경의 색깔이 언제든 변할 수 있기 때문에 이에 대한 고려가 필요하다.

양 끝점  $(x0, y0)$ 과  $(x1, y1)$ 이 주어졌을 때, 점  $(x, y)$ 을 원점으로 평행 이동함으로써  $y=kx$  형태의 직선의 식을 얻을 수 있다. 이 때,  $0 \leq k \leq 1$ 일 경우에 식(4)을 적용시키면 다음과 같다.

$$\begin{aligned} I(x, kx) &= I_0(kx - kx) \\ I(x, kx) &= I_0 - I(kx - kx) \end{aligned} \quad (6)$$

이 때, 필요한 것은 주어진  $x$ 에 대하여  $(x, kx)$ 와  $(x, kx)$ 의 위치와  $I(x, kx)$ 와  $I(x, kx)$ 의 값이다. 이 모든 값은 기울기  $k$ 를 이용하여 구할 수 있다. 직선이 그려질 디바이스의 밝기 정보가  $2^m - 1$ 이라 하고 배경이  $2^m - 1$ 의 밝기 정보를 갖고, 직선이 0의 밝기 정보를 갖는다면,  $I(x, kx)$ 와  $I(x, kx)$ 의 값은 다음 식(6)에 의해서 결정될 수 있다.

$$\begin{aligned} I(x, kx) &= I_0(kx - kx) = (2^m - 1)*D \\ I(x, kx) &= I_0 - I(kx - kx) \\ &= (2^m - 1) - (2^m - 1)*D \end{aligned} \quad (6)$$

이 때, Display Device는 0으로 초기화된 후,  $x0$ 에서  $x1$ 까지 한 화소씩 증가할 때마다, 기울기  $k$ 만큼 같이 증가되어 가고,  $D$ 가 1보다 커지면,  $y$ 값을 증가시키고 정수 부분만큼 감소시킴으로써 얻어질 수 있다.

본 시스템에서와 같이 직선과 배경이 임의의 색을 가질 경우에, 식(6)은 다음과 같이 바뀐다.

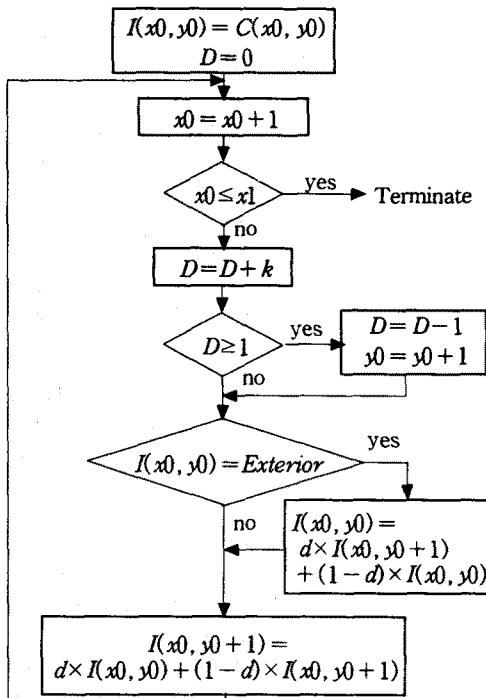
$$\begin{aligned} I(x, kx) &= I_0(kx - kx) \\ &= C(x, kx) - (C(x, kx) - C(x, kx))*D \\ I(x, kx) &= I_0 - I(x - kx) \\ &= C(x, kx) - I(x, kx) \\ &= (C(x, kx) - C(x, kx))*(1 - D) \end{aligned} \quad (7)$$

이 때,  $C(i, j)$ 는 점  $(i, j)$ 에서의 화소값이다.

이와는 달리 Anti-Aliasing 방법이 Solid Model에 적용될 때, 즉, 물체의 경계면에 적용될 때는 위의 식(6)이나 (7)을 그대로 적용할 수 없다. 이 때에는 물체의 경계면에서만 High Frequency가 나타나게 되므로 두 점 중 배경에 접한 점을 찾아낸 후 그 점에서의 값만 다음 식에 의해서 배경색과 합해주면 된다.

$$I = dIo + (1 - d)Ib \quad (8)$$

이때,  $Io$ 는 물체의 색을,  $Ib$ 는 배경색을 뜻하며,  $d$ 는 배경에 접한 점과 물체의 경계면과의 거리인데, 본 시스템의 경우처럼 직선의 경우에는 기울기  $k$ 를 이용한다. 이 알고리즘의 순서도는 <그림 4>와 같다.

