

FDF

“Fils De Fer” which means “wireframe” in French

21.09.04
Yongjule

FDF

About fdf

- fdf형식에 맞게 주어지는 map 파일을 받아 wireframe을 만드는 프로그램
- (x, y, z)좌표를 기준으로 wireframe을 그림
- isometric projection을 이용하여야 함.
- mlx의 image를 쓰는것 강추

FDF

About fdf_bonus

- isometric projection이 아닌 다른 projection도 적용해보기!
- zoom 기능 추가, 평행이동 기능 추가
- 회전 기능 추가

FDF

About FDF

x-axis →

y-axis ↓

	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	10	10	0	0	10	10	0	0	0	10	10	10	10	10	10	0	0	0
0	0	10	10	0	0	10	10	0	0	0	0	0	0	0	10	10	0	0	0
0	0	10	10	0	0	10	10	0	0	0	0	0	0	0	10	10	0	0	0
0	0	10	10	0	0	10	10	0	0	0	0	0	0	0	10	10	0	0	0
0	0	10	10	10	10	10	10	0	0	0	0	10	10	10	10	10	0	0	0
0	0	0	10	10	10	10	10	0	0	0	10	10	0	0	0	0	0	0	0
0	0	0	0	0	10	10	0	0	0	10	10	0	0	0	0	0	0	0	0
0	0	0	0	0	0	10	10	0	0	0	10	10	10	10	10	10	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- The value corresponds to its altitude(z-axis)

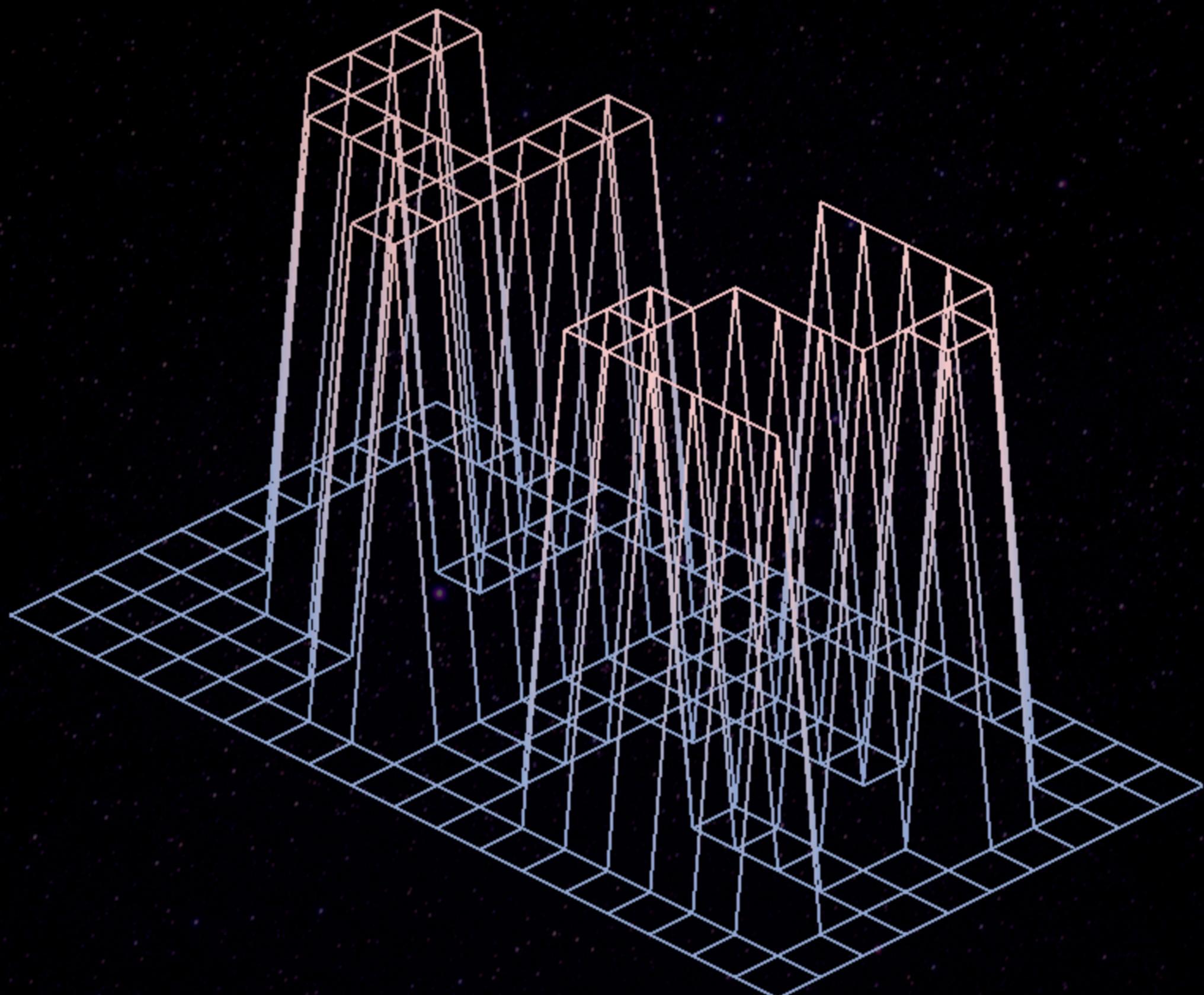
FDF

About FDF

- Each value after ‘,’ means color(hex color)

