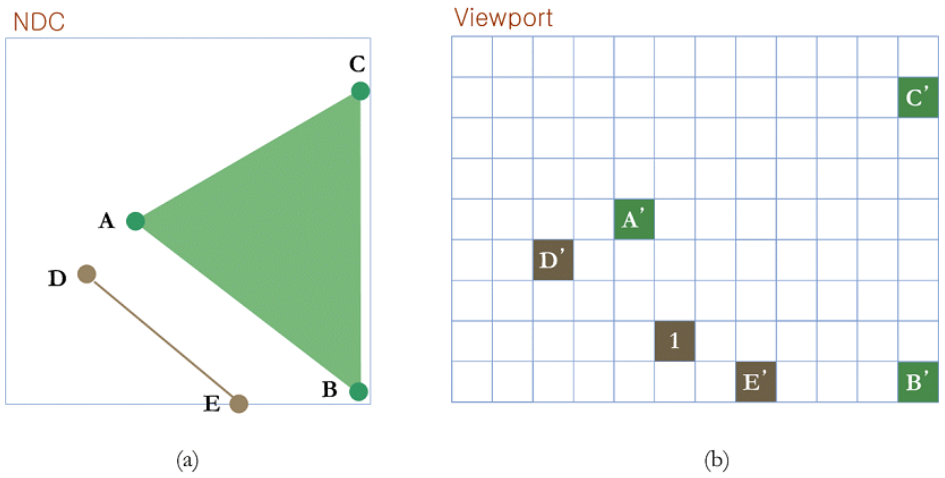
9장. 래스터 변환

- 학습목표

- 래스터 변환이 필요한 이유를 이해한다.
- 지-버퍼 알고리즘에 의한 은면제거가 래스터 변환과 병행되어야 하는 이유를 이해한다.
- 선분의 래스터 변환에 있어서 브레스넘 알고리즘의 장점을 이해한다.
- 주사선 채움 알고리즘 및 활성화 선분 리스트의 사용법을 이해한다.
- 경계채움 알고리즘과 홍수채움 알고리즘의 차이점을 이해한다.
- 선형보간 방법을 이해한다.
- 비트맵과 포스트스크립트의 개념상 차이점 및 저장방식의 차이점을 이해한다.
- 에일리어싱이 발생하는 이유와 앤티-에일리어싱 기법에 대해 이해한다.

래스터 변환(Rasterization)

- 래스터 변환 또는 스캔 변환(Scan Conversion)
  - Raster = 화소
  - 물체를 표현하기 위해 어떤 화소를 밝힐 것인지를 결정하는 작업
  - 정규화 가시부피에서 뷰포트로의 사상
  - 정점좌표를 화면좌표로 변환한 결과를 기준으로
    - 선분을 화면좌표로 변환
    - 내부면을 화면좌표로 변환



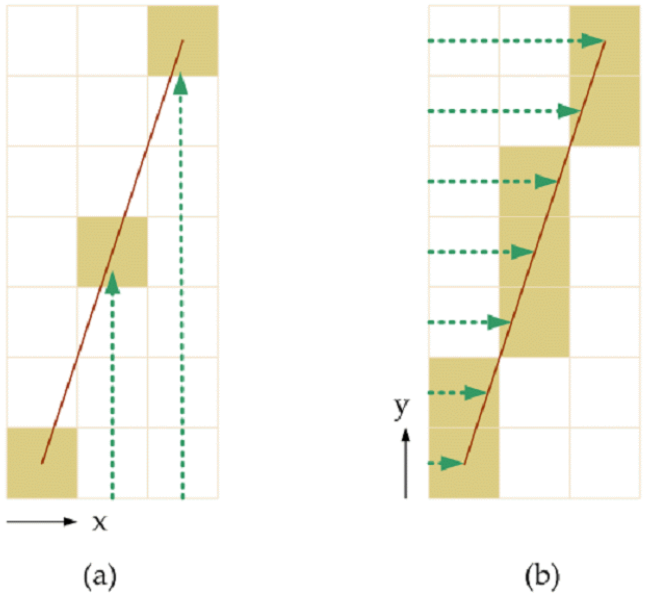
지엘의 래스터변환

- 은면제거와 동시에 진행
  - 깊이와 색을 보간
  - 정점의 z 값으로부터 선분 및 내부면의 깊이를 보간
  - 정점의 색으로부터 선분 및 내부면의 색을 보간

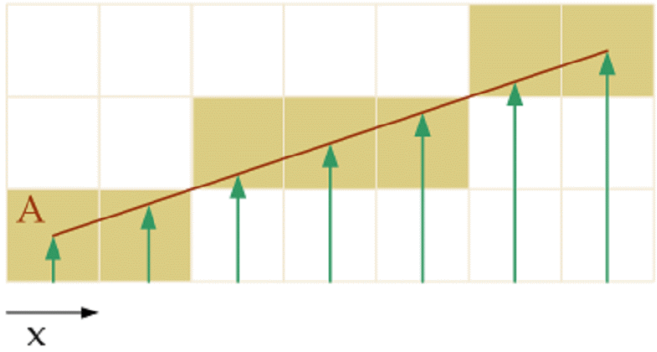
- 화면에 보이는 모든 것은 래스터 변환결과
  - 최대의 연산속도, 최대의 정확성이 요구됨

선분의 래스터 변환

- 기울기를 기준으로 샘플링
  - 1보다 크면 y 좌표를 증가
  - 1보다 작으면 x 좌표를 증가



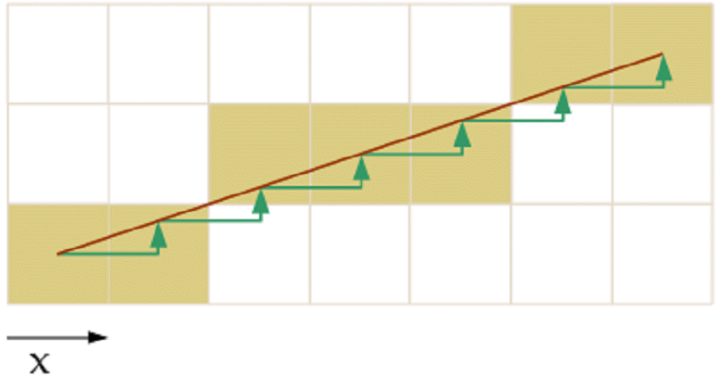
교차점 계산에 의한 변환



- 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

DDA(Digital Differential Analyzer)



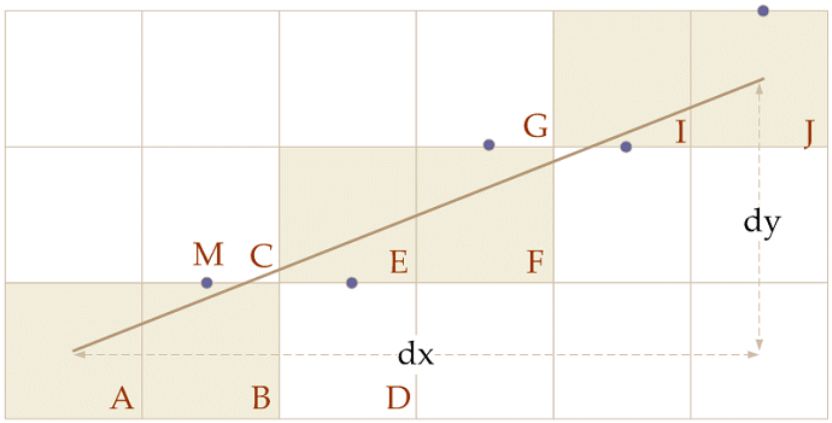

```
void LineDraw(int x1, int y1, int x2, int y2) {
    float m, y;   int dx, dy;
    dx = x2 - x1;  dy = y2 - y1;
    m = dy / dx;
    y = y1;
    for (int x = x1; x <= x2; x++) {
        DrawPixel(x, round(y));
        y += m;
    }
}
```

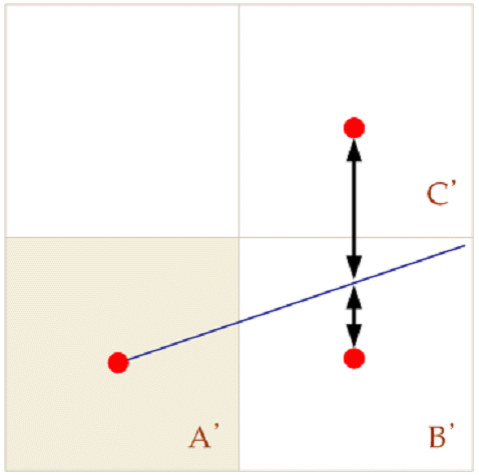
DDA 단점

- 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)

브래스넘 알고리즘

- 브레스넘 알고리즘(Bresenham Algorithm) 또는 중점 알고리즘(中點, Midpoint Algorithm)





- A 선택
  - 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

브래스넘 알고리즘

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

$$ydx = xdy + bdx$$

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

$$F(x, y) = 2ydx - 2xdy - 2bdx = 0$$

$$F(x, y)$$
$$= F(x1 + 1, y1 + 1/2)$$
$$= 2(y1 + 1/2)dx - 2(x1 + 1)dy - 2bdx$$

$$= 2y1dx + dx - 2x1dy - 2dy - 2bdx$$

$$F(x1, y1) = 2y1dx - 2x1dy - 2bdx = 0$$

$$F(x, y) = F(x1 + 1, y1 + 1/2) = dx - 2dy$$

- A= (x1, y1)이면 화소 B, C 경계선의 중점 M= (x1+1, y1+1/2)
- 결정변수 F(x, y)에 의해 중점이 선분의 위인지 아래인지를 판단

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

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

브래스넘 알고리즘

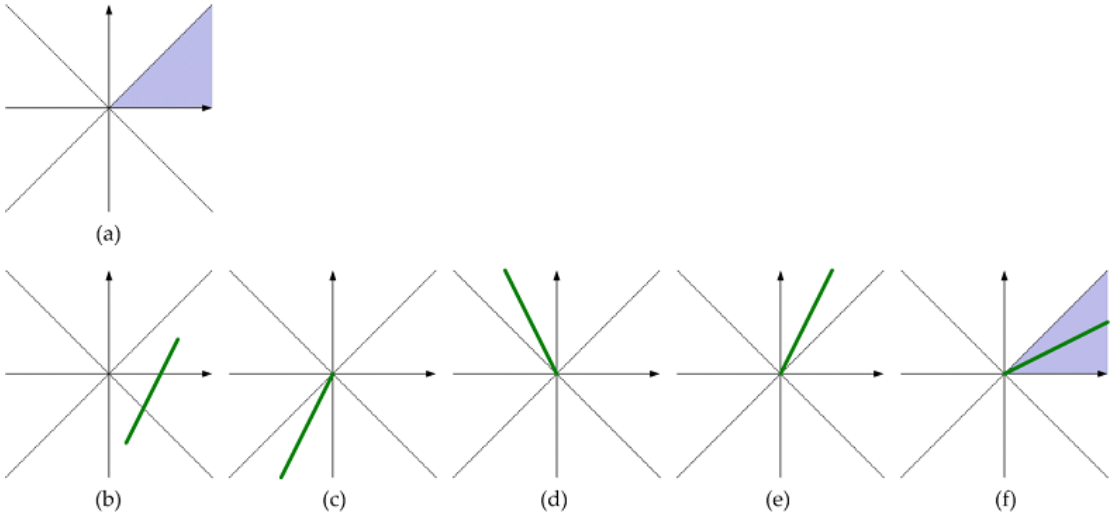
- 다음 결정변수와 현 결정변수의 차이

$$incE = F(x+1, y) - F(x, y) = 2dy$$
$$incNE = F(x+1, y+1) - F(x, y) = 2dy - 2dx$$

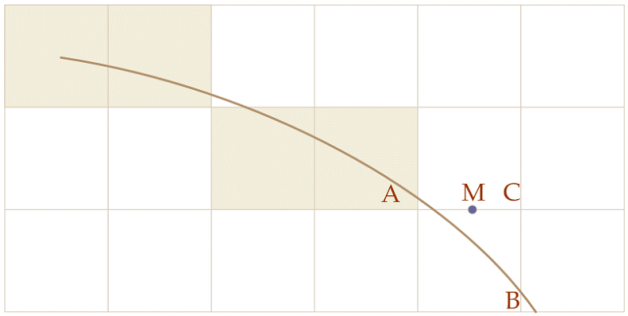
```
void MidpointLine(int x1, int y1, int x2, int y2){
    int dx, dy, incrE, incrNE, D, x, y;
    dx = x2 - x1; dy = y2 - y1;
    D = 2*dy - dx;                  결정변수 값을 초기화
    incrE = 2*dy;                   동쪽 화소 선택시 증가분
    incrNE = 2*dy - 2*dx;           동북쪽 화소 선택시 증가분
    x = x1;      y = y1;            첫 화소
    DrawPixel(x, y)                 첫 화소 그리기
    while (x < x2) {
        if (D <= 0) {               결정변수가 음수. 동쪽화소 선택
            D += incrE;             결정변수 증가
            x++;                    다음 화소는 동쪽
        }
        else{                       결정변수가 양수. 동북쪽 화소 선택
            D += incrNE;            결정변수 증가
            x++; y++;               다음 화소는 동북쪽
        }
        DrawPixel (x, y);           화소 그리기
    }
}
```

