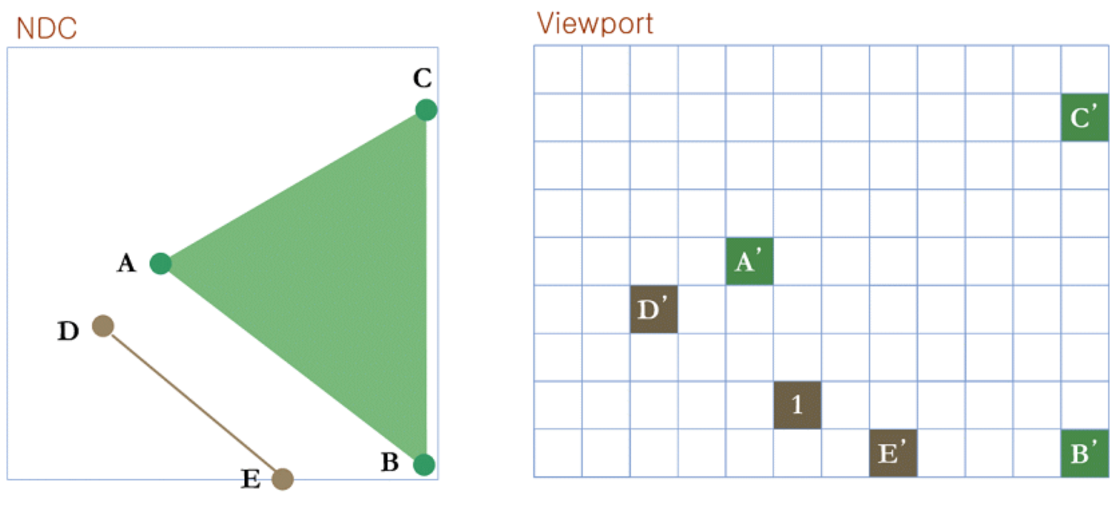
Note 11-Rasterization

Metaverse

Scan Conversion

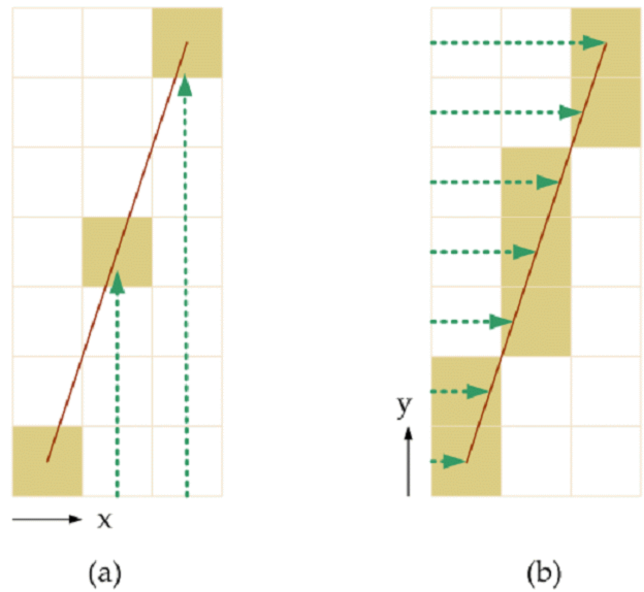
- Raster = 화소 (pixel)
- 물체를 표현하기 위한 화소로의 사상
- CVV --> Viewport
    - 3D vertex position (선분, 내부면)--> 2D screen space position



(a) (b)

Line rasterization

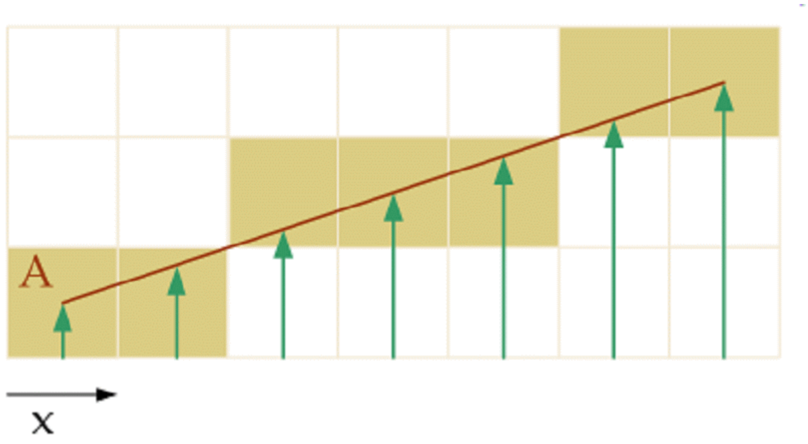
- **기울기**를 기준으로 샘플링
  - 1보다 크면 y 좌표를 증가
  - 1보다 작으면 x 좌표를 증가



x좌표 기반 **y좌표 기반**

Line rasterization

- **교차점 계산에 의한 변환**

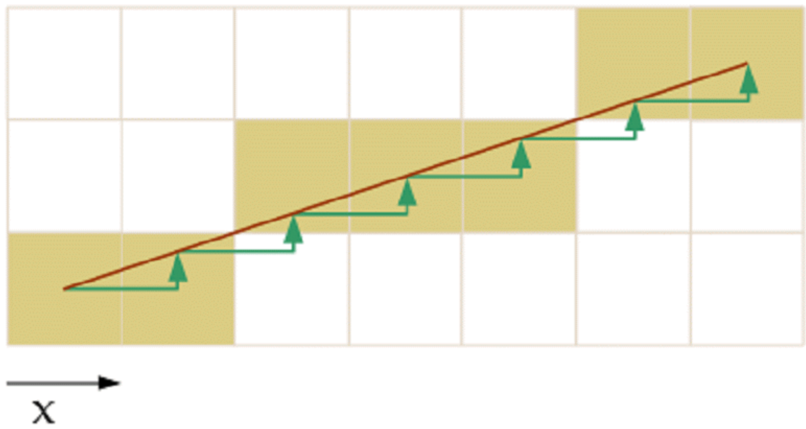



```
void LineDraw(int x1, int y1, int x2, int y2){
    float y, m;
    int dx, dy;
    dx = x2 - x1;  dy = y2 - y1;
    m = dy / dx;      //
    for (x = x1; x <= x2; x++) {
        y = m*(x - x1) + y1;
        DrawPixel(x, round(y));
    }
}
```