FDF

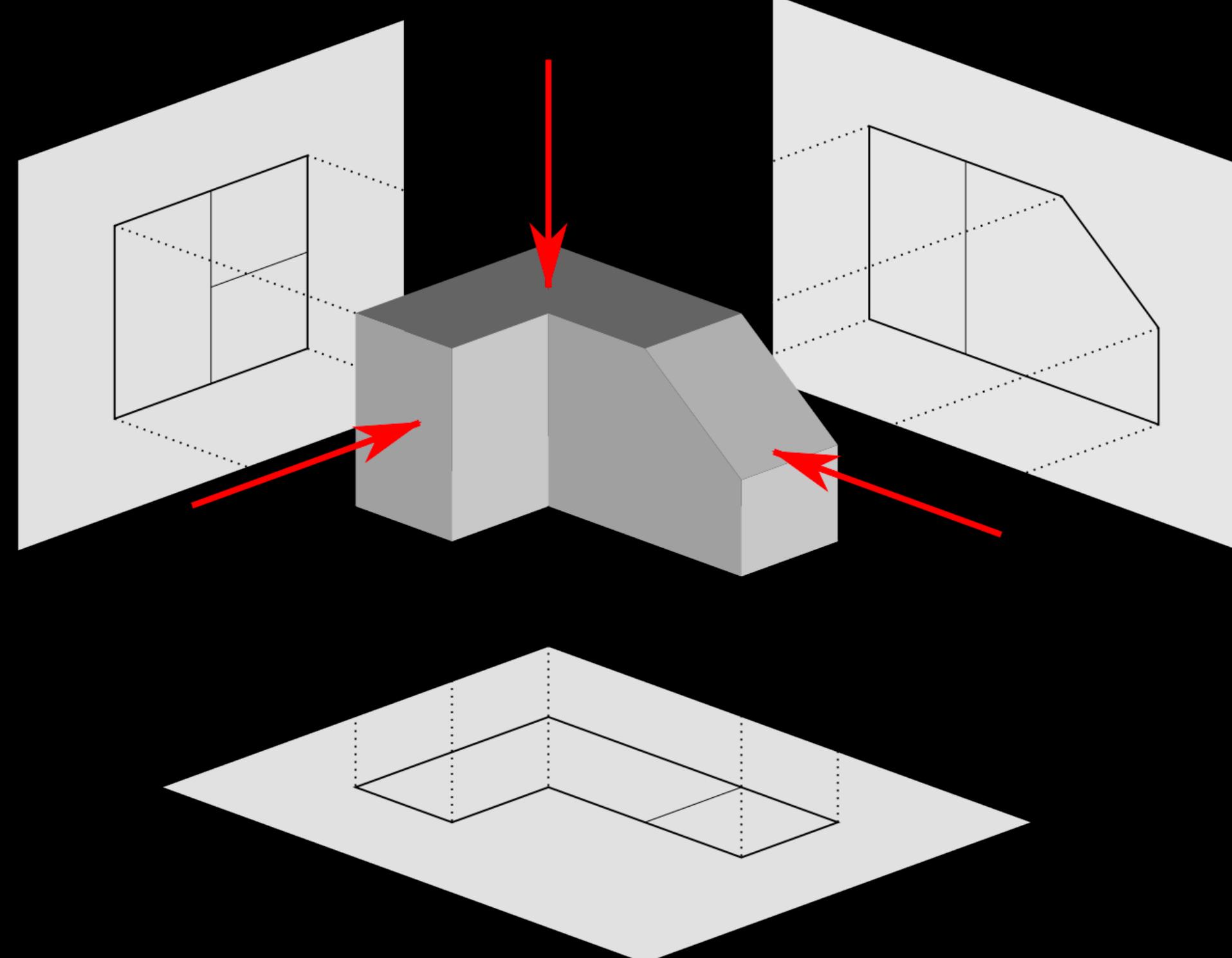
About FDF



Projection

Projection

About projection

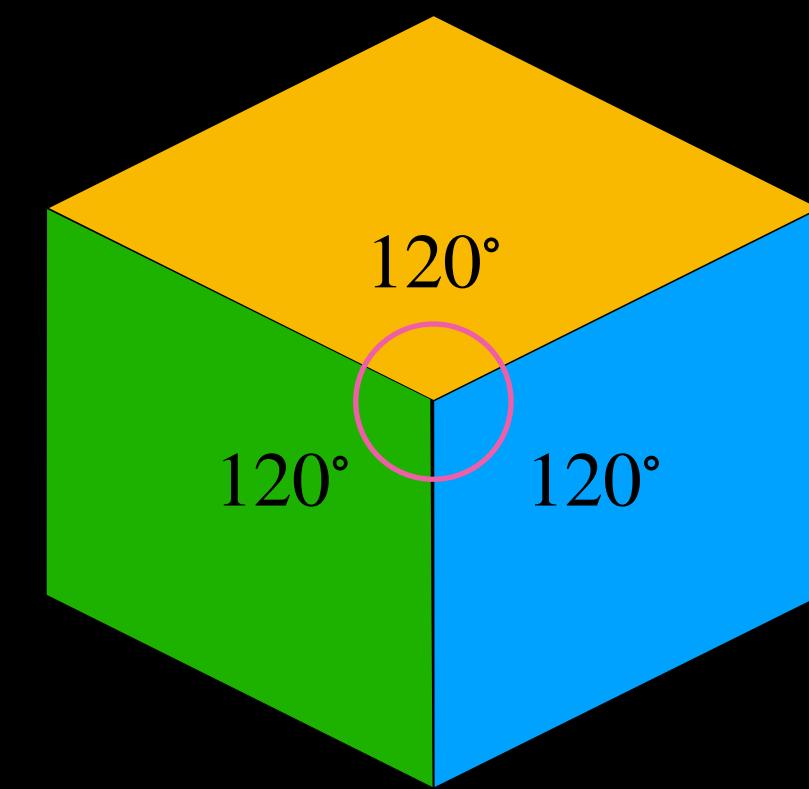
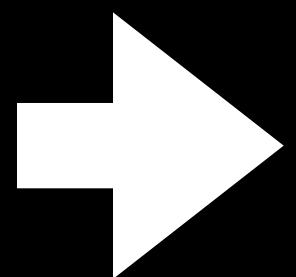
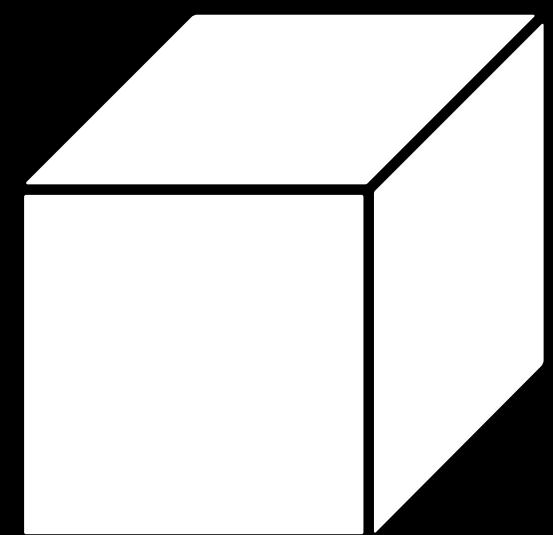
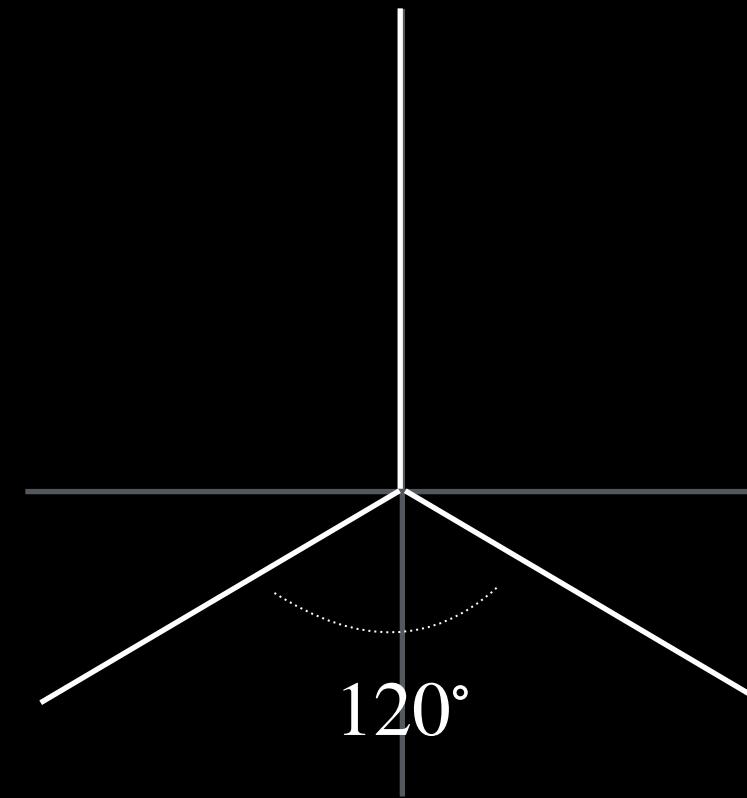
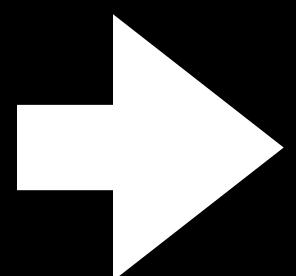