브래스넘 알고리즘

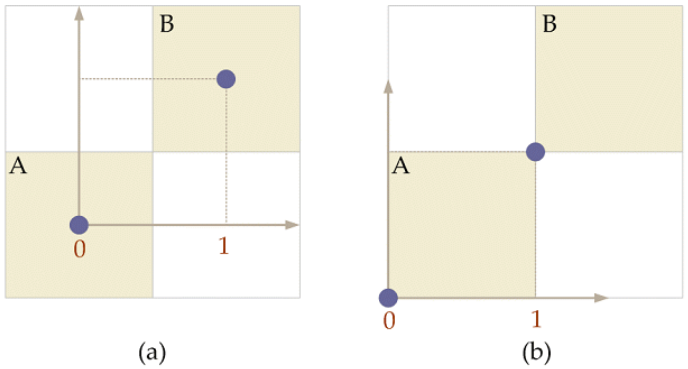
- 정수연산에 의한 속도증가 + 하드웨어로 구현
- 첫 팔분면에서만 정의
  - 다른 선분은 이동, 반사하여 적용

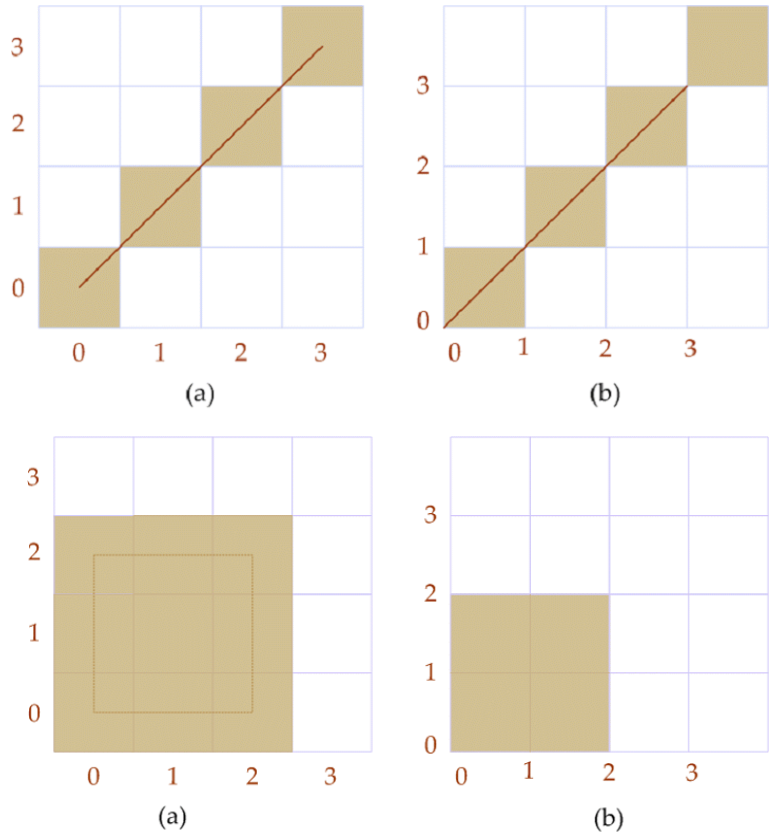


- 원 생성 알고리즘
  - 선분생성 알고리즘과 유사



화소좌표





- 화소의 좌하단을 기준으로 부여하는 것이 일반적
  - 선분길이 조정을 위해 마지막 화소는 제외시킴
  - 면적 조정을 위해 외곽 화소는 제외시킴

그래픽 수식 표현

- 현시적 표현(Explicit Representation)

$$y = 2x + 4$$

- 묵시적 표현(Implicit Representation)

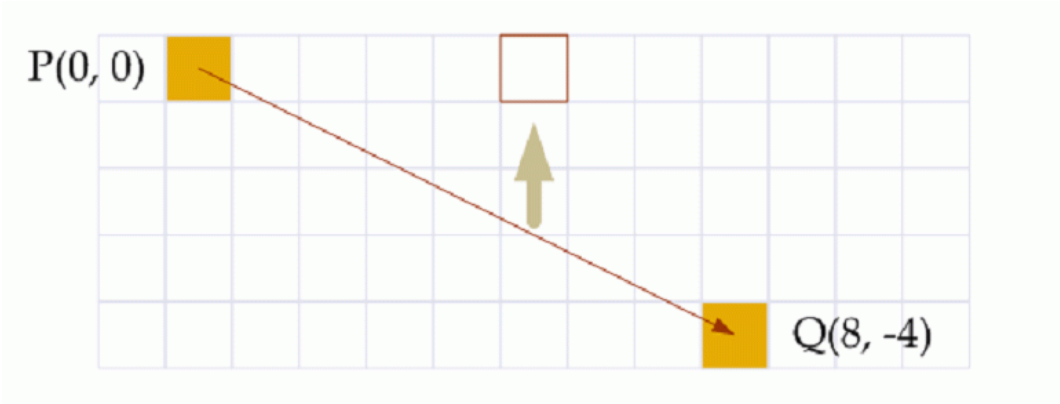
$$f(x, y) = y - 2x - 4 = 0$$

- 파라미터 표현(Parametric Representation)

$$(t, 2t + 4) \text{ 또는 } (t^2 + 1, 2(t^2 + 1) + 4)$$

  - 단일하지 않음
  - $X^2 + y^2 - 1 = 0$ => $(\cos\theta, \sin\theta)$

상하 및 내외 판단



$$y - y_1 = (y_2 - y_1)(x - x_1)/(x_2 - x_1)$$

$$(y - y_1)(x_2 - x_1) = (y_2 - y_1)(x - x_1)$$

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

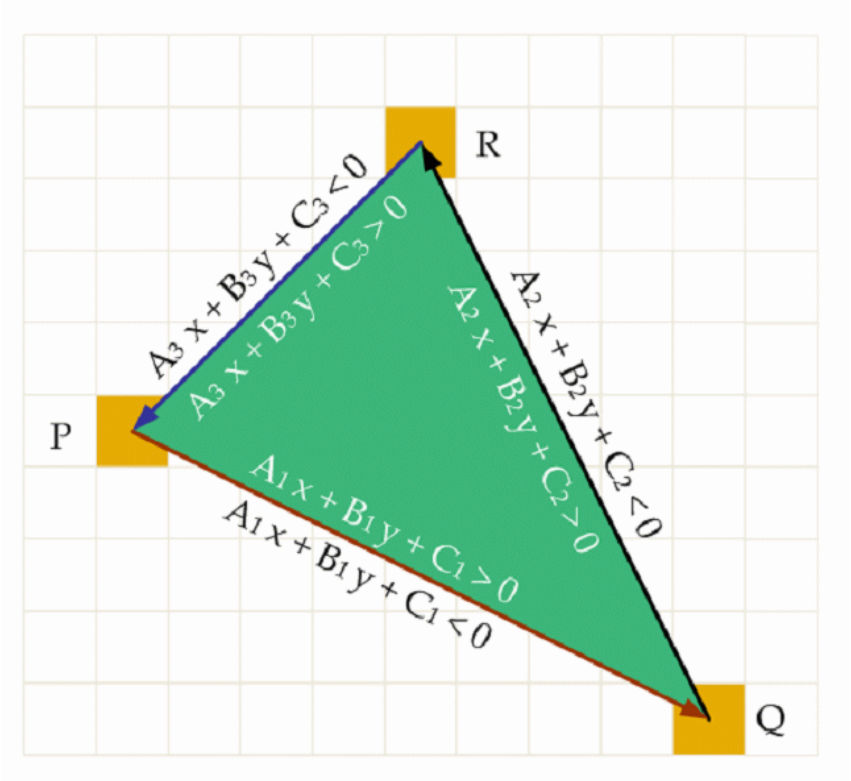
$$f(x, y) = (0 - (-4))x + (8 - 0)y + 0 - 0 = 0$$

$$f(x, y) = 4x + 8y = 0$$

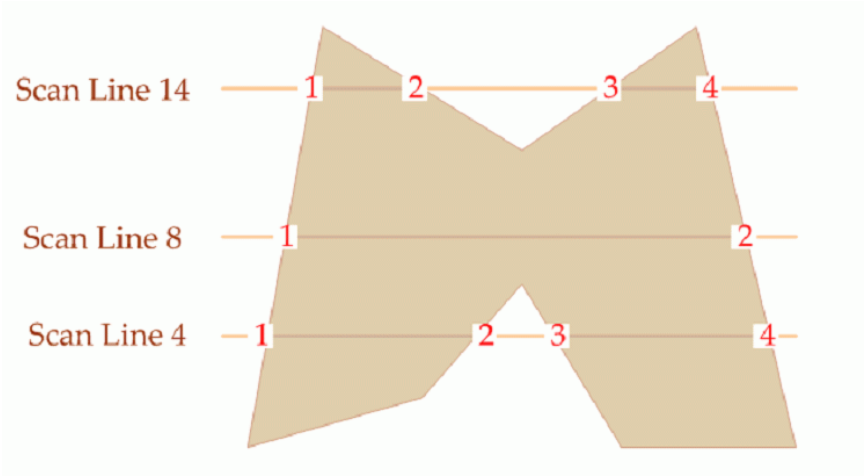
- (5, 0)를 대입하면 결과는 f(x, y) = 4×5 + 8×0 = 20 > 0으로서 양수. 따라서 선분의 위쪽
- x = 2를 기준으로 할 경우. 상하 판단이 어려움

삼각형의 래스터 변환

- 주어진 화소가 삼각형의 내부인지를 판단
  - 다각형의 모든 정점을 항상 반 시계 방향으로 정의.
  - 먼저 정의된 정점을 (x1, y1)으로, 나중 정의된 정점을 (x2, y2)로
  - 선분은 반 시계 방향으로 진행할 때 진행방향의
    - 왼쪽에 대해서는 f(x, y) > 0
    - 오른쪽에 대해서는 f(x, y) < 0

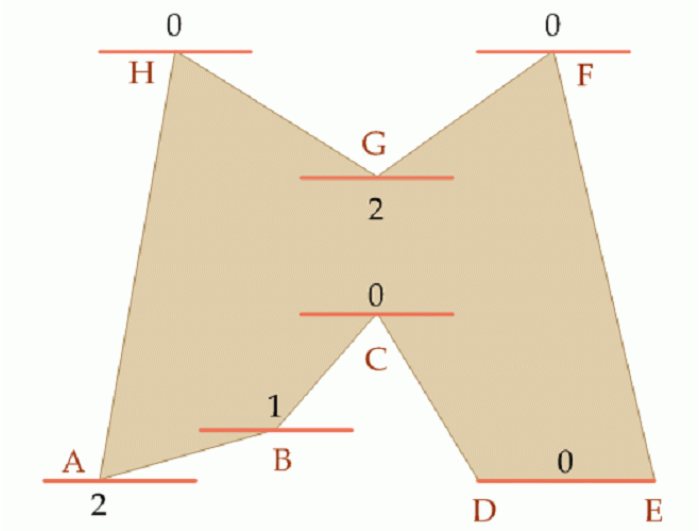


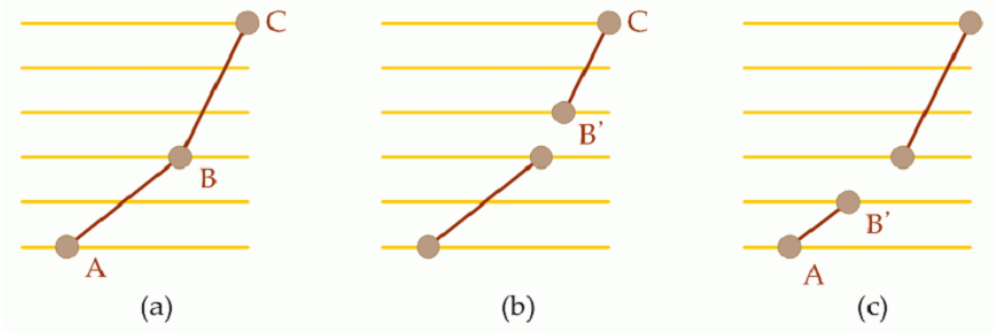
주사선 채움 알고리즘(Scan Line Fill Algorithm)



- 홀수 규칙(Odd Parity Rule, Even-Odd Rule)
  - 홀수번째 교차화소부터 짝수번째 교차화소 직전 직전까지 채움
  - 짝수번째를 포함하지 않는 이유: 길이보존
    - 14번: $1 \le x < 2,\ 3 \le x < 4$
    - 8번: $1 \le x < 2$
    - 4번: $1 \le x < 2,\ 3 \le x < 4$