: 부동소수 곱셈으로 인한 속도저하

Line rasterization

- **Digital Differential Analyzer**

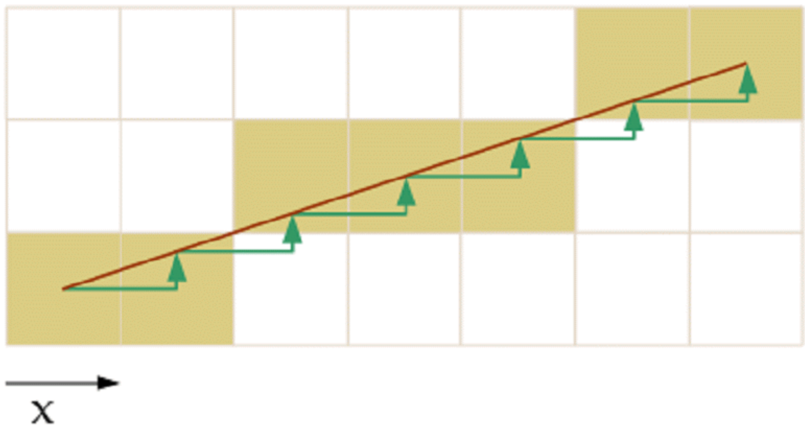



```
void LineDraw(int x1, int y1, int x2, int y2) {
    float m, y;   int dx, dy;
    dx = x2 - x1;  dy = y2 - y1;
    m = dy / dx;
    y = y1;
    for (int x = x1; x <= x2; x++) {
        DrawPixel(x, round(y));
        y += m;
    }
}
```

: 부동소수 곱셈 --> 부동소수 덧셈

Line rasterization

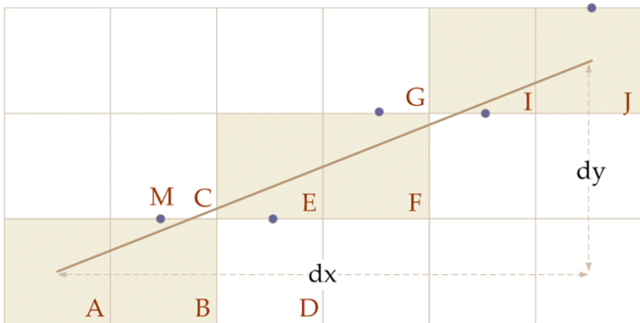
- **Digital Differential Analyzer 문제점**

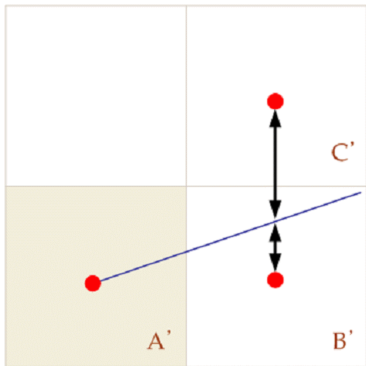


- 부동소수 연산
  - 부동소수 덧셈
  - 정수 연산에 비해 느림
- 반올림 연산
  - round( ) 함수 실행에 걸리는 시간
- 연산 결과의 정확도
  - 부동소수의 경우 뒷 자리가 잘려나감
  - 연속적인 덧셈에 의한 오류 누적
  - 선택된 화소가 실제 선분에서 점차 멀어져서 표류(Drift)

Bresenham Algorithm

- 중점 알고리즘 (midpoint algorithm)



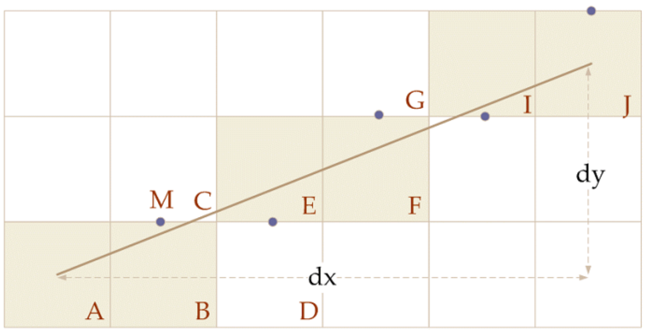


A 선택된 다음...

- 다음 화소는 B, C 중 하나
- 화소 중심과 선분간의 수직 거리에 의해 판단
- 선분이 중점 M의 아래에 있으면 화소 B, 위에 있으면 화소 C를 선택

Bresenham Algorithm

- 중점 알고리즘 (midpoint algorithm)



$$y = \frac{dy}{dx}x + b$$

$$ydx = xdy + bdx$$

$$f(x, y) = ydx - xdy - bdx = 0$$

$$F(x, y) = 2ydx - 2xdy - 2bdx = 0$$ **편한계산을 위해..**

M의 좌표: <x+1, y+1/2>

F(x+1,y+1/2) -F(x,y)= 2dy-dx

$$F(x, y) = 2dy - dx$$

$$\text{if } (F(x, y) > 0) \; Select \; NorthEast \; Pixel;$$

$$else \; Select \; East \; Pixel;$$

정수연산에 의한 속도증가 + 하드웨어로 구현

Polygon rasterization

- 수식표현

  - 현시적 표현(Explicit Representation)

    $y = 2x + 4$ **장점: x 입력에 대한 y 출력**

  - 묵시적 표현(Implicit Representation)

    $f(x, y) = y - 2x - 4 = 0$ **평면에 위에 있는지.. 앞에있는지..**

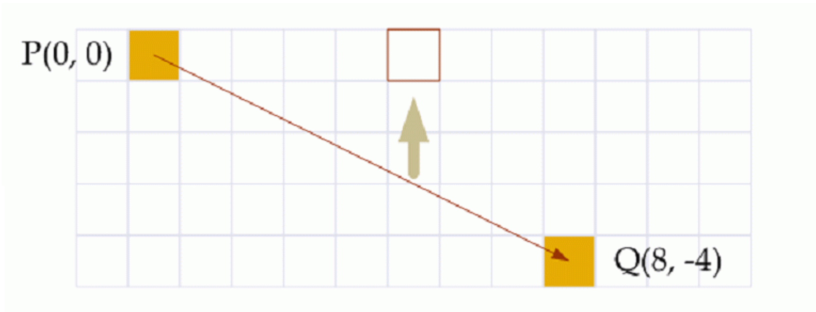
  - 파라미터 표현(Parametric Representation)