- 3d 물체를 2d 평면에 표현하기 위한 방식 중 하나
- 여러 방식의 projection이 존재하며, fdf에서는 isometric projection을 구현해야함!
- 주로 projection matrix를 이용하여 좌표 변환을 하는 방식

Isometric Projection

Isometric Projection

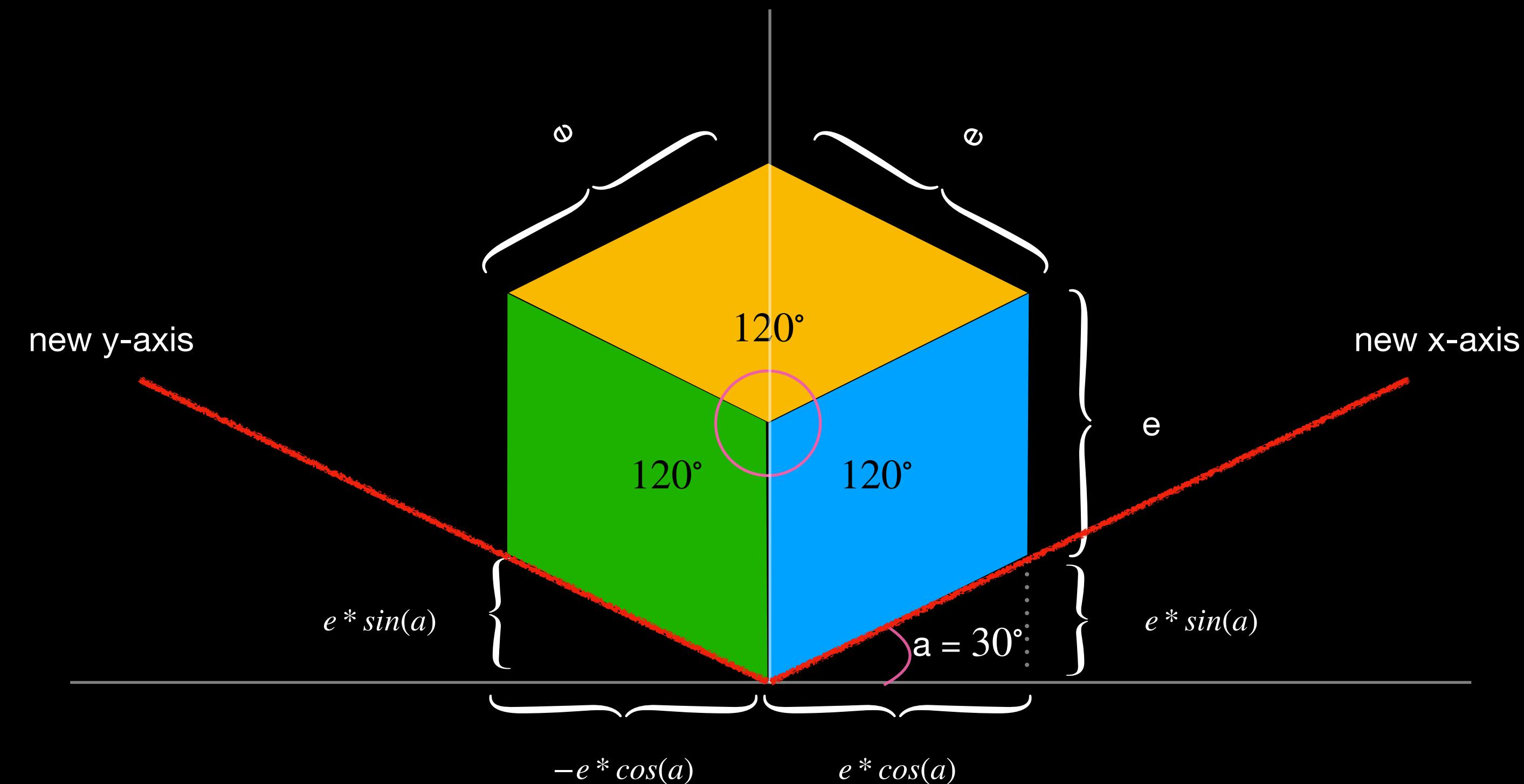
About isometric projection



- x축을 30° , y축을 60° 회전하여 좌표축에 올리는 projection
- z축은 그대로 유지
- 이 isometric projection을 한 좌표가 x-y 평면에서 어디에 찍히는지 계산하면 됨!