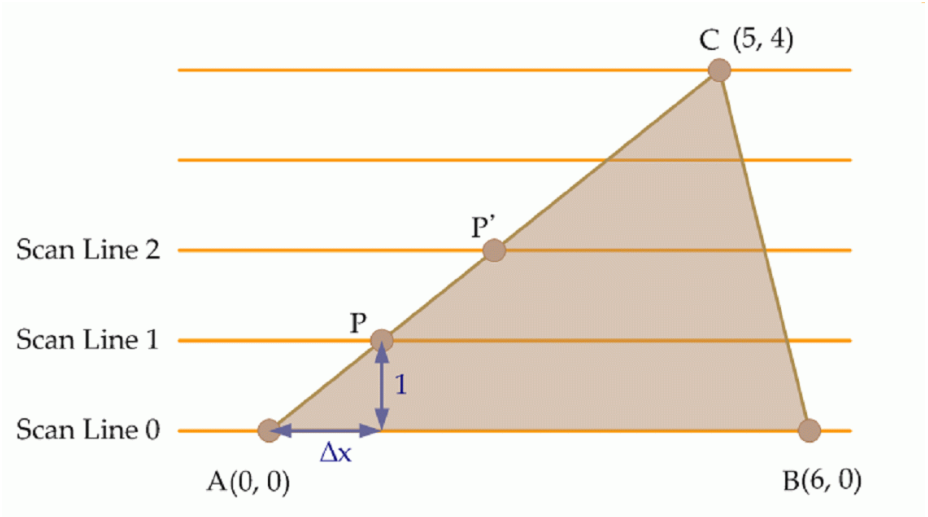
특수 경우 처리





- 길이보존
  - 극대점: 교차하지 않은 것으로 간주(H, F, C)
  - 극소점:각각 교차한 것으로 간주: 2번(G, A)
- 극대극소: 1번 교차(B): 2개의 선분으로 분할
- 주사선과 평행
  - 선분이 없는 것으로 간주(DE)
  - CD, FE에 의해서 처리됨

공간적 응집성(Spatial Coherence)



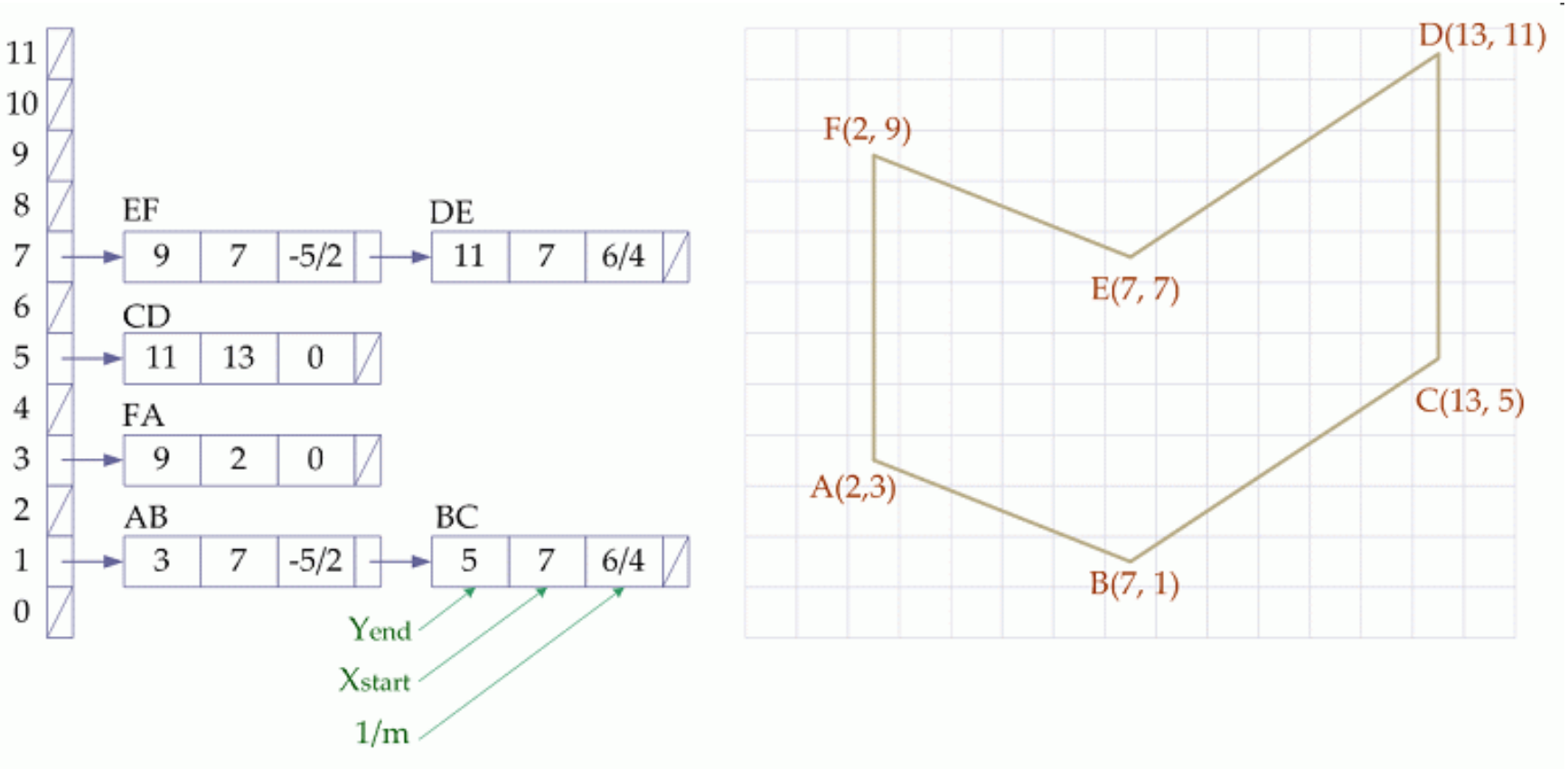
- 내부채움
  - 인접 화소끼리는 같은 색이 칠해질 확률이 높다
- 선분
  - 주사선 0번이 선분 AC와 만났다면 바로 위 주사선 1번도 선분 AC와 만날 확률이 높다.

$$\mathrm{m} = 1/\Delta\mathrm{x}$$

$$\text{if } \textit{Intersection of Scan Line } k = (x_k, y_k)$$

$$\textit{then Intersection of Scan Line } (k+1) = (x_k + 1/m, y_k + 1)$$

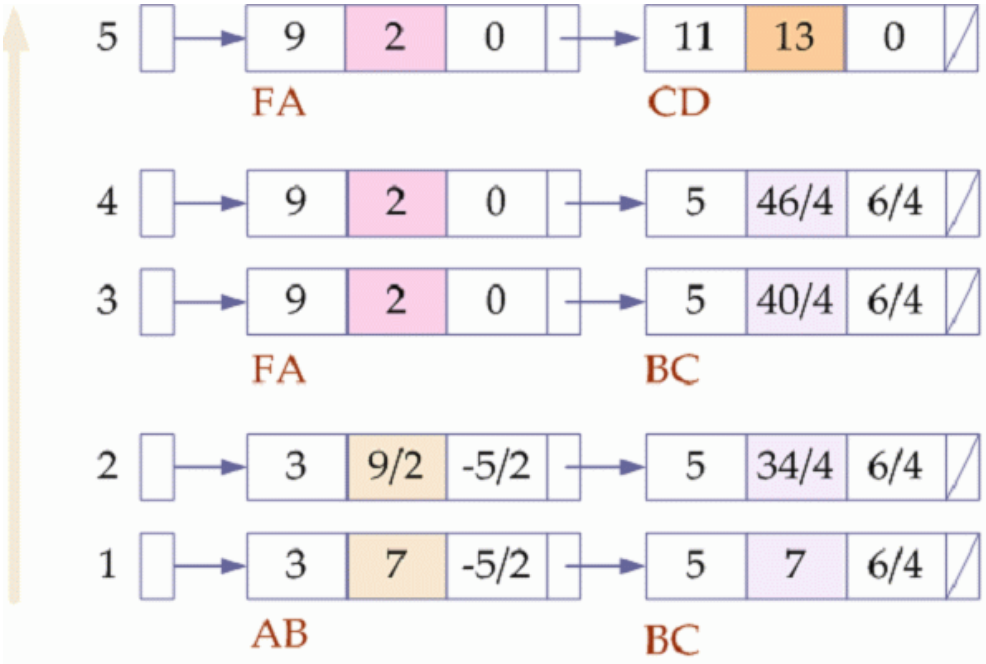
선분 리스트(선분 테이블)

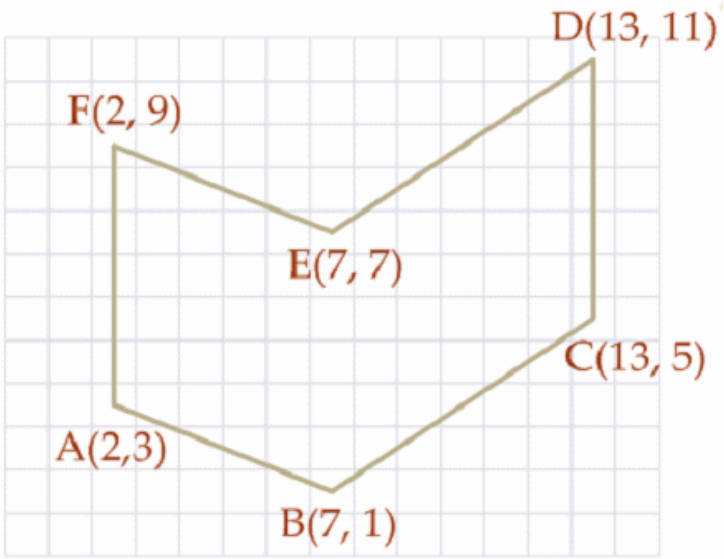


- 선분 위쪽 끝점의 y 좌표(Yend), 아래쪽 시작점의 x 좌표(Xstart), 선분 기울기의 역수(1/m)
- 주사선 1번: (7, 1), 2번: (7+(-5/2), 2) = (9/2, 2), 3번: (9/2+(-5/2), 3) = (4/2, 3) … Yend와 일치할 때까지 계속

활성화 선분 리스트(Active Edge List)

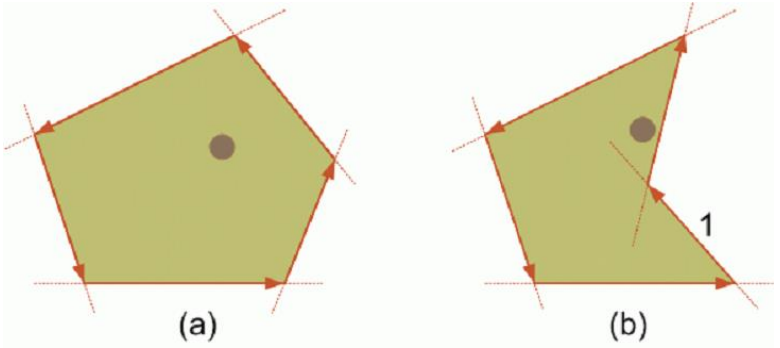
- 주사선 2번으로 증가할 경우 교차점의 x
  - 선분 AB: 7+(-5/2) = 9/2, 선분 BC: 7+(6/4) = 34/4
  - 오름차순으로 정렬 => (9/2, 34/4). 그 사이의 화소가 칠해짐..
- 주사선 3번으로 증가할 경우 현재의 주사선 번호가 Yend(=3)에 도달
  - AB는 비활성화 되어 리스트에서 제거
  - 선분 FA가 활성화 되어 활성화 선분 리스트에 삽입



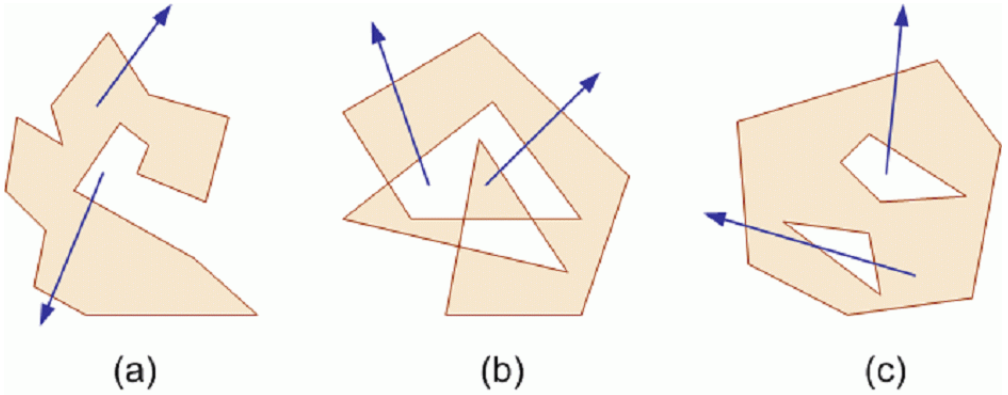


내외부 판정(Inside Outside Test)