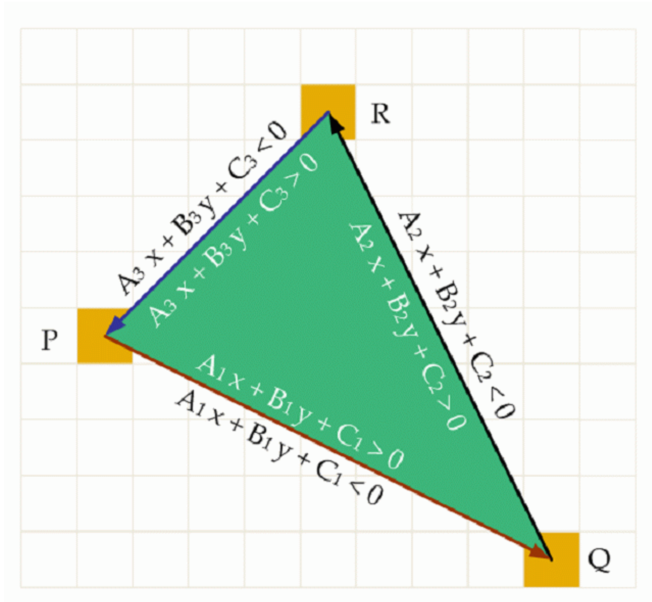
    $(t, 2t + 4)$ 또는 $(t^2 + 1, 2(t^2 + 1) + 4)$ **t값에 의한 시각화**

Polygon rasterization

- 상하 및 내외 판단

$$y - y_1 = (y_2 - y_1)(x - x_1)/(x_2 - x_1)$$
$$(y - y_1)(x_2 - x_1) = (y_2 - y_1)(x - x_1)$$
$$f(x, y) = (y_1 - y_2)x + (x_2 - x_1)y + x_1y_2 - x_2y_1 = 0$$

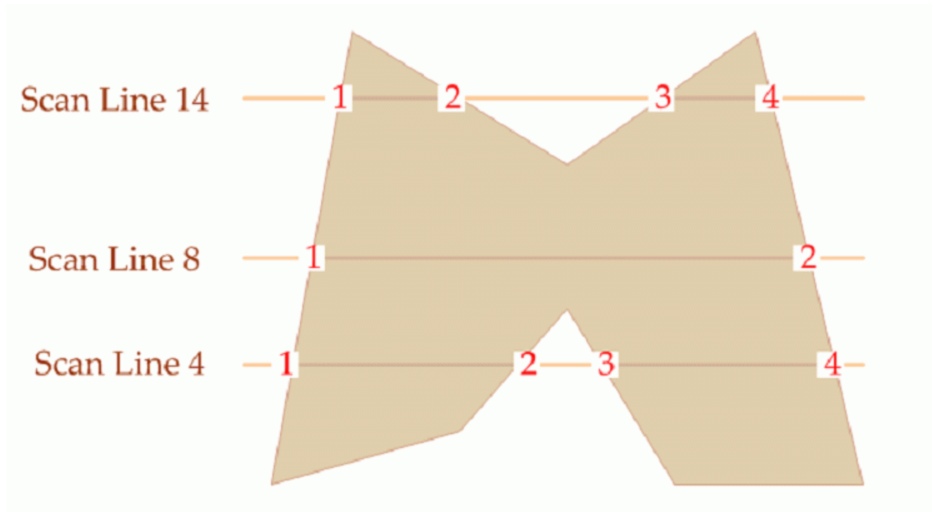




- 선분은 반 시계 방향으로 진행할 때 진행방향의
  - 왼쪽에 대해서는 f(x, y) > 0
  - 오른쪽에 대해서는 f(x, y) < 0

Polygon rasterization

- 보다 일반적인 방법: Scan Line Fill Algorithm

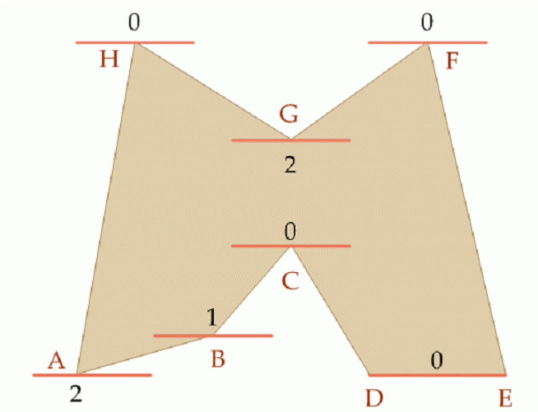


홀수 규칙(Odd Parity Rule, Even-Odd Rule)

- 홀수번째 교차화소부터 짝수번째 교차화소 직전 직전까지 채움
- 짝수번째를 포함하지 않는 이유 (<= 아닌 < 의 이유): 원본도형의 길이보존
  - 14번: $1 \le x < 2$, $3 \le x < 4$
  - 8번: $1 \le x < 2$
  - 4번: $1 \le x < 2$, $3 \le x < 4$

Polygon rasterization

- 보다 일반적인 방법: Scan Line Fill Algorithm



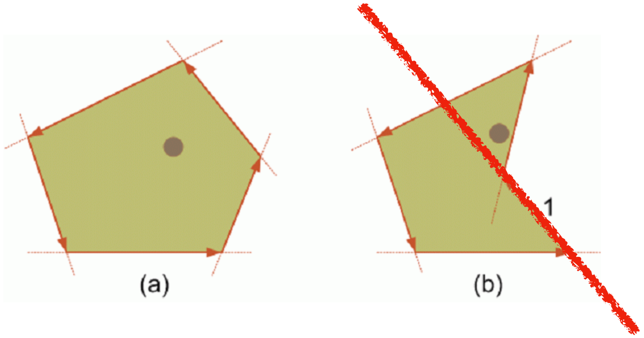
길이보존
- 극대점: 교차하지 않은 것으로 간주(H, F, C)
- 극소점:각각 교차한 것으로 간주: 2번(G, A)