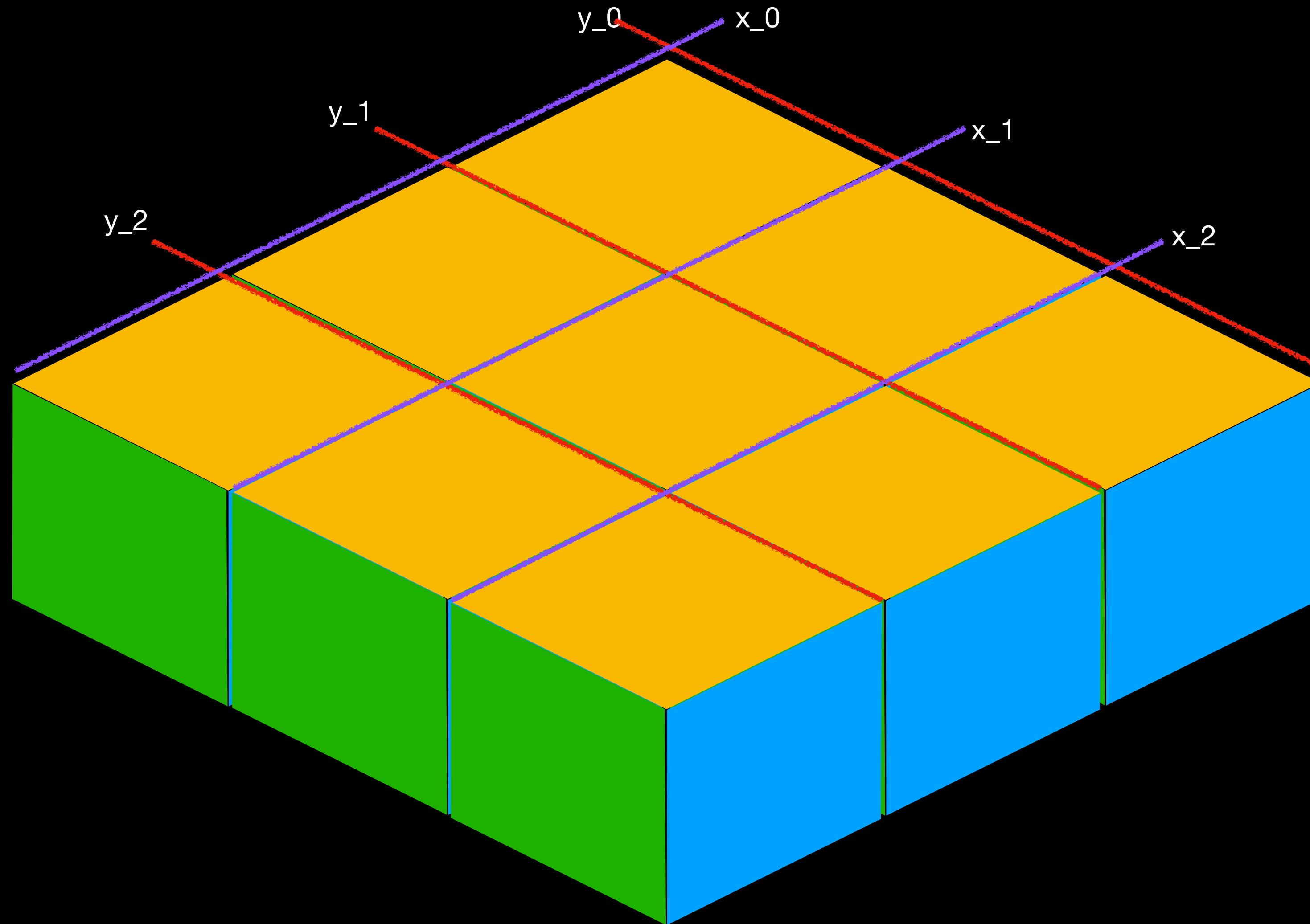
Isometric Projection

Mathematical base of isometric projection



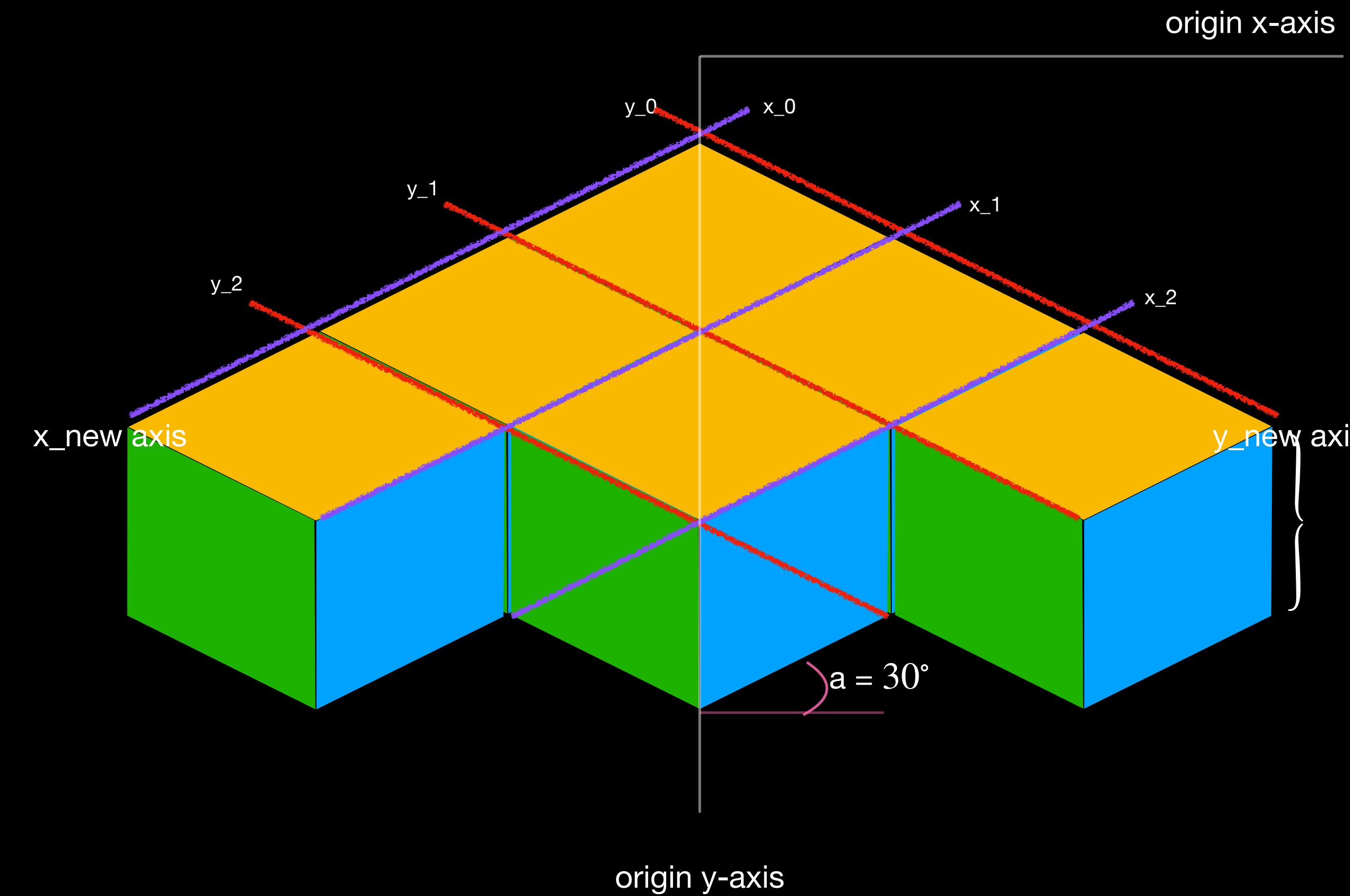
Isometric Projection

Mathematical base of isometric projection



Isometric Projection

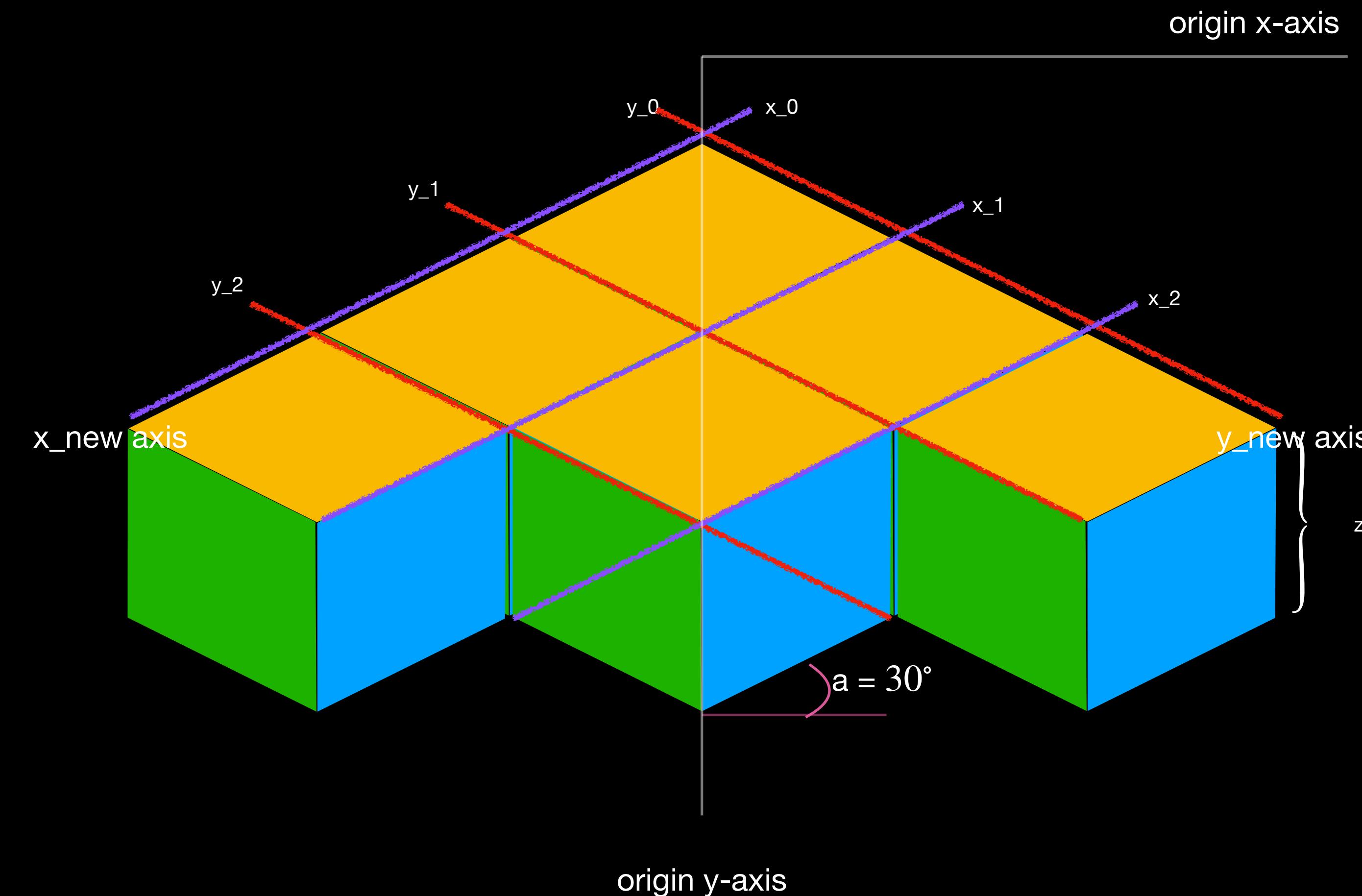
Mathematical base of isometric projection