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

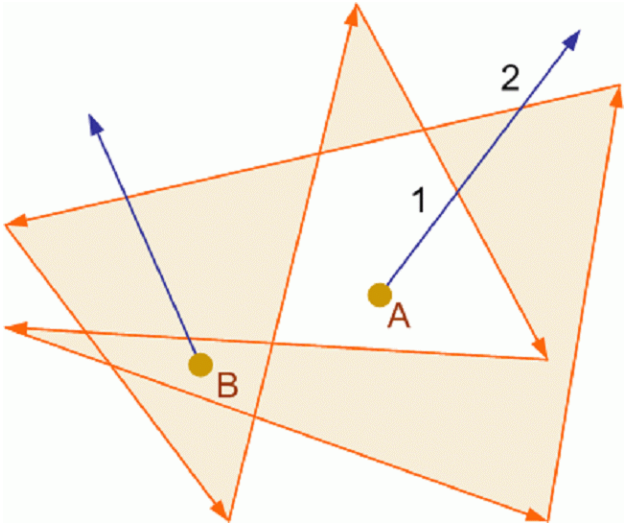


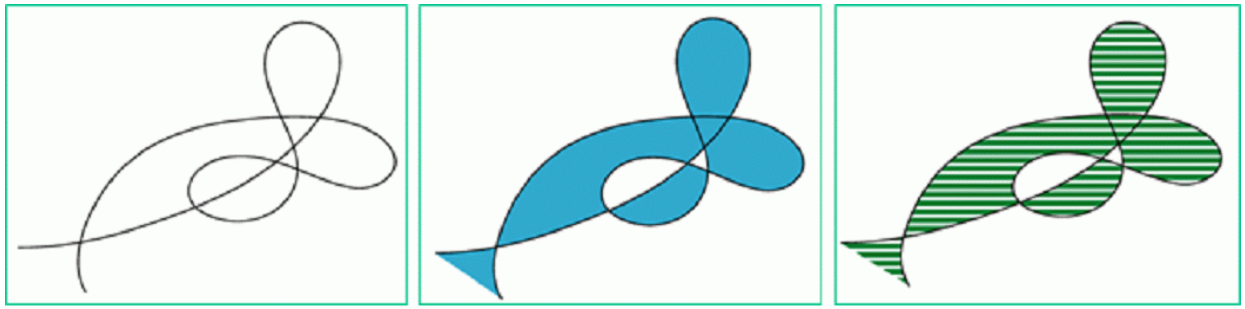
- 홀수 규칙(Odd Parity Rule, Even-Odd Rule)
  - 볼록, 오목에 무관하에 내외부 판정
  - 내부점으로부터 외부를 향한 직선은 다각형과 반드시 홀수 번 교차



내외부 판정(Inside Outside Test)

- 넌 제로 와인딩 규칙(Non-Zero Winding Rule): 선분의 방향을 고려
  - 감싸기 수(Winding Number)
    - 선분이 반 시계방향으로 그 점을 몇 번이나 감싸는가. 0으로 초기화. 선분의 오른쪽에서 왼쪽으로 건너가면 +1, 왼쪽에서 오른쪽으로 건너가면 -1. 최종 감싸기 수가 0이 아니면 내부점으로 간주

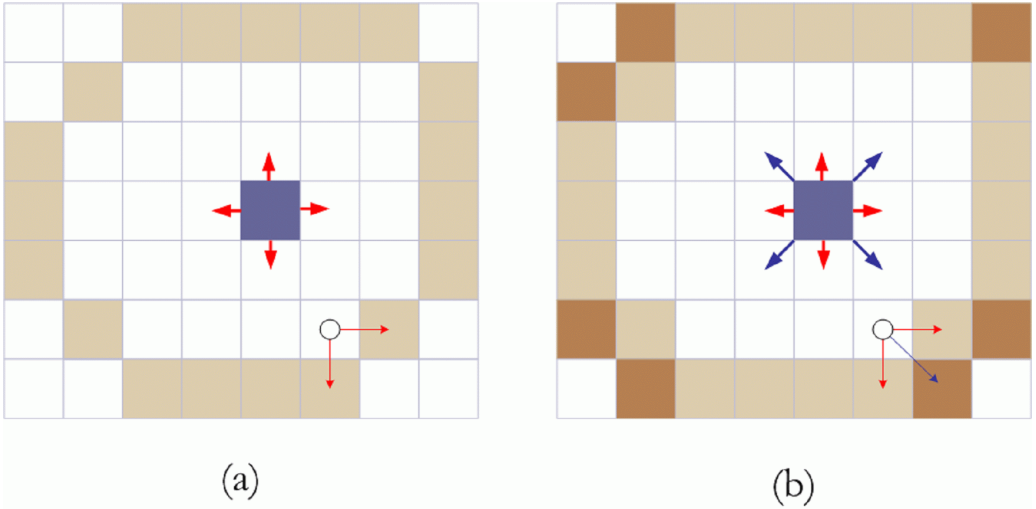




씨앗채움 알고리즘(Seed Fill Algorithm)

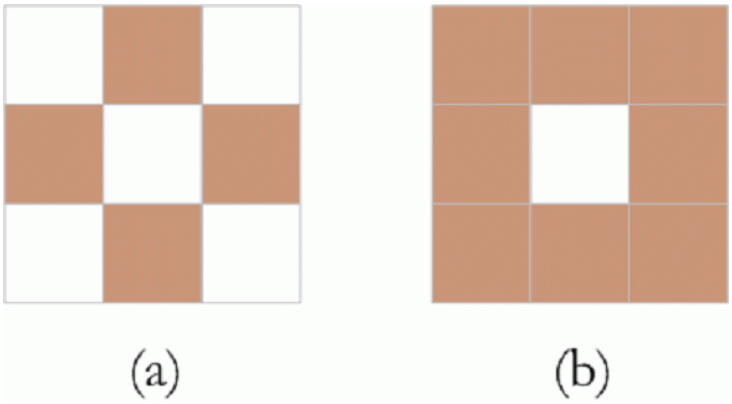
- 어떤 화소가 다각형 내부임이 확인
  - 이를 씨앗으로 해당 화소의 색을 인근으로 번져 나가게 함
  - 경계채움과 홍수채움

- 경계채움 알고리즘(Boundary Fill Algorithm)
  - 경계화소 색을 만날 때까지 4방 또는 8방으로 번짐.
  - 4방향 경계채움, 8방향 경계채움.

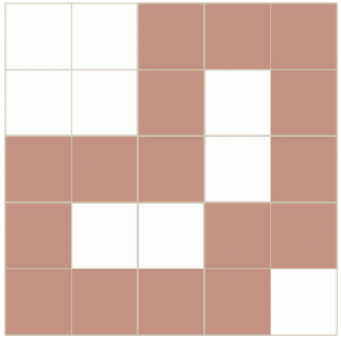


다각형의 연결

- 4방 연결(4-Connectedness), 8방 연결(8-Connectedness)

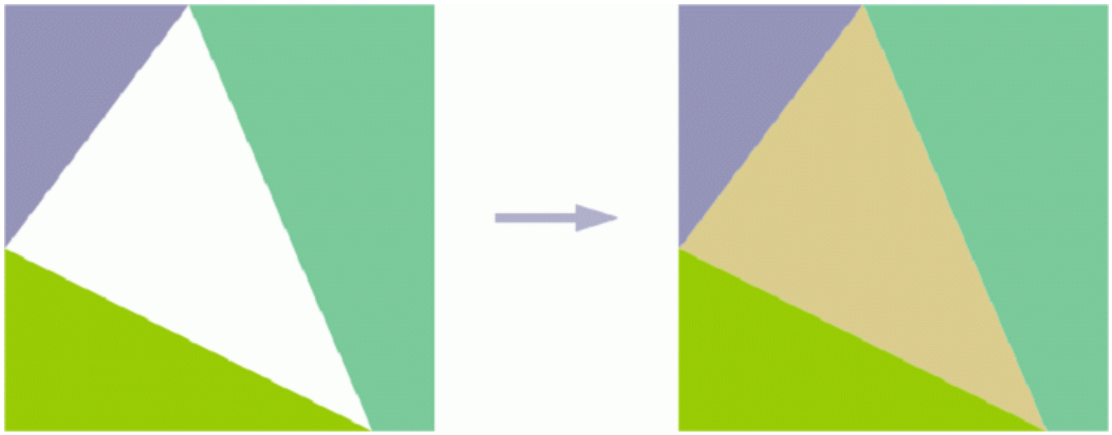


- 4방 연결 시에 8방향 경계채움을 적용하면 오류.
- 8방 연결 시에 4방향 경계채움을 적용하면 완전히 채워지지 않음.



홍수채움 알고리즘(Flood Fill Algorithm)

- 경계채움: 경계화소의 색이 동일
- 홍수채움
  - 경계화소 색이 상이
  - 삼각형 내부의 현재 색은 백색으로 모두 동일
  - 백색을 만날 때까지 4방 또는 8방으로 진행

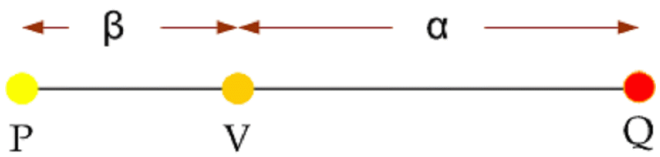


무게중심 좌표(Barycentric Coordinates)

- 선분의 무게중심 좌표(α, β)

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

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

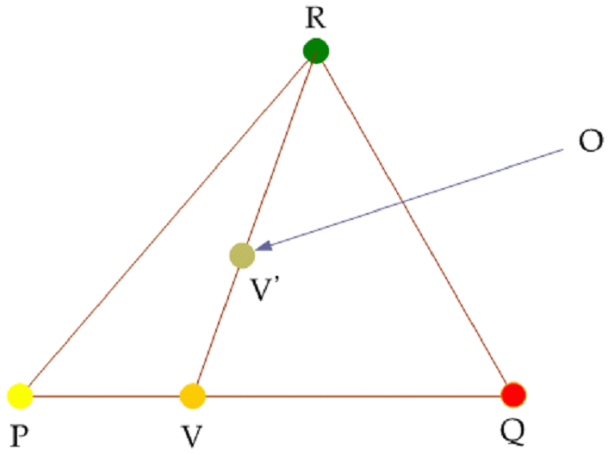


무게중심 좌표(Barycentric Coordinates)

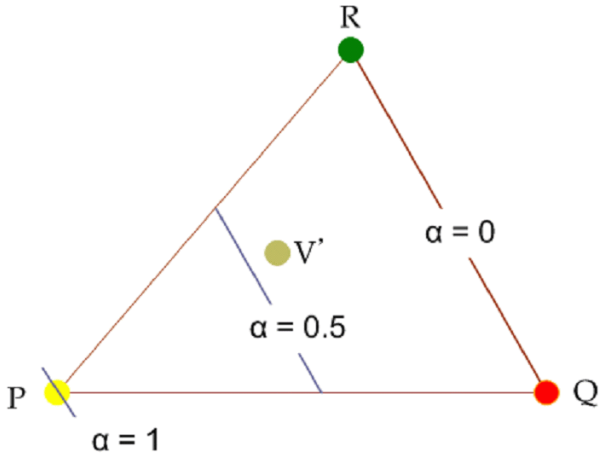
- 삼각형의 무게중심 좌표 (α, β, γ)

$$
\begin{aligned}
V' &= V + s(R - V) \\
&= P + t(Q - P) + s(R - (P + t(Q - P))) \\
&= (1 - t - s + st)P + (t - st)Q + sR \\
&= \alpha P + \beta Q + \gamma R
\end{aligned}
$$

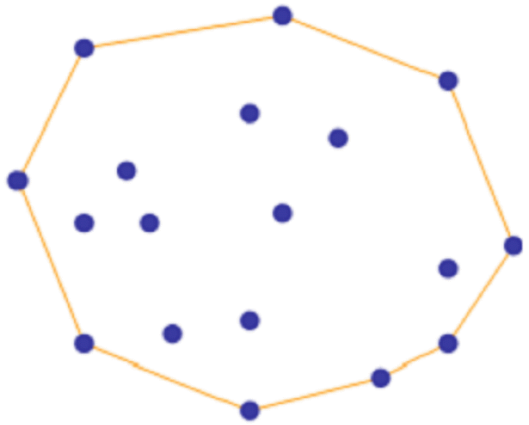
$$0 \leqq \alpha, \beta, \gamma \leqq 1, \ \alpha + \beta + \gamma = 1$$



- α 의 의미



컨벡스 헐(Convex Hull)



$$V = t_1 P_1 + t_2 P_2 + ... + t_n P_n$$

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

- 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

무게중심 좌표 계산

$$\alpha = area(V'QR)/area(PQR)$$

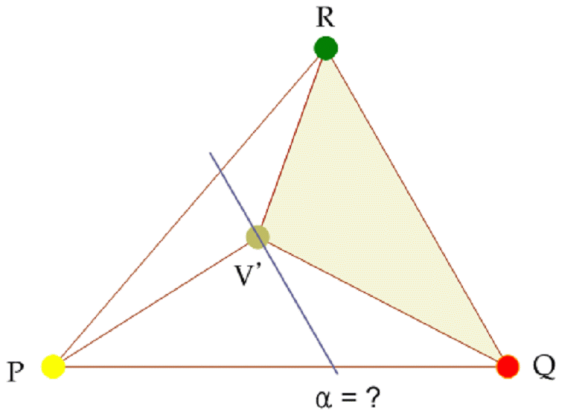
$$\beta = area(V'RP)/area(PQR)$$

$$\gamma = area(V'PQ)/area(PQR)$$

$$\alpha = abs((V'-Q)\times(R-Q))/abs((P-Q)\times(R-Q))$$

$$\beta = abs((V'-R)\times(P-R))/abs((P-Q)\times(R-Q))$$

$$\gamma = abs((V'-P)\times(Q-P))/abs((P-Q)\times(R-Q))$$



- 2차원 투상

$$area(PQR) = \frac{1}{2}\begin{vmatrix} P_x & Q_x & R_x \\ P_y & Q_y & R_y \\ 1 & 1 & 1 \end{vmatrix}$$