극대극소: 1번 교차(B): (AB)(BC)2개의 선분으로 분할

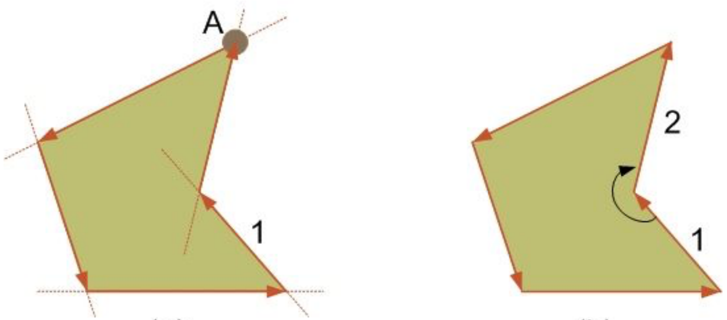
주사선과 평행
- 선분이 없는 것으로 간주(DE)

Inside Outside Test

- 진행방향의 왼쪽이 내부
  - 볼록 다각형에서만 성립
  - 오목 다각형의 경우 다각형 분할(Tessellation)에 의해 볼록 다각형의 집합으로 변형



(a) (b)

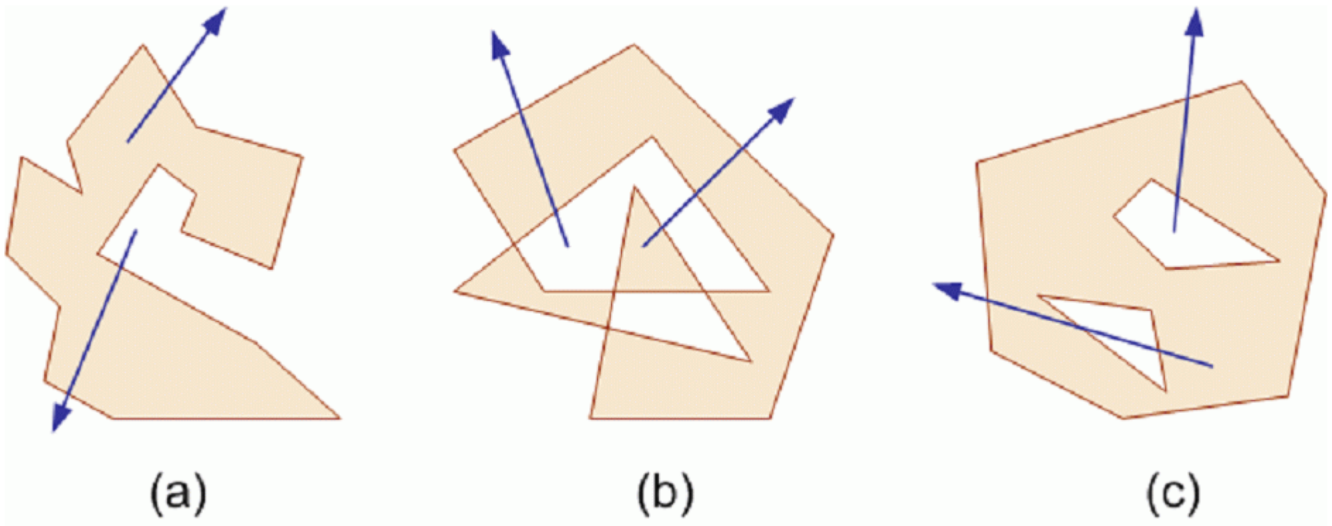


(a) (b)

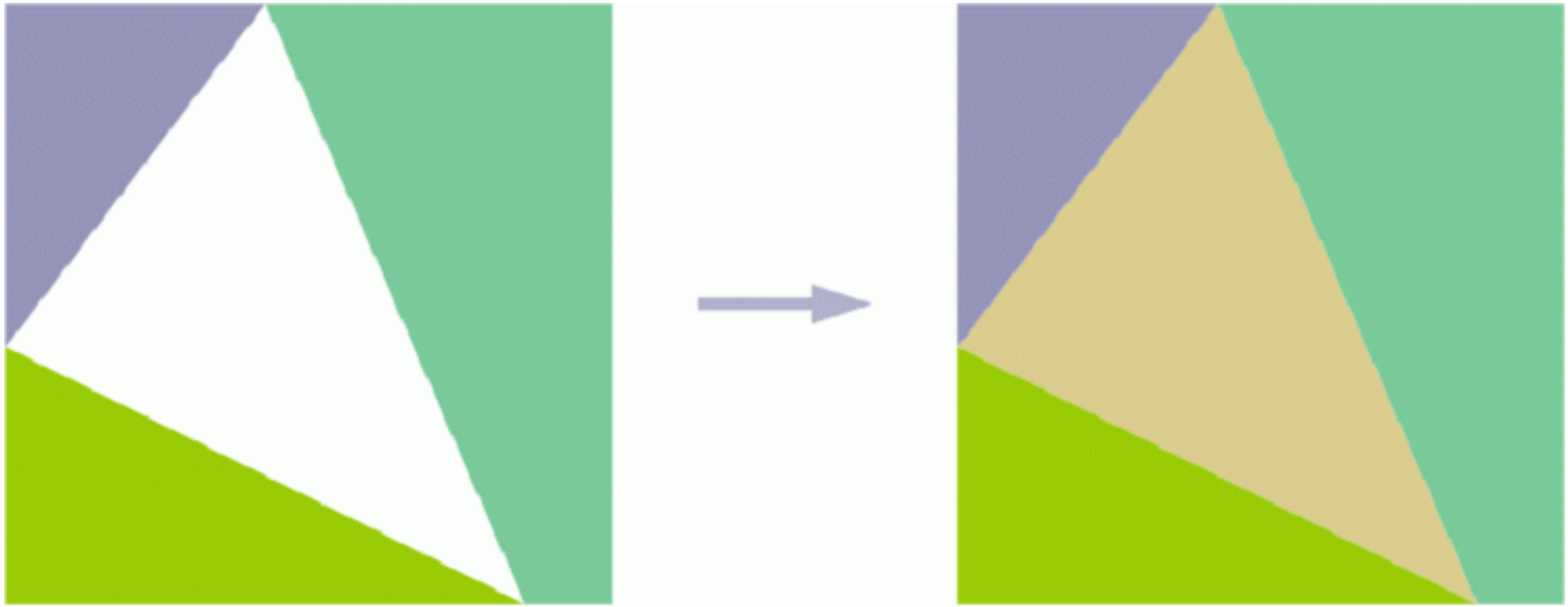
볼록, 오목의 판단은? 정점을 대상으로 모든 변과 테스트
이웃하는 변간의 내적값