- increasing $x_{newaxis}$
 - increase $x_{origin} * \cos(a)$
 - increase $y_{origin} * \sin(a)$
- increasing $y_{newaxis}$
 - decrease $x_{origin} * \cos(a)$
 - increase $y_{origin} * \sin(a)$
- z-axis
 - decrease y_{origin} by the true value

Isometric Projection

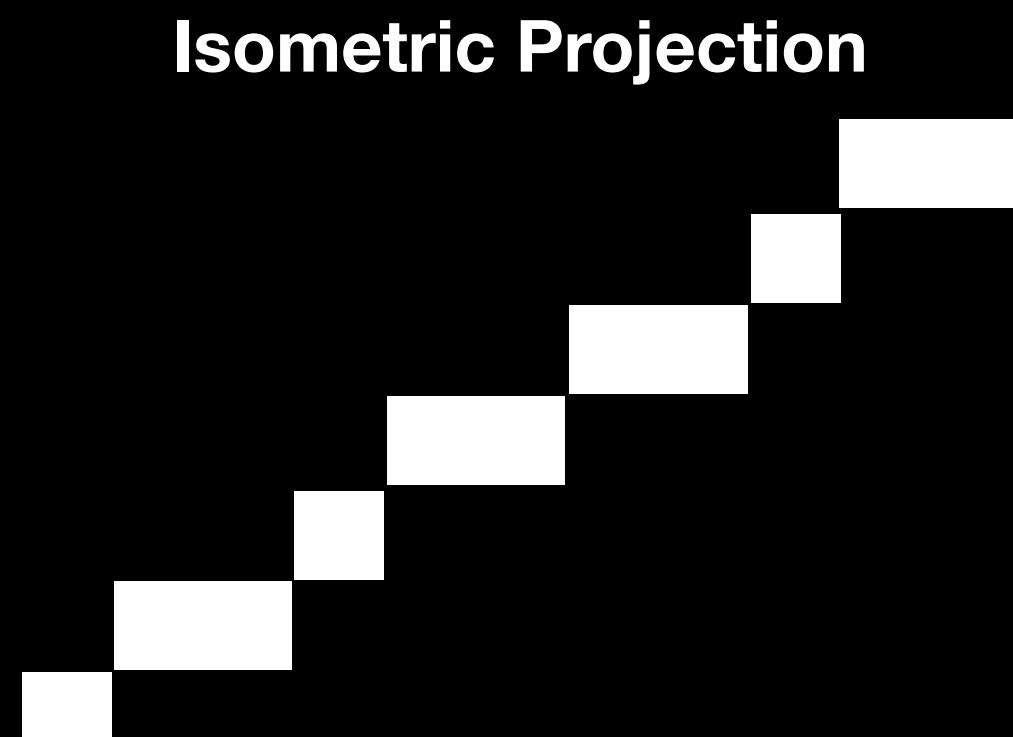
Mathematical base of isometric projection



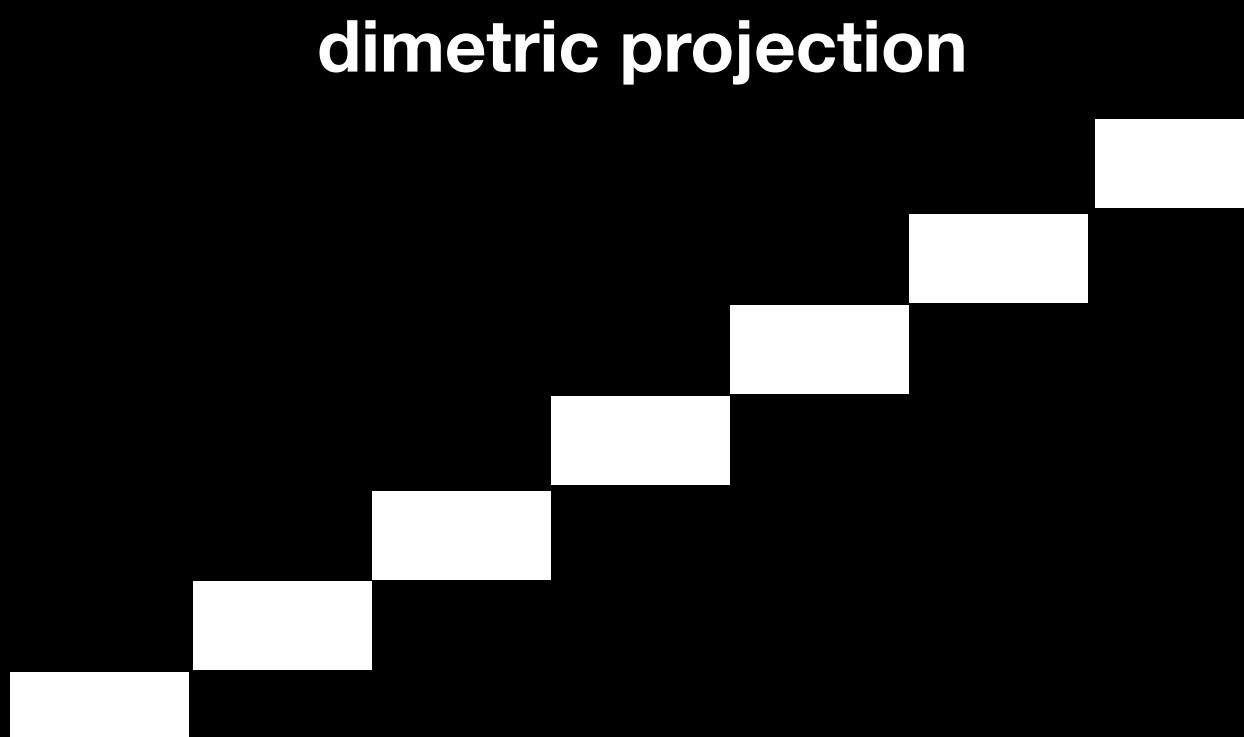
- $x_{proj} = (x - y) * \cos(a)$
- $y_{proj} = (-z + (x + y) * \sin(a))$