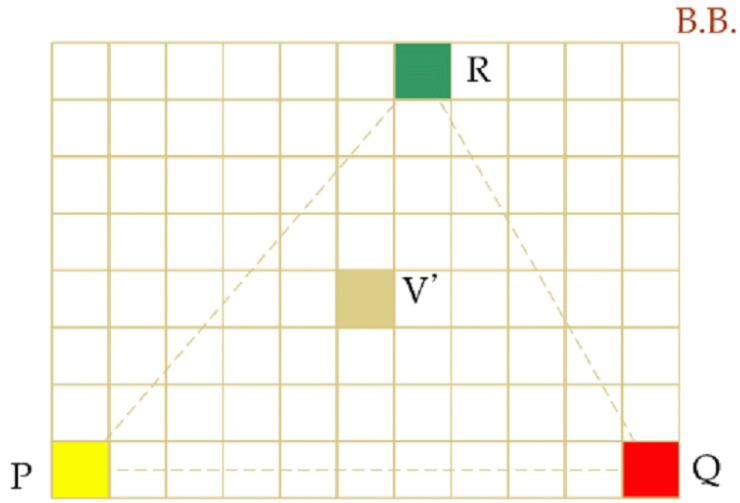
$$= \frac{1}{2}\left(\begin{vmatrix} Q_x & R_x \\ Q_y & R_y \end{vmatrix} + \begin{vmatrix} R_x & P_x \\ R_y & P_y \end{vmatrix} + \begin{vmatrix} P_x & Q_x \\ P_y & Q_y \end{vmatrix}\right)$$

$$= \frac{1}{2}(Q_xR_y - R_xQ_y + R_xP_y - P_xR_y + P_xQ_y - Q_xP_y)$$

$$= \frac{1}{2}((Q_x-P_x)(R_y-P_y) - (R_x-P_x)(Q_y-P_y))$$

무게중심 좌표에 의한 보간

- 경계상자(BB: Bounding Box)
  - 다각형을 둘러싼 최소크기 4각형



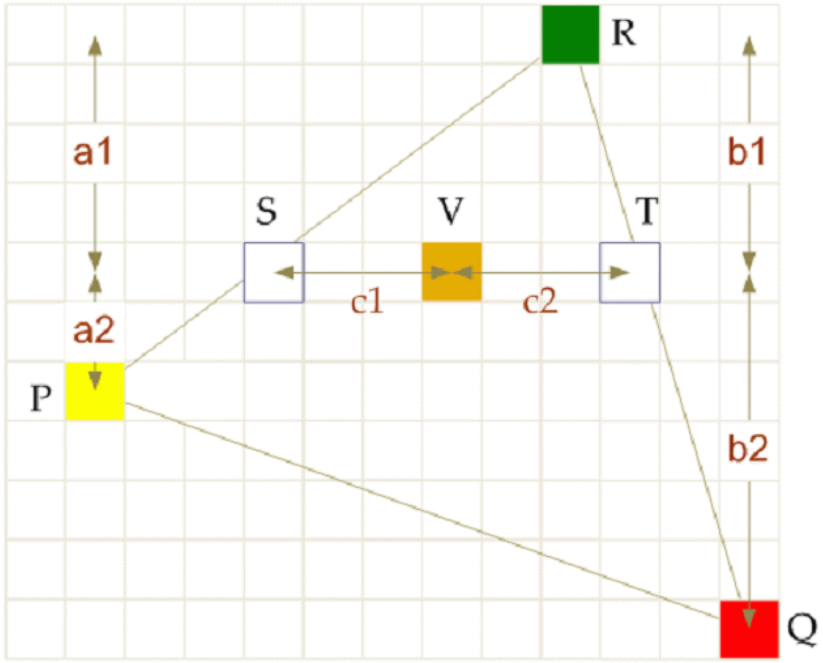
- 보간
  - 경계부피 내의 모든 화소에 대해 무게중심 좌표를 계산
  - 해당 화소가 삼각형 내부인지 판단
  - $r = \alpha P_r + \beta Q_r + \gamma R_r$

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

    $b = \alpha P_b + \beta Q_b + \gamma R_b \qquad z = \alpha P_z + \beta Q_z + \gamma R_z$

양방향 선형보간(Bilinear Interpolation)

- Y 방향 보간에 의해 S, T를 구함
- X 방향 보간에 의해 V를 구함
- 무게중심 좌표와 일치
- 연산속도는 더 빠름



$$S = \frac{a1}{a1+a2}P + \frac{a2}{a1+a2}R \qquad T = \frac{b1}{b1+b2}Q + \frac{b2}{b1+b2}R \qquad V = \frac{c2}{c1+c2}S + \frac{c1}{c1+c2}T$$

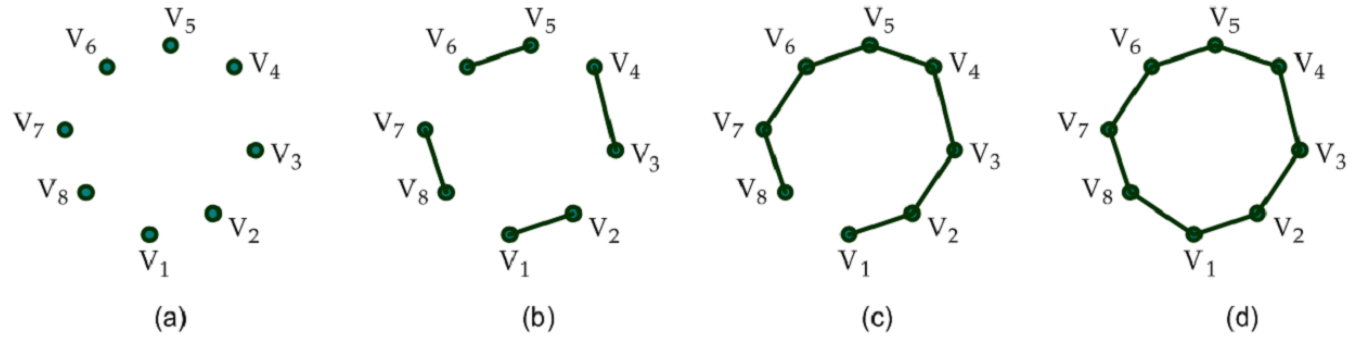
$$V = \frac{c2}{c1+c2}S + \frac{c1}{c1+c2}T$$

$$= \frac{c2}{c1+c2}\left(\frac{a1}{a1+a2}P + \frac{a2}{a1+a2}R\right) + \frac{c1}{c1+c2}\left(\frac{b1}{b1+b2}Q + \frac{b2}{b1+b2}R\right)$$

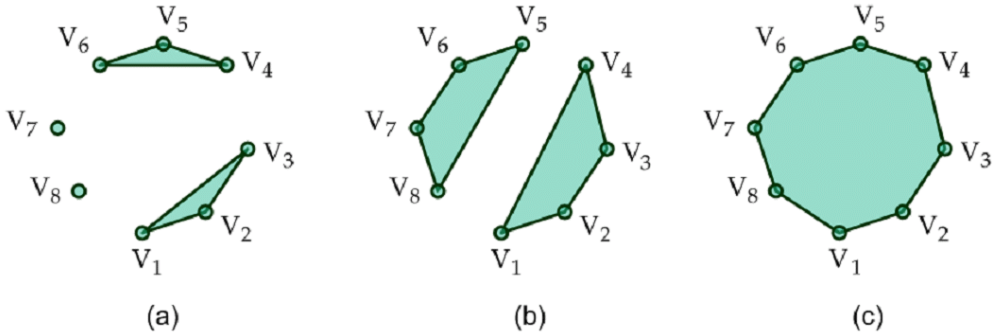
$$= \alpha P + \beta Q + \gamma R$$

지엘의 그래픽 기본요소(Primitives)

- GL_POINTS, GL_LINES, GL_LINE_STRIP, GL_LINE_LOOP

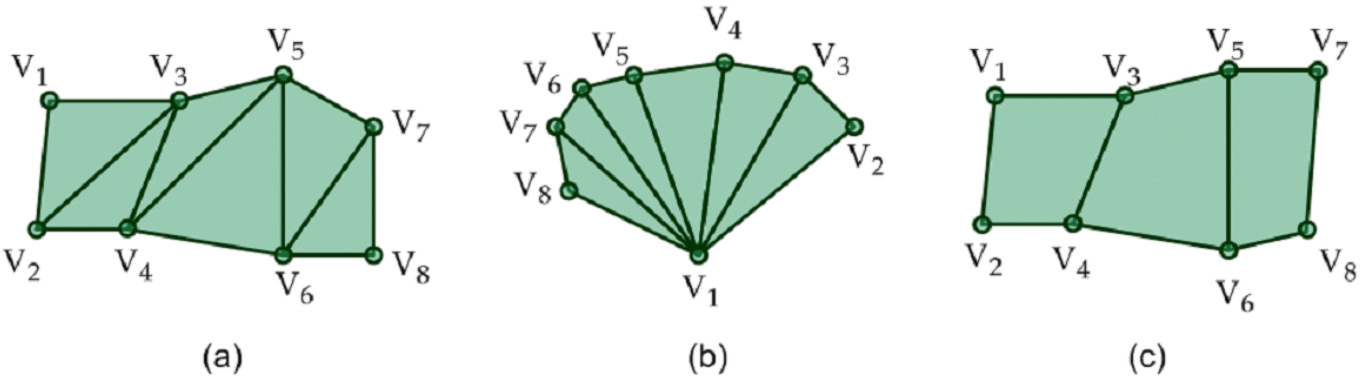


- GL_TRIANGLES, GL_QUADS, GL_POLYGON

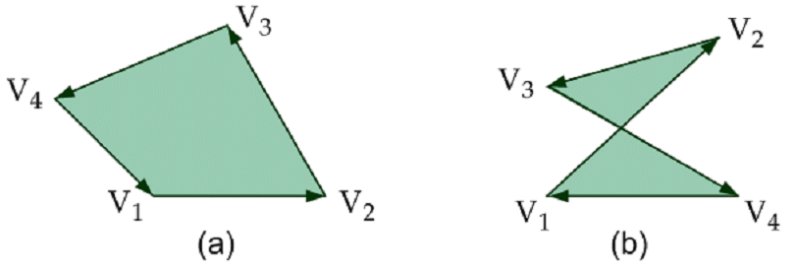


지엘의 그래픽 기본요소(Primitives)

- GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN, GL_QUAD_STRIP

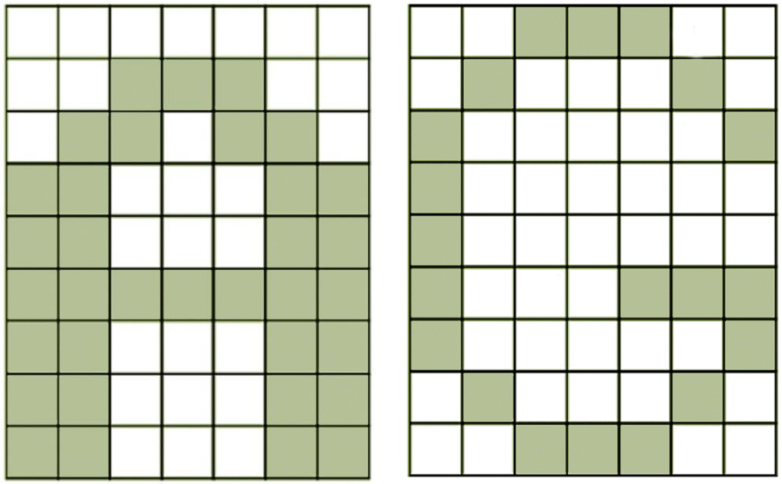


- 제약조건
  - 단순 다각형(Simple Polygon), 볼록 다각형(Convex Polygon), 평면 다각형(Flat Polygon)
  - 단순 다각형



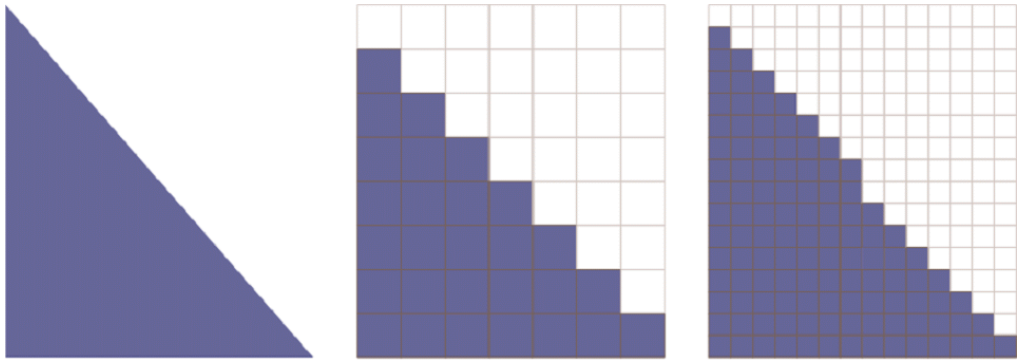
비트맵(Bitmap)