Inside Outside Test

- 홀수 규칙(Odd Parity Rule, Even-Odd Rule)
  - 볼록, 오목에 무관하에 내외부 판정
  - 내부점으로부터 외부를 향한 직선은 다각형과 반드시 홀수 번 교차



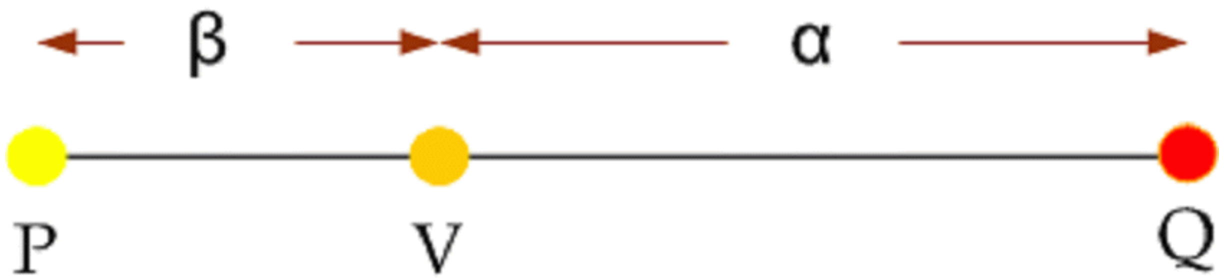
동색상으로 내부를 채우려면...



Flood Fill algorithm

Vertex의 색상이 다르면? --> 보간 (interpolation)

Barycentric Coordinate



$$V(t) = P + t\,(Q - P) = (1 - t)P + tQ = \alpha P + \beta Q$$

$$0 \leqq \alpha, \beta \leqq 1,\ \alpha + \beta = 1$$

Barycentric Coordinate

- 삼각형의 무게중심 좌표 (α, β, γ)

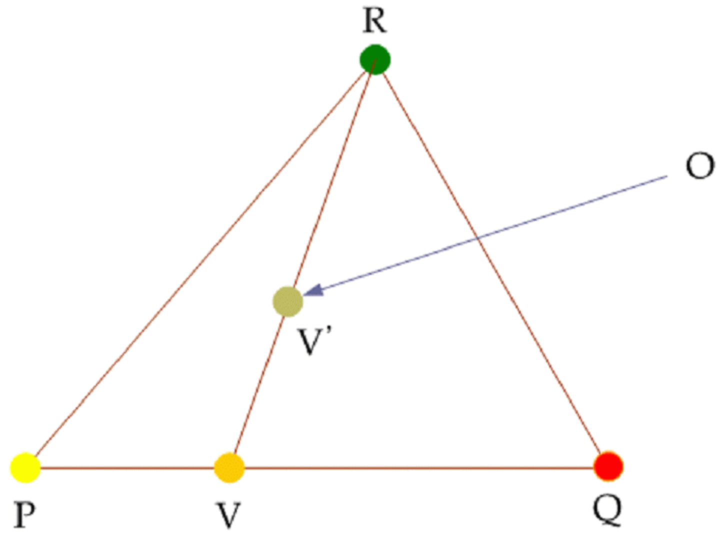
$$V' = V + s(R - V)$$

$$= P + t(Q - P) + s(R - (P + t(Q - P)))$$

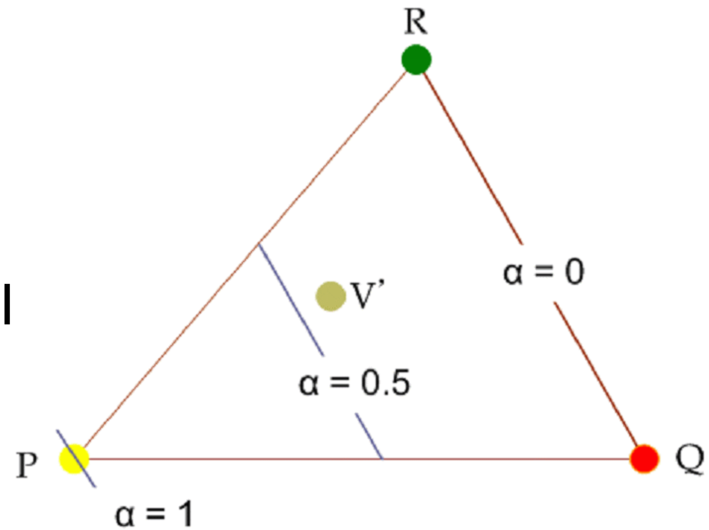
$$= (1 - t - s + st)P + (t - st)Q + sR$$

$$= \alpha P + \beta Q + \gamma R$$

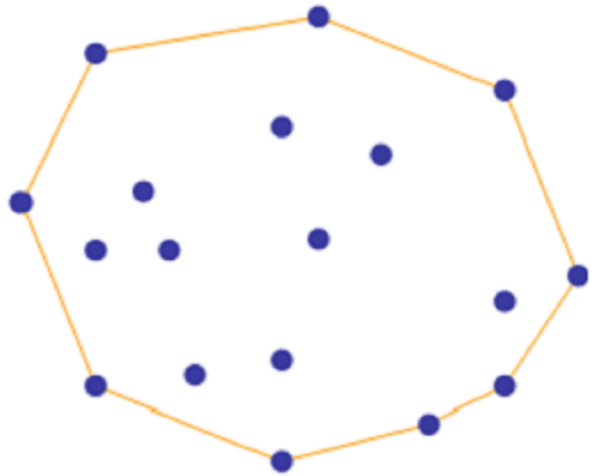
$$0 \leqq \alpha, \beta, \gamma \leqq 1, \ \alpha + \beta + \gamma = 1$$