Isometric Projection

Isometric Projection in Computer Graphics



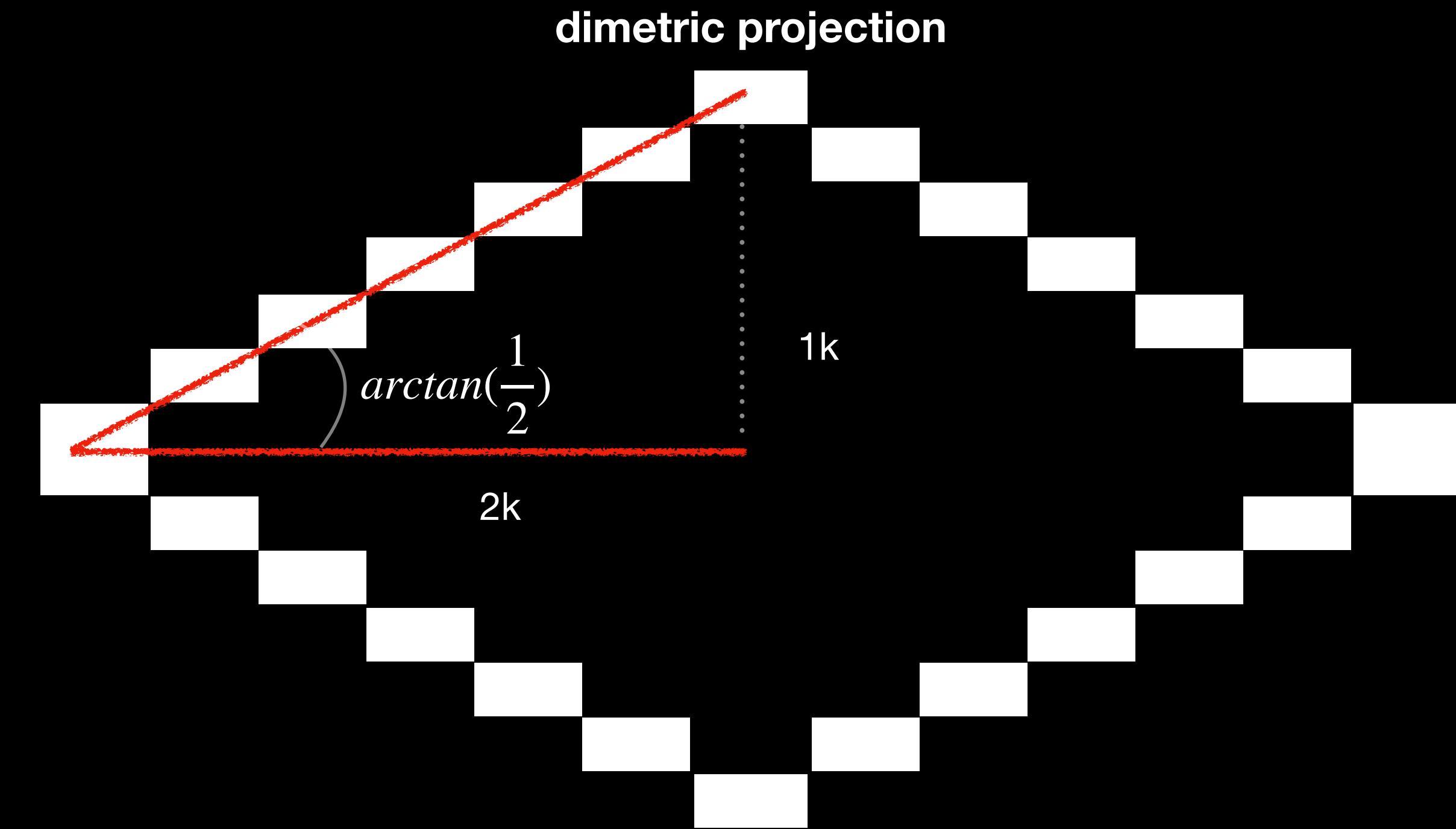
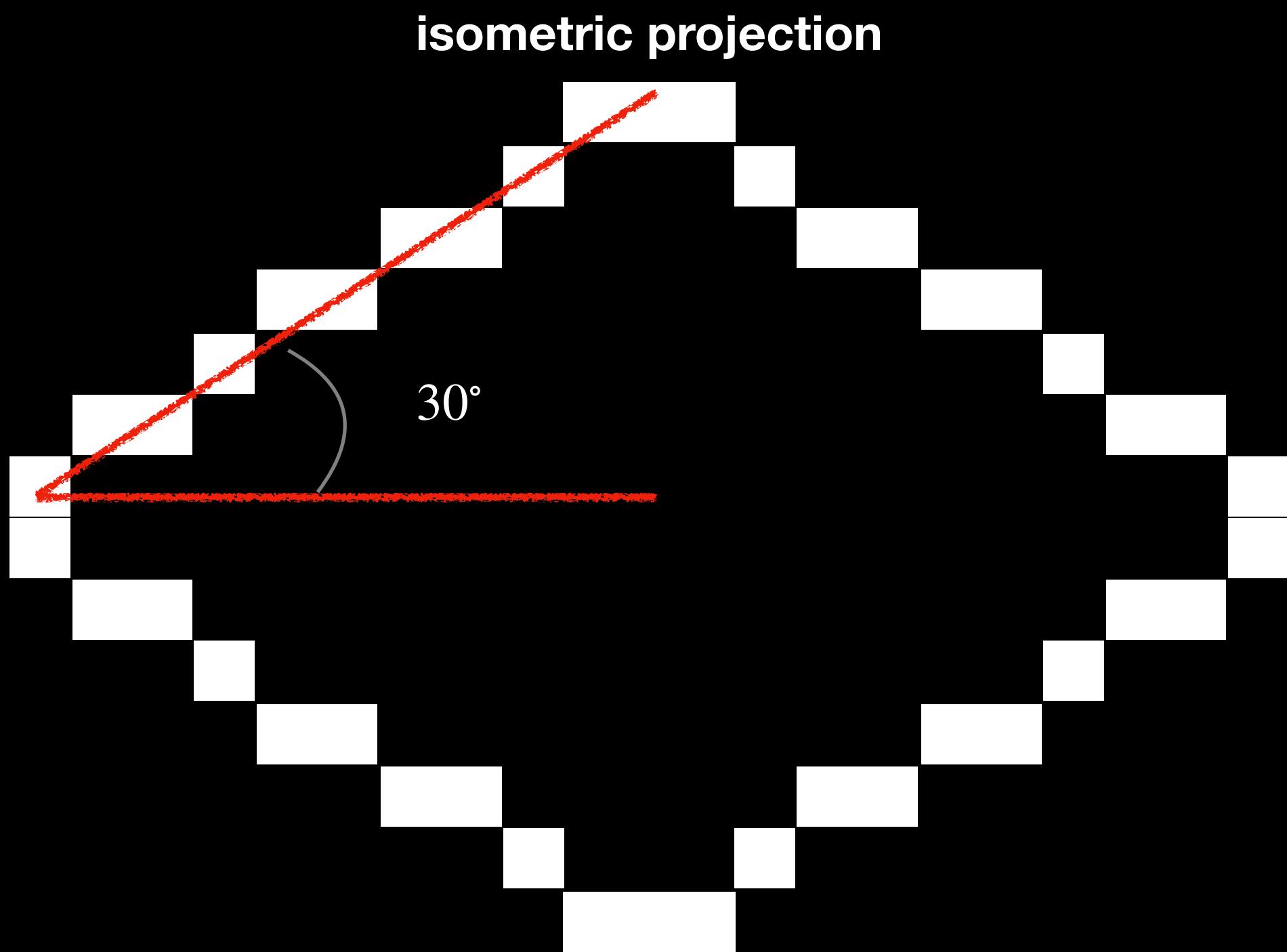
30° line