- 비트맵 편집기: Adobe Photoshop
  - cf. 포스트스크립트 편집기: Adobe Illustrator
- 래스터 모니터 영상, 스캐너로 읽은 영상, 팩스에 인쇄된 영상, 페인트 브러시로 만든 영상
- 영상을 구성하는 개별 화소의 색을 표현하고 저장
  - 예: 7X9 = 63 개의 화소배열



에일리어스(Alias)

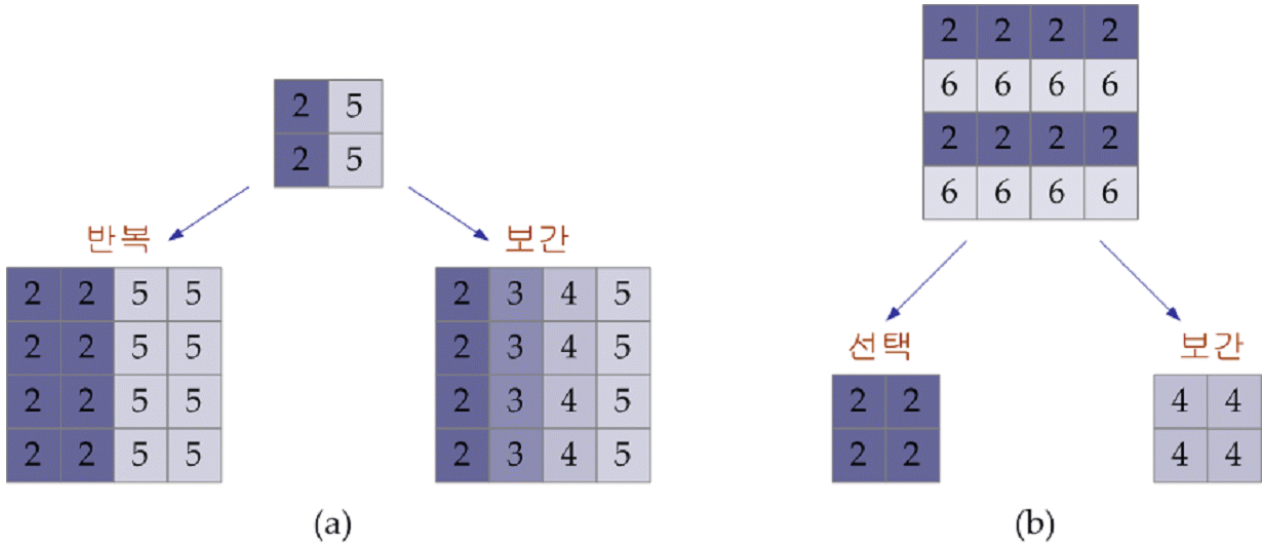
- 계단(Stair-step, Jaggies) 모양의 거친 경계선
  - 비트맵 표현에서는 화소 단위로 근사화 할 수 밖에 없기 때문
  - 무한 해상도를 지닌 물체를 유한 해상도를 지닌 화소 면적 단위로 근사화 할 때 필연적으로 일어나는 현상



비트맵의 확대

- 필연적으로 에일리어싱을 수반
  - 추가의 화소를 채우기 위한 별도 정보가 없음.

- 비트맵 영상의 확대/축소

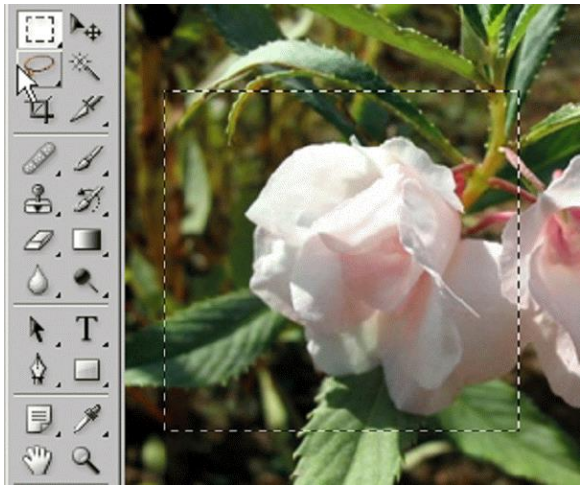


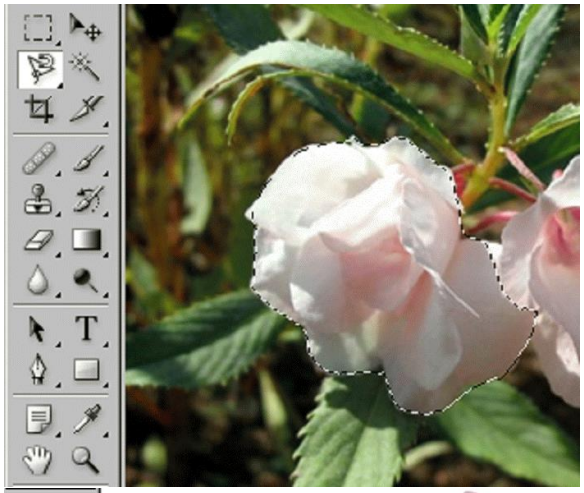
포스트스크립트(Postscript)

- 벡터 그래픽 장비로부터 유래
  - 화소라는 개념이 없음. 무한 해상도
- 실제로는 영상을 그려내는 방식
  - 물체(객체, Object)단위로 물체를 표현
  - 화소 대신 정점좌표를 사용

비트맵과 포스트스크립트

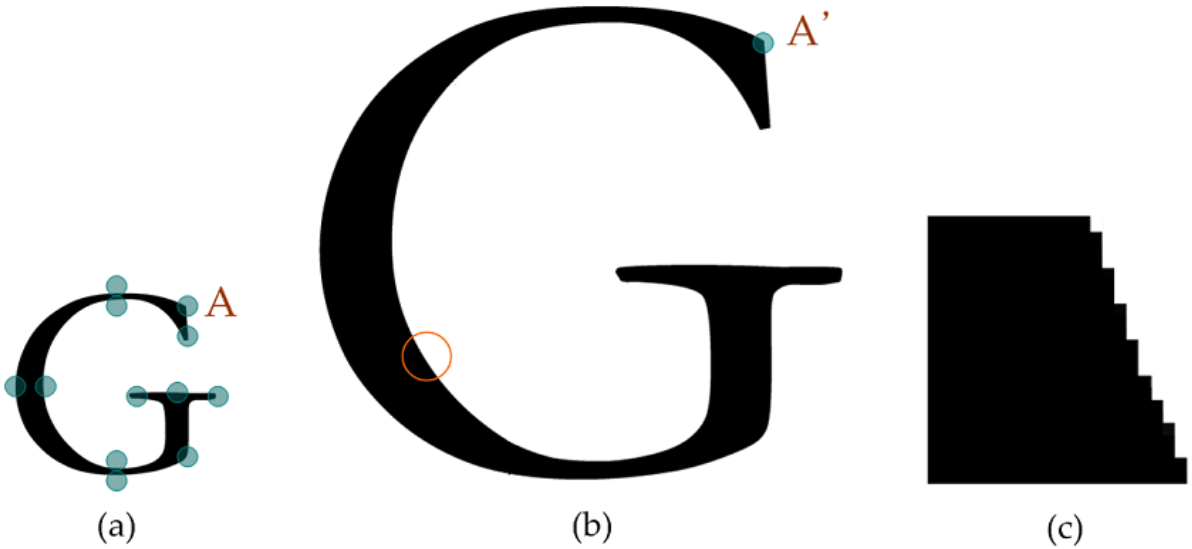
- 비트맵 그리기 = PAINTING, 포스트스크립트 그리기 = DRAWING
- 영상선택
  - 비트맵: 비트 단위, 포스트스크립트: 물체(객체) 단위





포스트 스크립트 글꼴

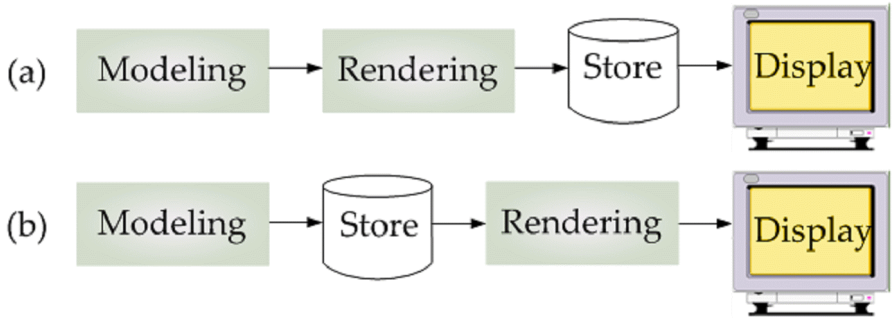
- 영상의 윤곽선을 수식으로 표현
  - 특징적인 정점의 좌표, 이를 연결하는 보간 곡선의 수식을 명시
  - 특징적인 정점 = 제어점(Control Point)
  - 확대된 정점 위치에 보간 곡선을 다시 적용
    - 비트맵 보다 매끄러운 곡선
    - 에일리어싱 완화



그래픽 파일 형식

- 영상압축
  - 무손실 압축(Lossless Compression),손실압축(Lossy Compression)
- BMP(BitMapped Picture)
  - 마이크로 소프트 윈도우즈 운영체제의 기본 비트맵 파일. 일반적으로 압축을 가하지 않은 파일.
- GIF(Graphic Interchange Format)
  - 무손실 압축을 사용한 비트맵 파일. 8비트 컬러 256 컬러 중 하나를 투명성을 구현하는데 사용
- GIF 89a(Graphic Interchange Format 89a)
  - 애니메이션을 위한 파일 형식으로서 하나의 파일에 일련의 영상을 저장. Moving GIF. 프레임 재생률 제어가능. 256 컬러. 사운드 추가할 수 없음. 단순한 웹 애니메이션
- PNG (Portable Network Graphics)
  - W3C에서 추천 파일형식. 향상된 투명성 제어기능. 무손실
- JPEG(Joint Photographic Expert Group)
  - JPEG은 엄밀한 의미에서 일종의 압축 기법. 파일 형식이 아님. 24비트 컬러를 지원. 손실압축.
- TIFF(Tagged Image File Format)
  - 8비트, 24비트 컬러 지원. JPEG 및 기타 압축방법을 수용

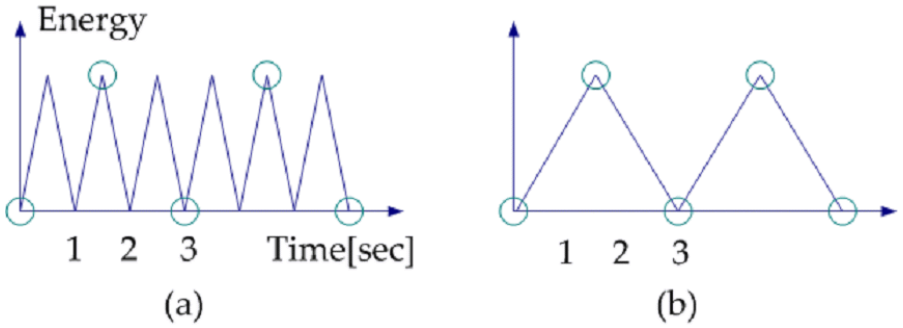
그래픽 파일형식



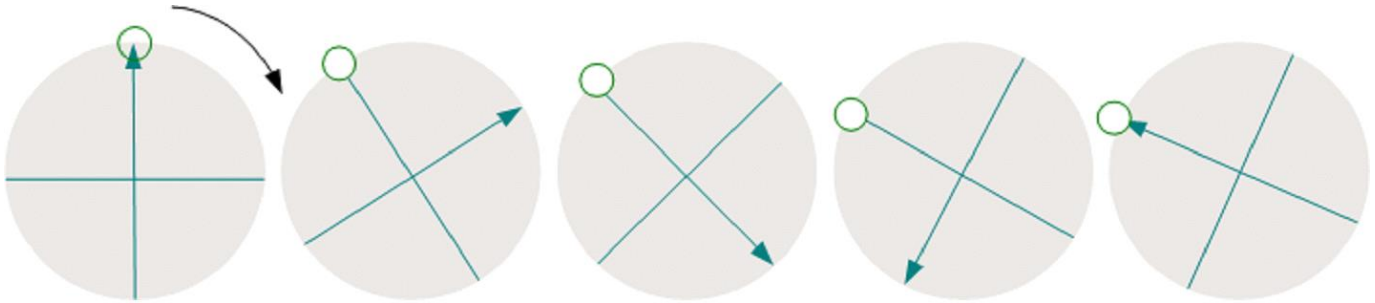
- 메타파일
  - 렌더링 결과 저장: 비트맵 파일
  - 모델링 결과와 렌더링 명령어 저장: 메타파일(예: 포스트스크립트 파일)
  - Ex. PDF(Postscript Description File)
  - 0 1 0 setrgbcolor 현재 색을 녹색으로 설정
  - 0 0 128 128 rectfill 외부 사각형을 채움
  - 1 0 1 setrgbcolor 현재 색을 자홍으로 설정
  - 32 32 64 64 rectfill 내부 사각형을 채움
- EPS(Extended PostScript), SWF(Shockwave Flash), WMF(Windows Meta File), SVG(Scaleable Vector Graphic), PICT(PICTure)
  - 메타파일. 포스트스크립트, 비트맵, 텍스트를 동시에 저장.
  - SWF: 플래시 애니메이션을 위한 파일형식.웹 애니메이션에서 사실 표준, WMF: 마이크로소프트 윈도우즈에서 사용하는 파일
  - SVG: W3C 추천하는 그림파일 형식. XML(Extensible Markup Lang)에서 자주 사용, PICT: 매킨토시에서 사용하는 표준 메타파일 형식.