- α 의 의미



Convex Hull

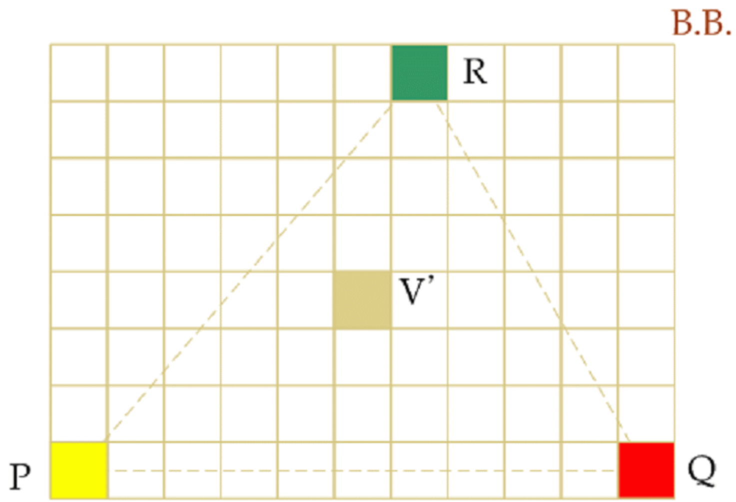


$$V = t_1 P_1 + t_2 P_2 + \dots + t_n P_n$$

$$0 \leqq t_1, t_2, \dots, t_n \leqq 1,\ t_1 + t_2 + \dots + t_n = 1$$

- 컨벡스 헐
  - 주어진 점을 모두 포함하는 가장 작은 볼록 다각형
- 컨벡스 헐 특성(Convex Hull Property)
  - 위 식으로 표현된 정점 V는 항상 컨벡스 헐 내부에 존재

Interpolation by B.C



$$r = \alpha P_r + \beta Q_r + \gamma R_r$$

$$g = \alpha P_g + \beta Q_g + \gamma R_g$$

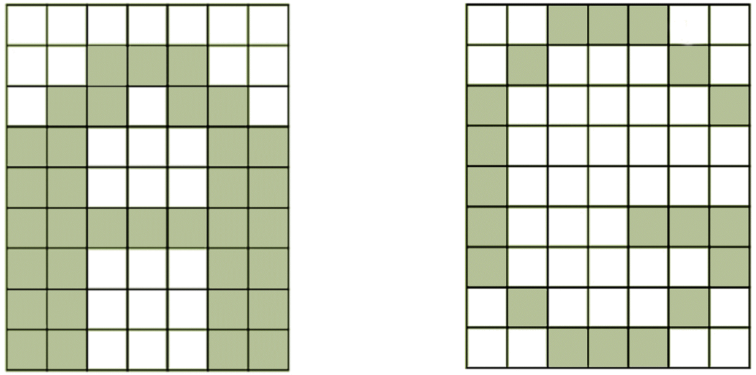
$$b = \alpha P_b + \beta Q_b + \gamma R_b$$

$$z = \alpha P_z + \beta Q_z + \gamma R_z$$

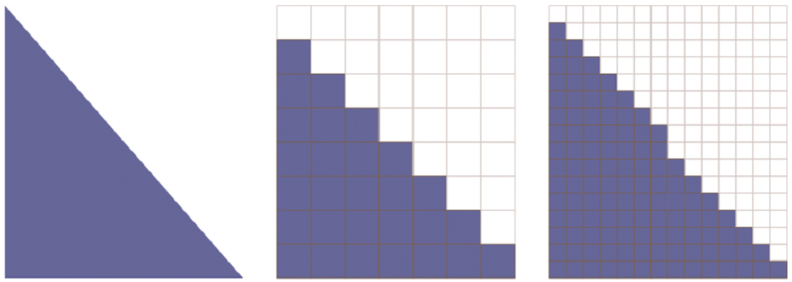
Depth

Antialiasing

- Bitmap



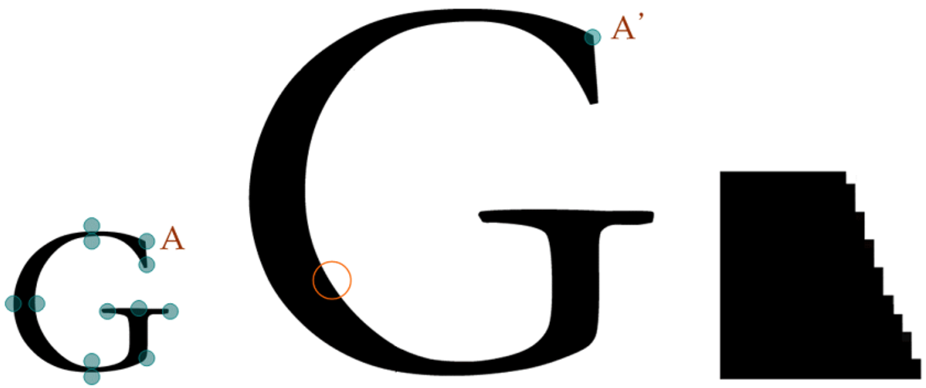
- Alias
  - 계단(Stair-step, Jaggies) 모양의 거친 경계선
    - 비트맵 표현에서는 화소 단위로 근사화 할 수 밖에 없기 때문
    - 무한 해상도를 지닌 물체를 유한 해상도를 지닌 화소 면적 단위로 근사화 할 때 필연적으로 일어나는 현상



(a) (b) (c)

Antialiasing

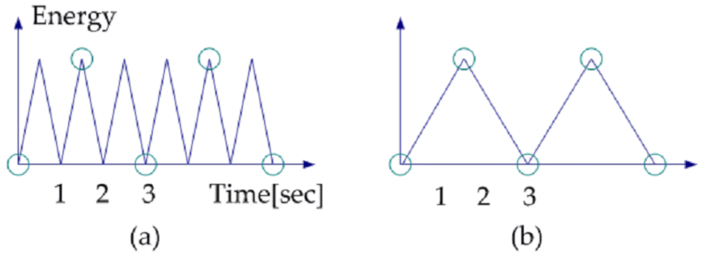
- Postscript (vector)
  - 영상의 윤곽선을 수식으로 표현
    - 특징적인 정점의 좌표, 이를 연결하는 보간 곡선의 수식을 명시
    - 특징적인 정점 = 제어점(Control Point)
    - 확대된 정점 위치에 보간 곡선을 다시 적용
      - 비트맵 보다 매끄러운 곡선
      - 에일리어싱 완화