1:2 line

Isometric Projection

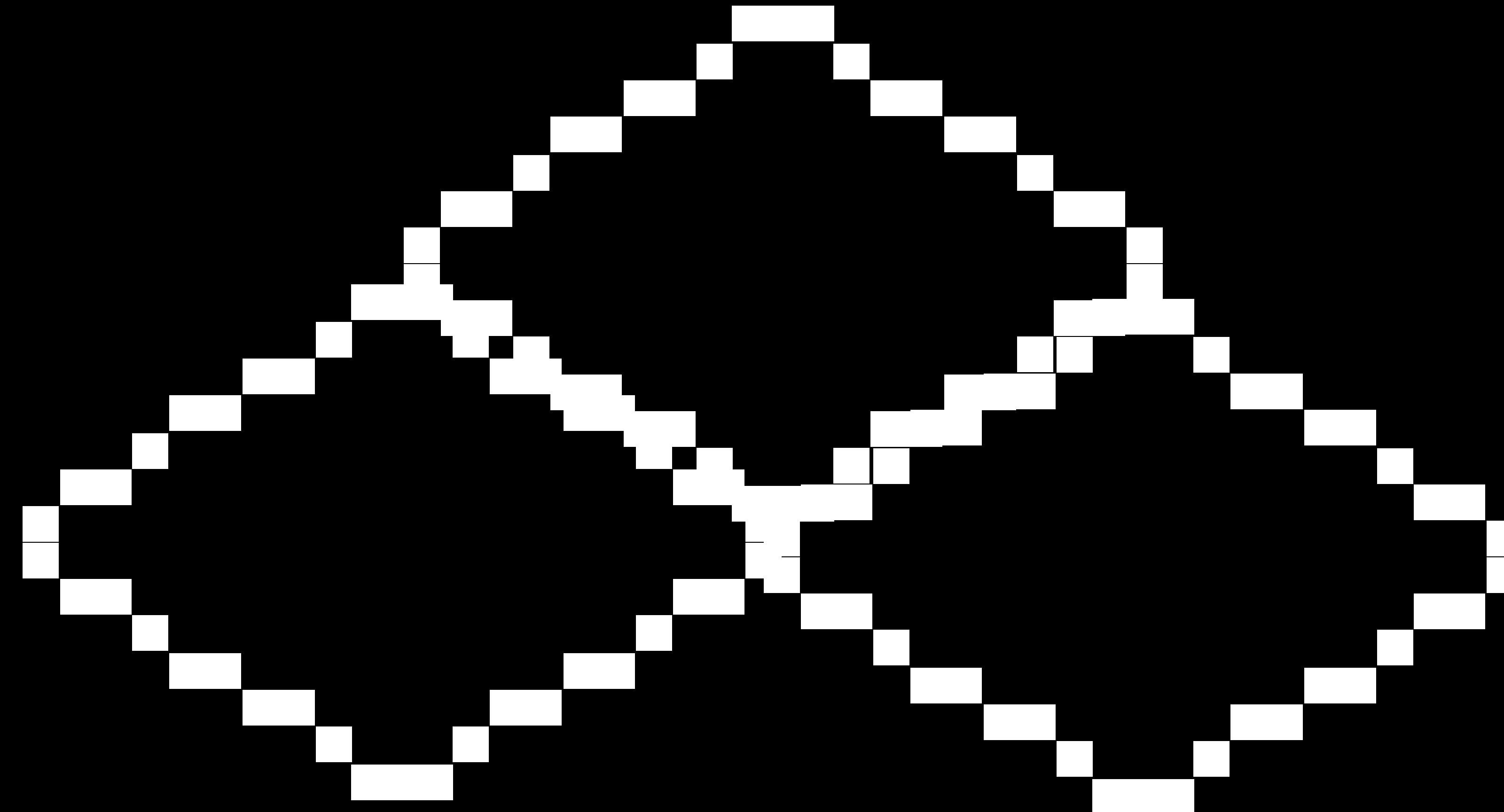
Isometric Projection in Computer Graphics



Isometric Projection

Isometric Projection in Computer Graphics

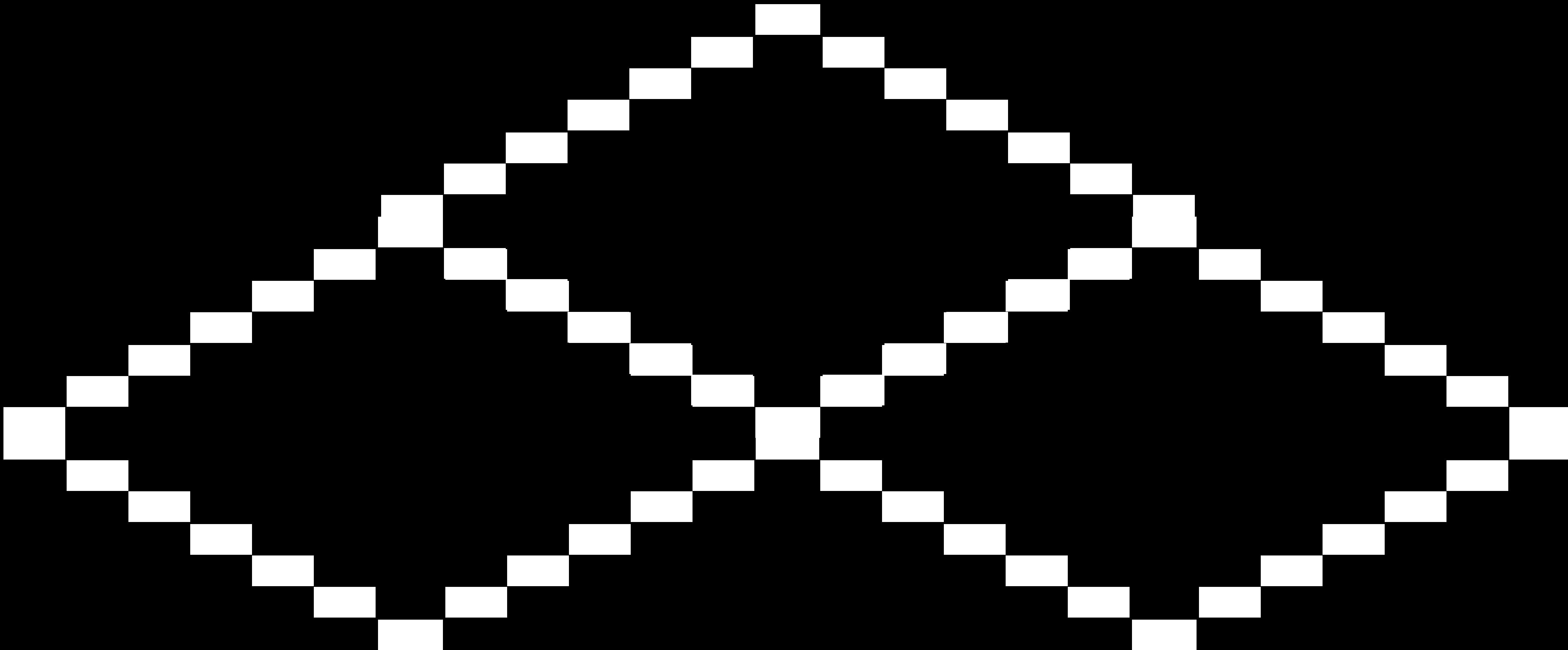
Isometric projection(30°) -> aw... weird...