에일리어싱

- 언더 샘플링으로 인함
  - 신호의 복원
  - 나이퀴스트 주파수

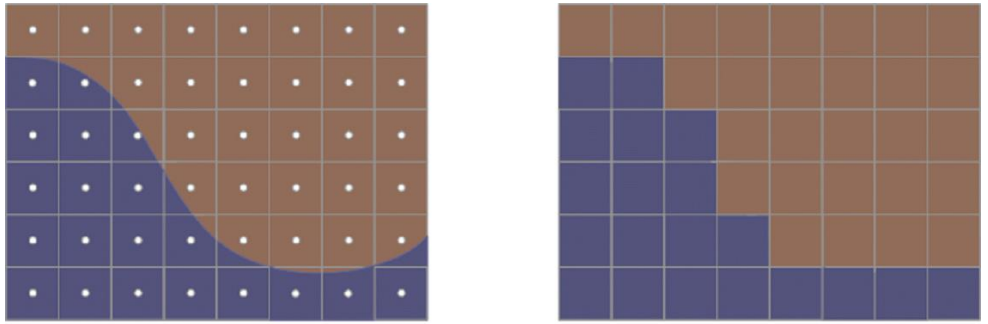


- Stroboscopic Effect
  - 시간적 에일리어싱



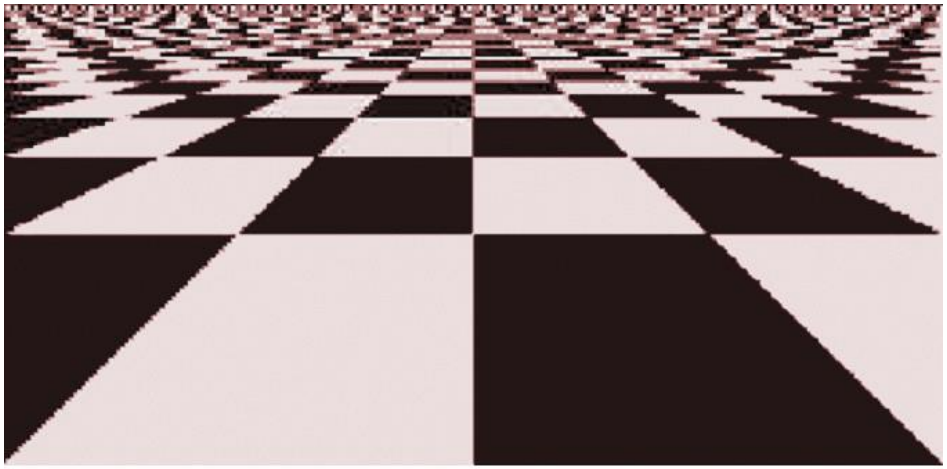
점 샘플링과 모아레 패턴

- 점 샘플링으로 인한 에일리어싱



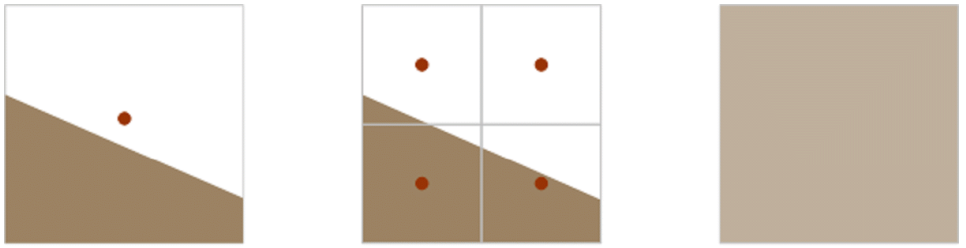
(a) (b)

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

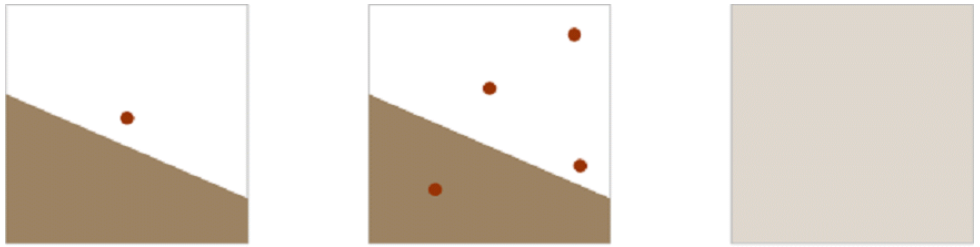


앤티 에일리어싱(Anti-Aliasing)

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

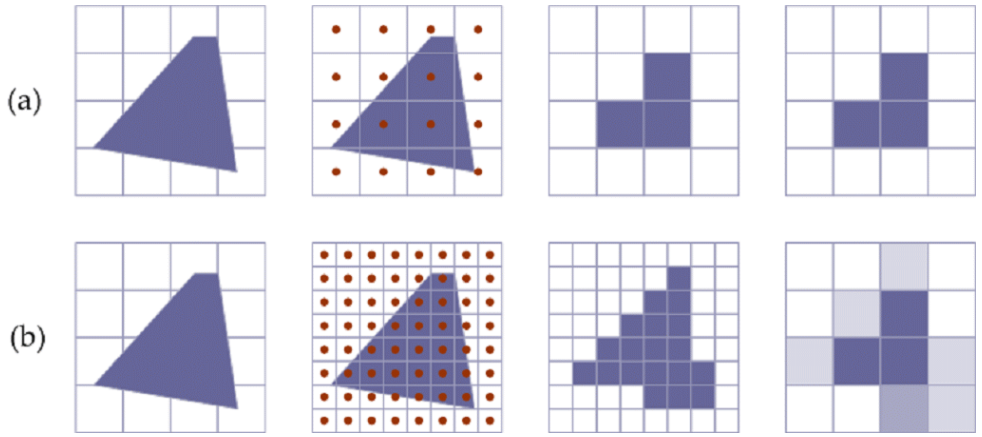


  - 지터에 의한 수퍼 샘플링
    - 물체 자체가 불규칙이라면 불규칙 샘플링이 유리

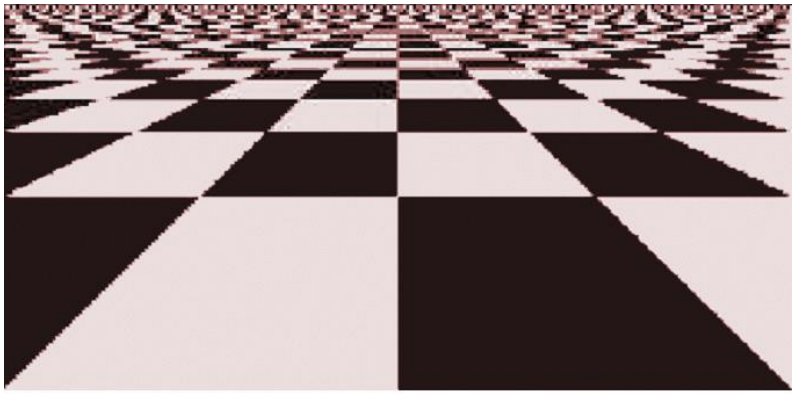


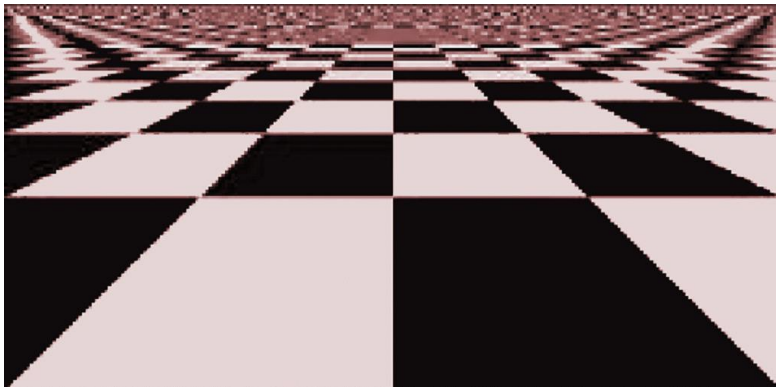
수퍼샘플링

- 포인트 샘플링, 수퍼 샘플링

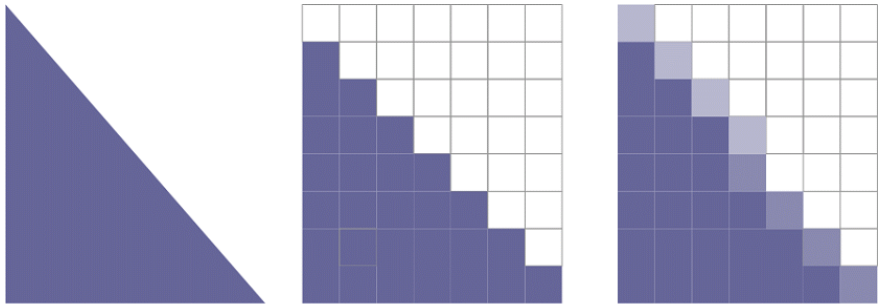


- 수퍼 샘플링

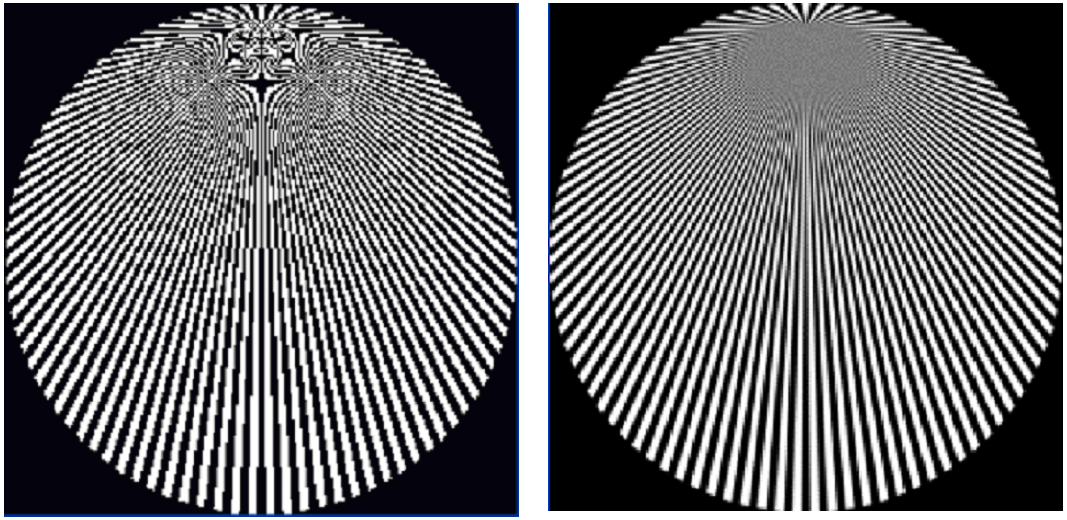




수퍼 샘플링

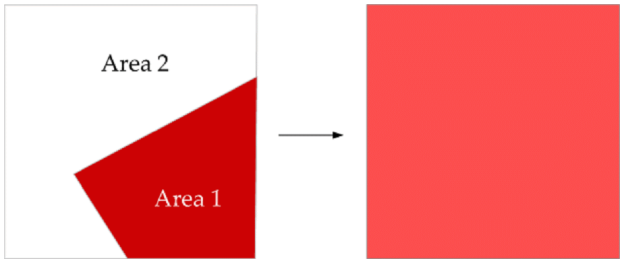


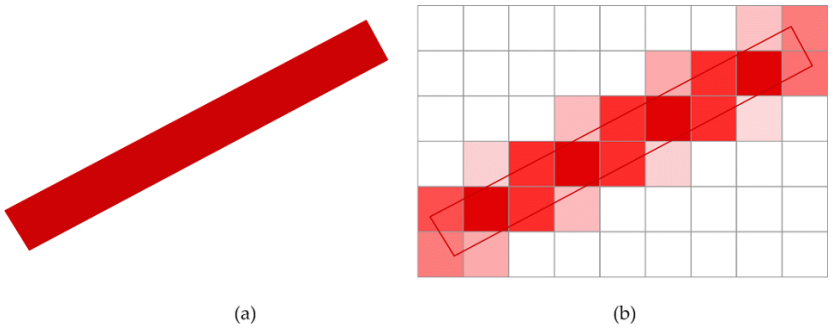
- 포인트 샘플링, 지터링에 의한 수퍼 샘플링



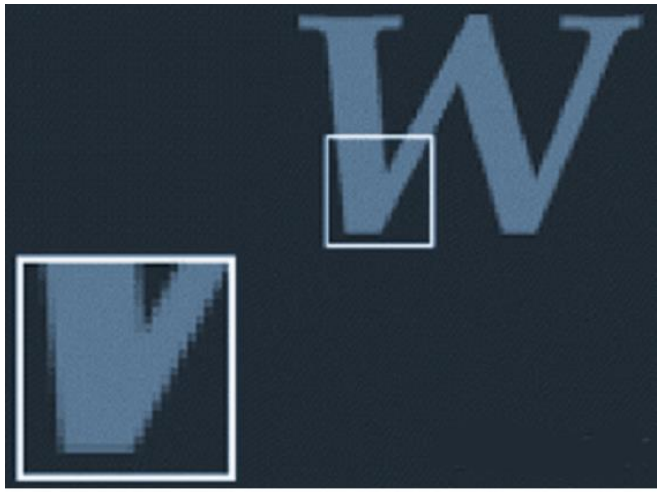
영역 샘플링(Area-Sampling)