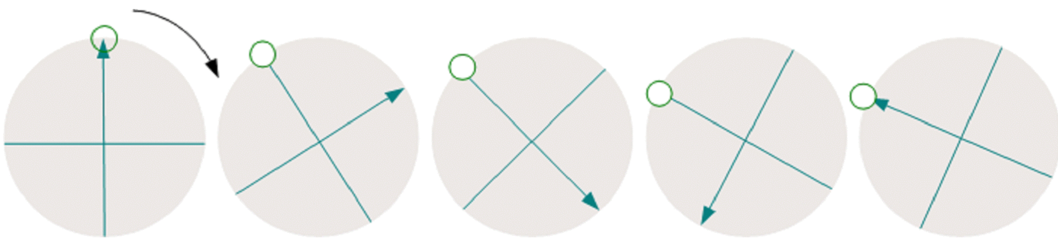
(a) (b) (c)

Antialiasing

- Aliasing
  - 언더 샘플링으로 인함
    - 신호의 복원
    - 나이퀴스트 주파수

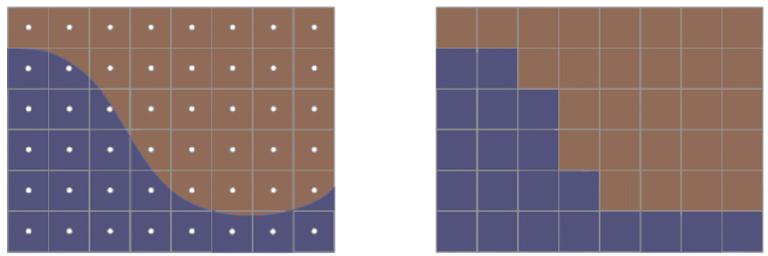


  - Stroboscopic Effect
    - 시간적 에일리어싱



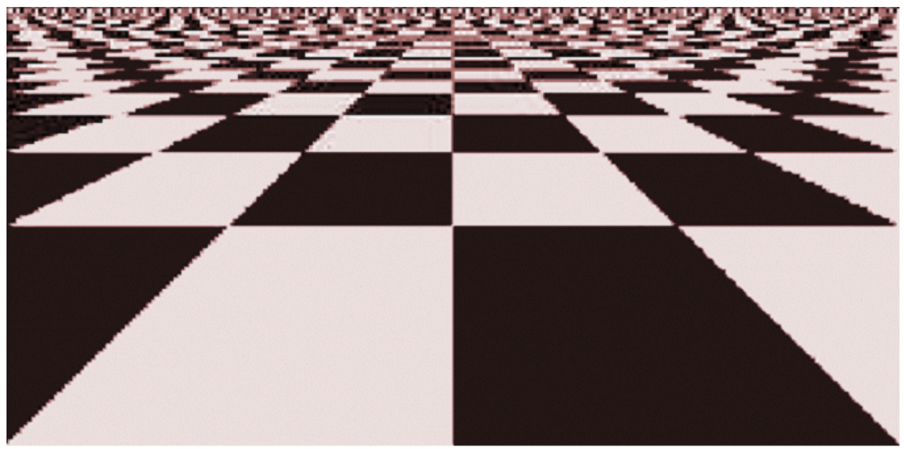
Antialiasing

- Aliasing
  - 점 샘플링으로 인한 에일리어싱



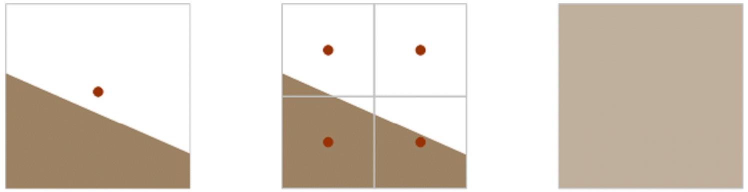
(a) (b)

  - 므와르 패턴
    - 뒷 부분의 높은 주파수를 화소 크기가 수용하지 못함



Antialiasing

- Anti-Aliasing
  - 수퍼 샘플링(Super-Sampling)
    - 부분화소에서 샘플링. 사후 필터링
    - 부분화소의 평균값을 반영

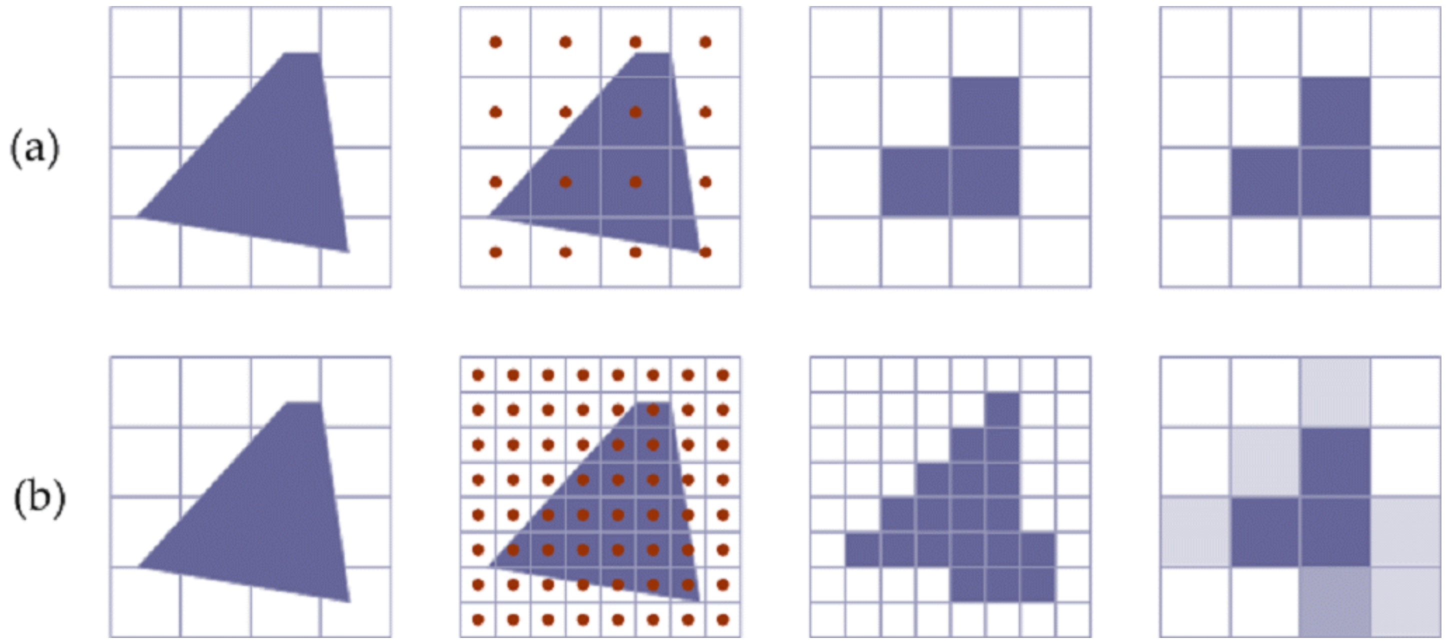


(a) (b) (c)