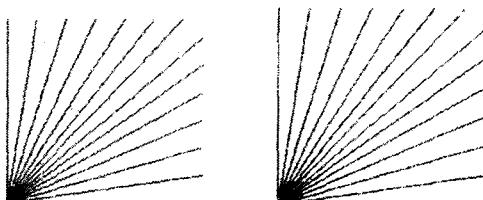


〈그림 4〉 Anti-Aliasing Edge Generator( $0 \leq k \leq 1$ )

### 2.2.2 다른 알고리즘과의 비교

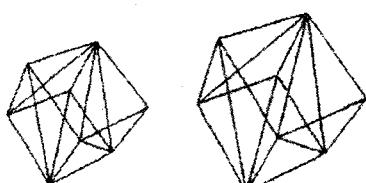
개선된 Two-Point Anti-Aliasing기법을 이용한 알고리즘과 브레센행의 선그리기 알고리즘, 그리고 대표적인 Anti-Aliasing알고리즘인 A-Buffer알고리즘을 서로 비교하여 본다.

〈그림 5〉에서 보는 바와 같이(a)Bresenham에 의해 그려진 사선보다 (b)의 Two-Point Anti-Aliasing기법으로 그려진 것이 더 Aliasing효과가 적다.



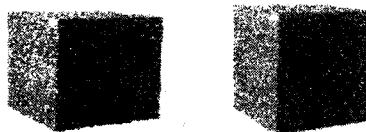
(a)Bresenham      (b)Two-Point Anti-Aliasing  
〈그림 5〉 직선그리기

〈그림 6〉은 WireFrame Model에 두 알고리즘을 적용시킨 것이다. 속도는 (a)가 조금 빨랐으나 효과에 비해 아주 적은 차이만이 있을 뿐이었다.



(a)Bresenham      (b)Two-Point Anti-Aliasing  
〈그림 6〉 WireFrame Model

〈그림 7〉은 Solid Model에 개선된 Two-Point Anti-Aliasing 알고리즘을 적용하여 A-Buffer알고리즘과 비교하였다. 시간은 2배이상의 차이를 보였으나, 품질은 유사함을 볼 수 있다.



(a)Two-Point Anti-Aliasing    (b)A-Buffer  
〈그림 7〉 Solid Model

### 3. 결 론

본 논문에서는 실시간 렌더링 시스템에 사용될 수 있는 Anti-Aliasing방법을 제시하였다. 이 방법의 특징은 A-Buffer알고리즘을 비롯한 기존의 방법들에 비해 뛰어나 속도를 갖는다는 것이다. A-Buffer알고리즘의 경우 한 화소의 색을 결정하기 위해 한 화소를 32개로 나누어 복잡한 화소의 구조를 가지고, 색을 결정하기 위한 여러 가지 계산이 행해지게 된다. 따라서 필연적으로 본 논문에 제시된 방법에 비해 속도가 떨어지게 된다. 물론, Two-Point Anti-Aliasing방법은 인접한 두 화소만으로 Anti-Aliasing효과를 내기 때문에, 한 화소를 여러 개의 부분 화소로 나누어 계산하는 기존의 방법들에 비해 성능이 떨어진다는 문제가 남아 있지만, 3차원 게임과 같이 시간이 중요한 요소인 응용 프로그램의 경우에 유용하게 사용될 수 있으리라 본다.

### (참 고 문 헌)

- [1] Oppenheim, A.V. and R.W. Shafer, Digital Signal Processing, Prentice-Hall, Englewood, N.J., 1975
- [2] Xiaolin Wu, "An Efficient Antialiasing Technique", Computer Graphics, Vol. 25, Number 4, pp 143-152, July 1991
- [3] J.E.Bresenham, "Algorithm for computer control of digital plotter", IBM syst. J., Vol. 4, Number 1, pp 25-30, 1965
- [4] Loren Carpenter, "The A-Buffer, an Antialiasing Hidden Surface Method", Proceedings of SIGGRAPH '84, pp 103-108, 1984
- [5] Edwin Catmull, "A Hidden-Surface Algorithm with Anti-Aliasing", Proceedings of SIGGRAPH '79, pp 6-11, 1978
- [6] Andreas Schilling, "A New Simple and Efficient Anti-Aliasing with Subpixel Masks", Computer Graphics, Vol. 25, Number 4, pp 133-141, July 1991
- [7] Crow, Frank, "The Aliasing Problem in Computer-Generated Shaded Images", CACM November 1977
- [8] D.Hearn, M.P.Baker, Computer Graphics, 2nd ed, C version, Prentice Hall, 1994
- [9] Alan Watt, 3D Computer Graphics, Addison-Wesley publishing Co., 1993
- [10] Jackie Neider, Tom Davis, Mason Woo, "OpenGL Programming Guide OpenGL ARB", Addison-Wesley publishing Co., 1997