- 면적에 비례. 사전 필터링
  - (백색 × Area2 + 적색 × Area1)/(Area1 + Area1)



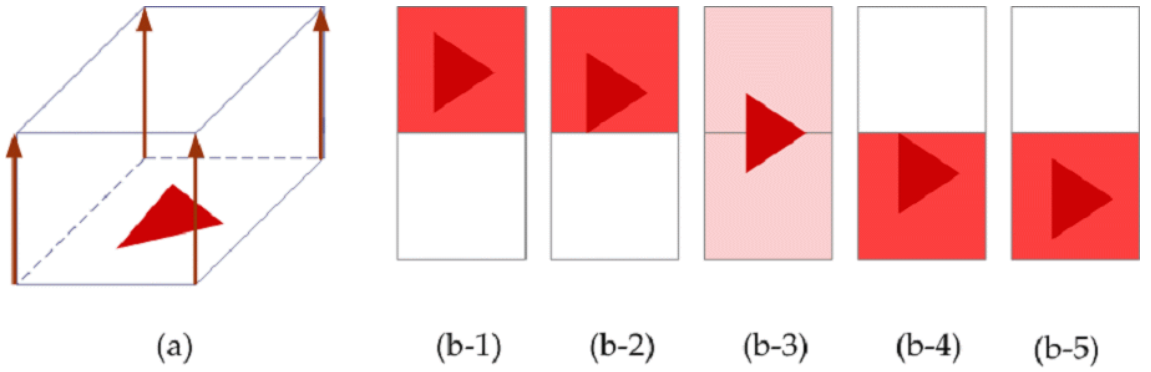


- 포인트 샘플링, 영역 샘플링

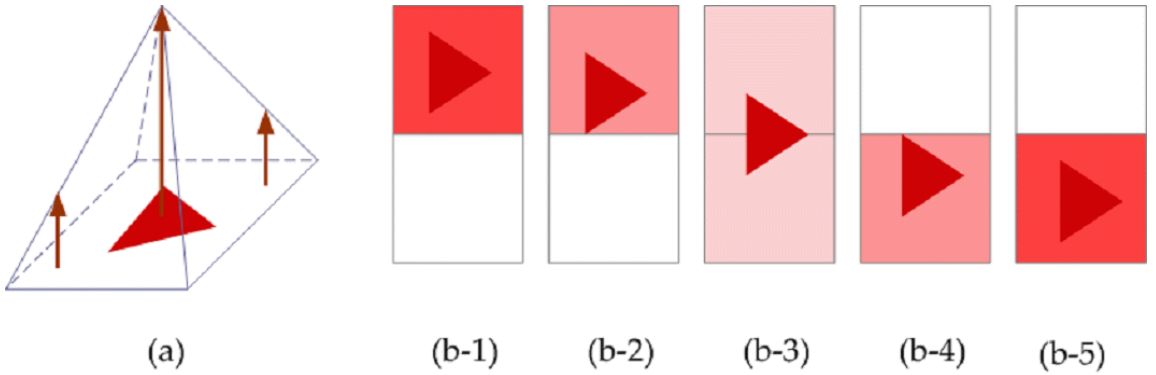


영역 샘플링

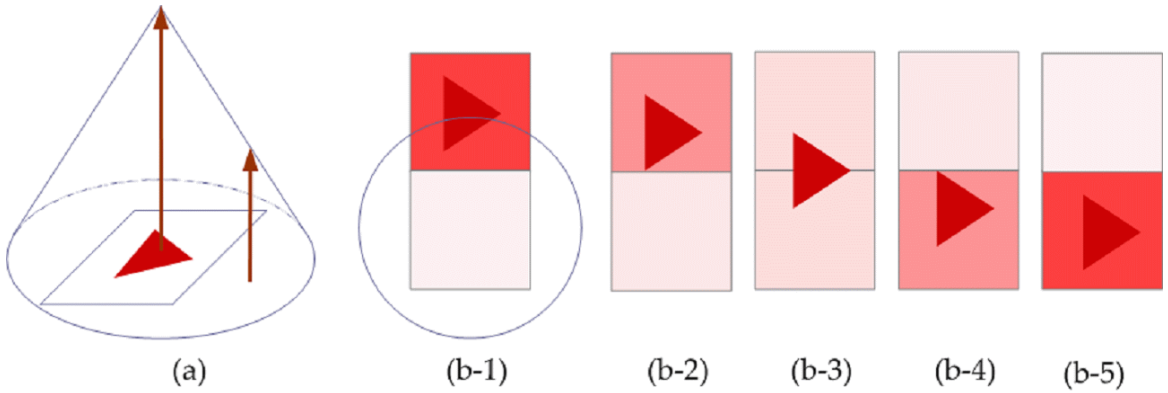
- 동일 가중치



- 피라미드 가중치

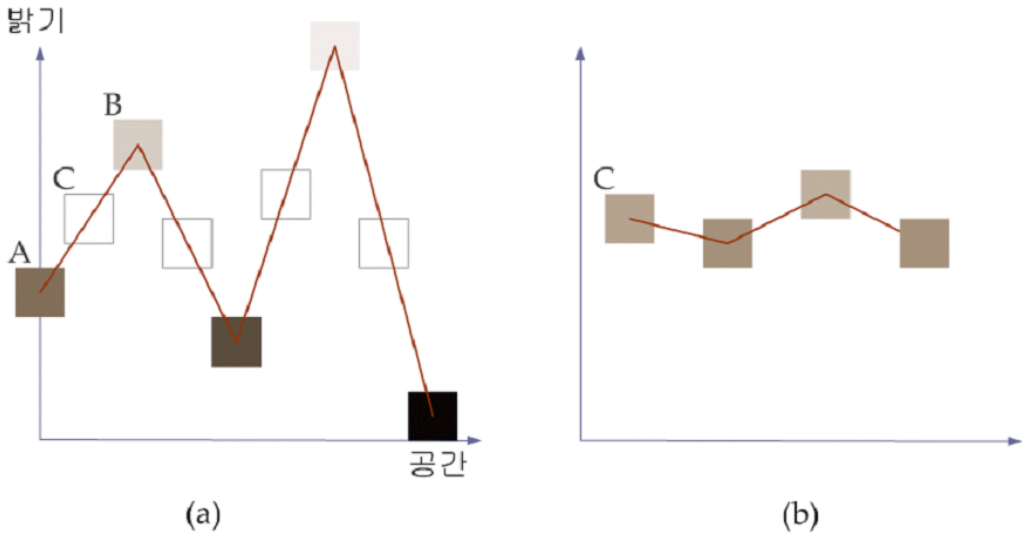


- 원뿔 가중치



영상 필터링

- 화소그룹 처리(Pixel Group Processing)
  - 어떤 화소의 색에 인접화소의 색이 영향을 주는 것.
  - Ex. 저역통과 필터(LPF: Low-Pass Filter) 또는 블러링(Blurring)



컨볼루션 마스크(Convolution Mask)

- Blurring, Sharpening

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

(a)

117	117	27	27
117	117	27	27
117	117	27	27

87	57

(b)

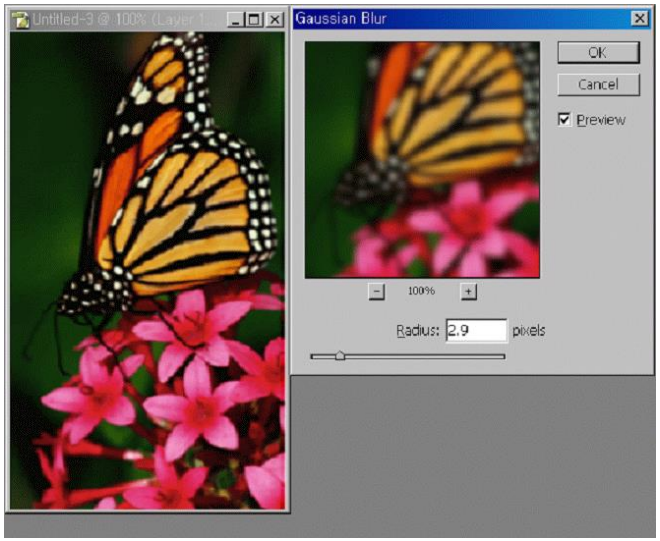
-1	-1	-1
-1	9	-1
-1	-1	-1

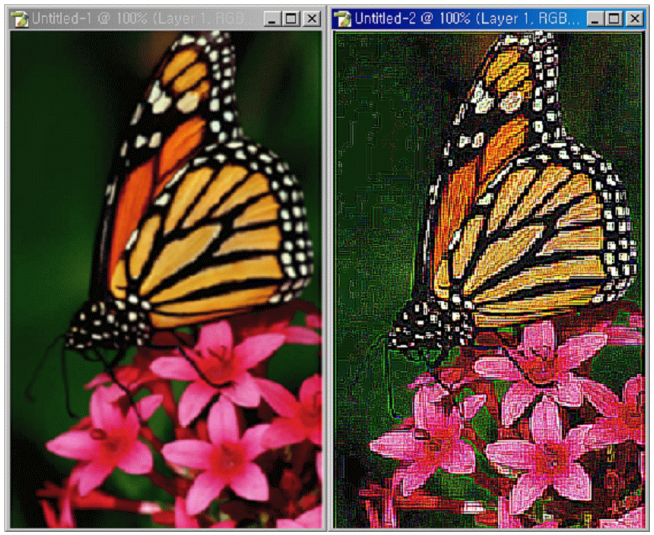
(a)

117	117	51	27
117	117	51	27
117	117	51	27

317	-45

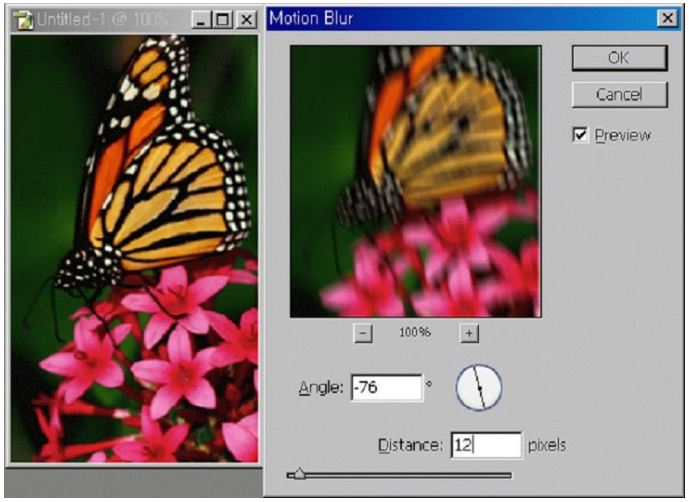
(b)

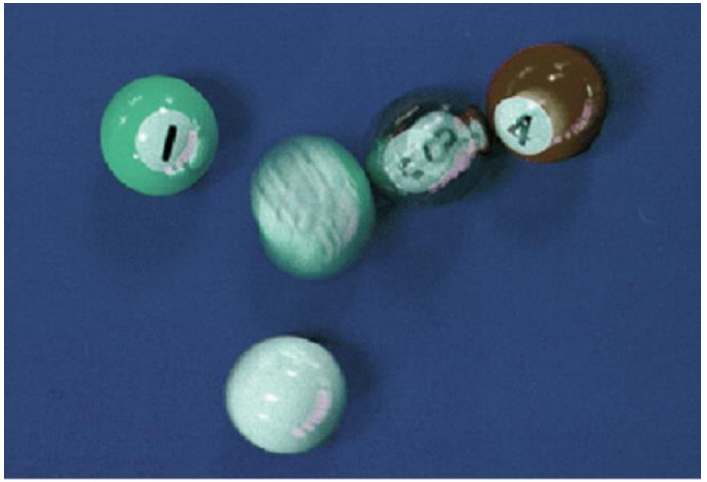




모션 블러(Motion Blur)

- 컨볼루션 마스크가 중앙 화소를 중심으로 방향성을 지님.
- 물체가 움직이는 방향에 있는 화소들에 대해서만 가중치를 적용





블러링에 의한 앤티-에일리어싱

- 수퍼 샘플링에 비해 고속처리
- 수퍼 샘플링은 원래 화면의 해상도 보다 훨씬 많은 샘플링을 요구
- 블러링은 해상도를 그대로 둔 채 인접 화소 정보 만을 이용
- 블러링은 수퍼샘플링에 비해 실질적 해상도 저하