- 지터에 의한 수퍼 샘플링
  - 물체 자체가 불규칙이라면 불규칙 샘플링이 유리

(a) (b) (c)

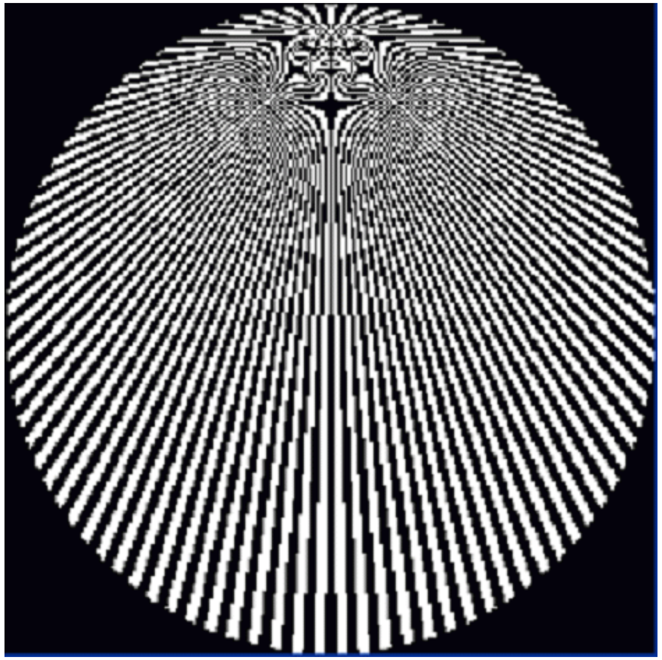
Antialiasing



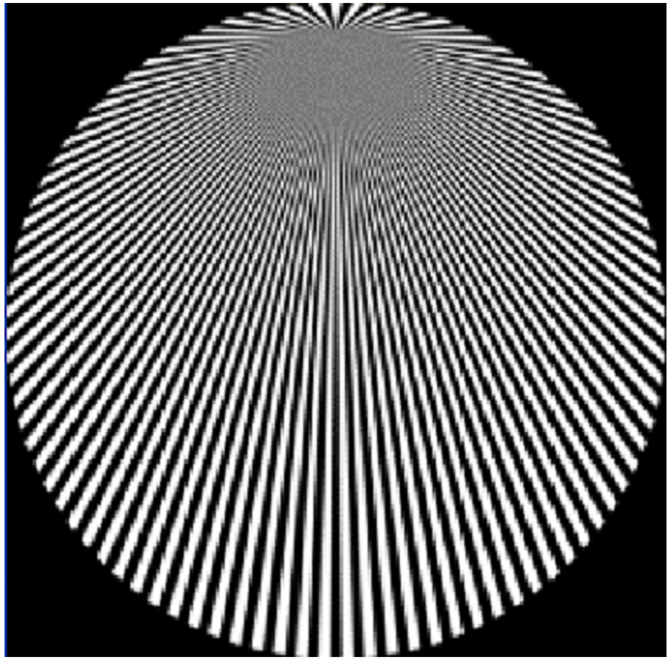
without supersampling

with supersampling

Antialiasing



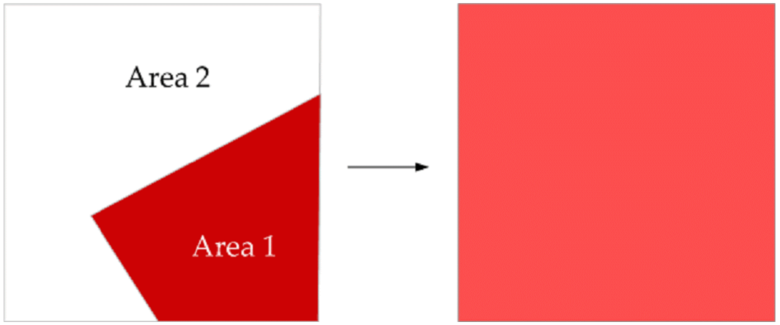
supersampling

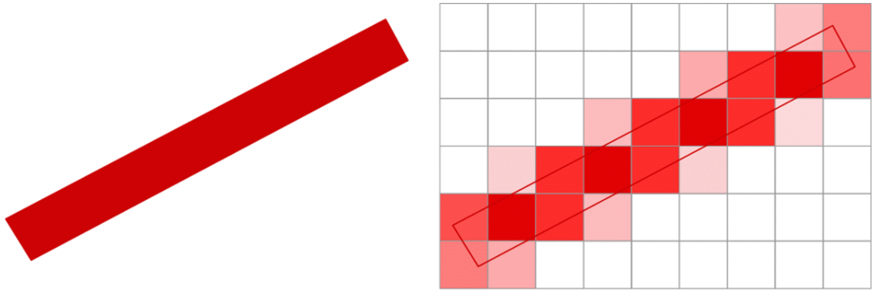


supersampling with jittering

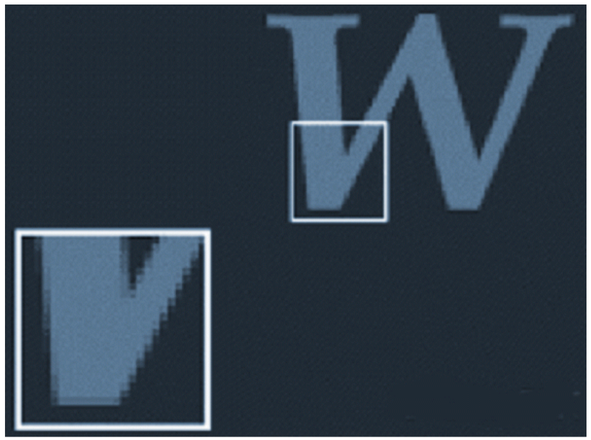
Antialiasing

- Area-Sampling





(a) (b)



Antialiasing

- Isotropic Vs. Anisotropic