Isometric Projection

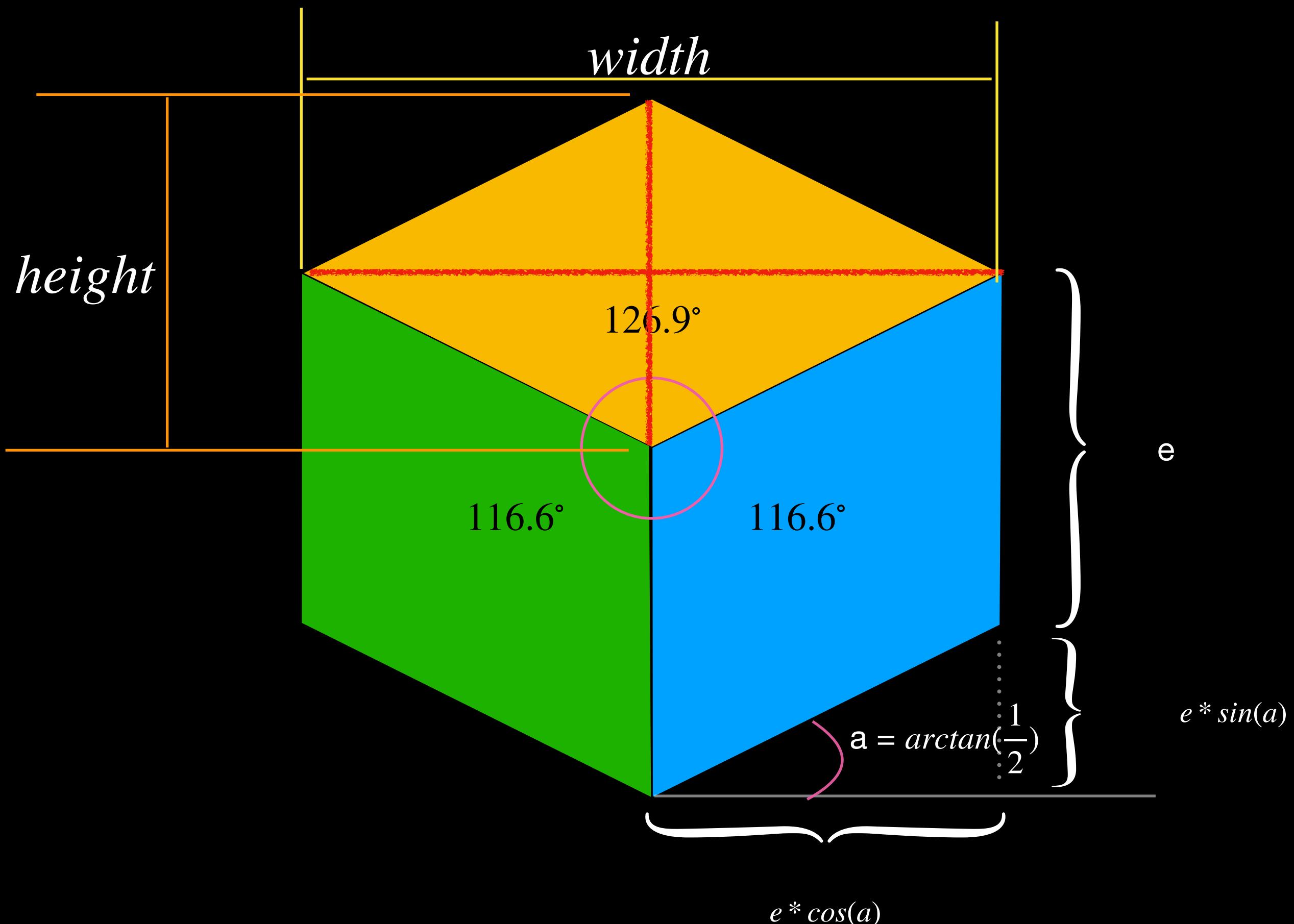
Isometric Projection in Computer Graphics

dimetric projection -> Cool~!



Isometric Projection

Isometric Projection in Computer Graphics

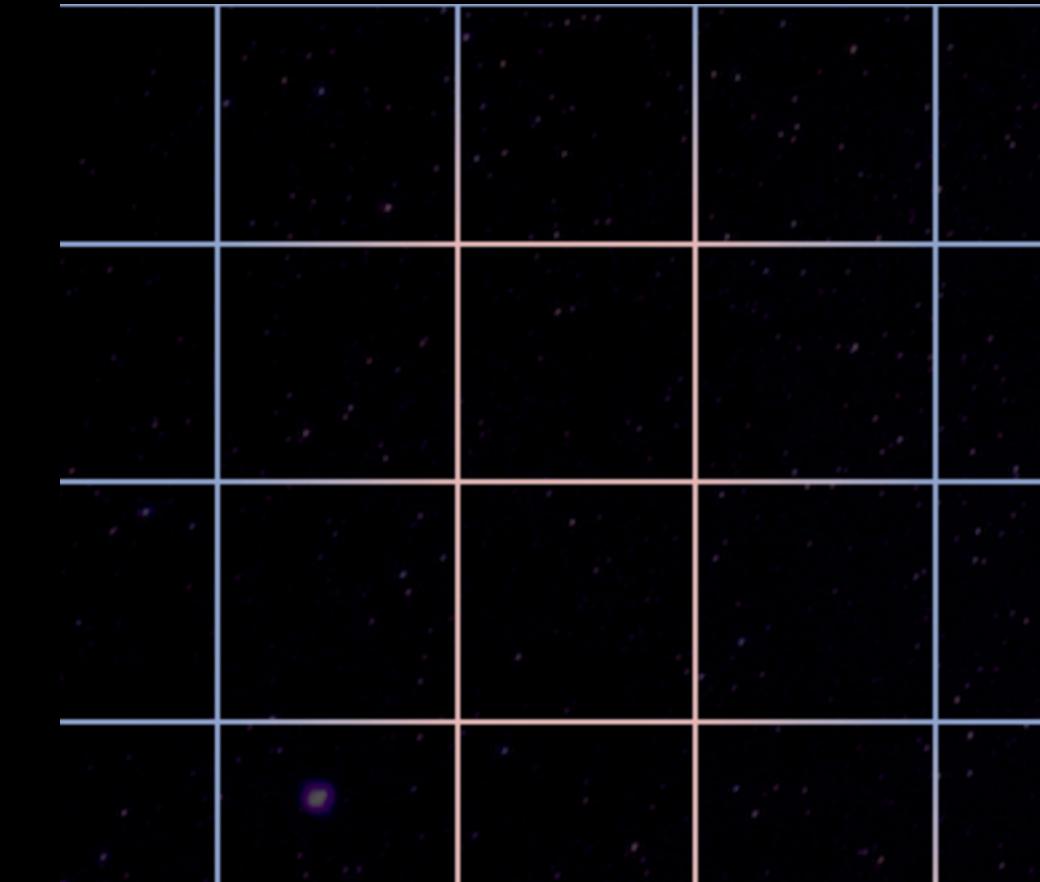
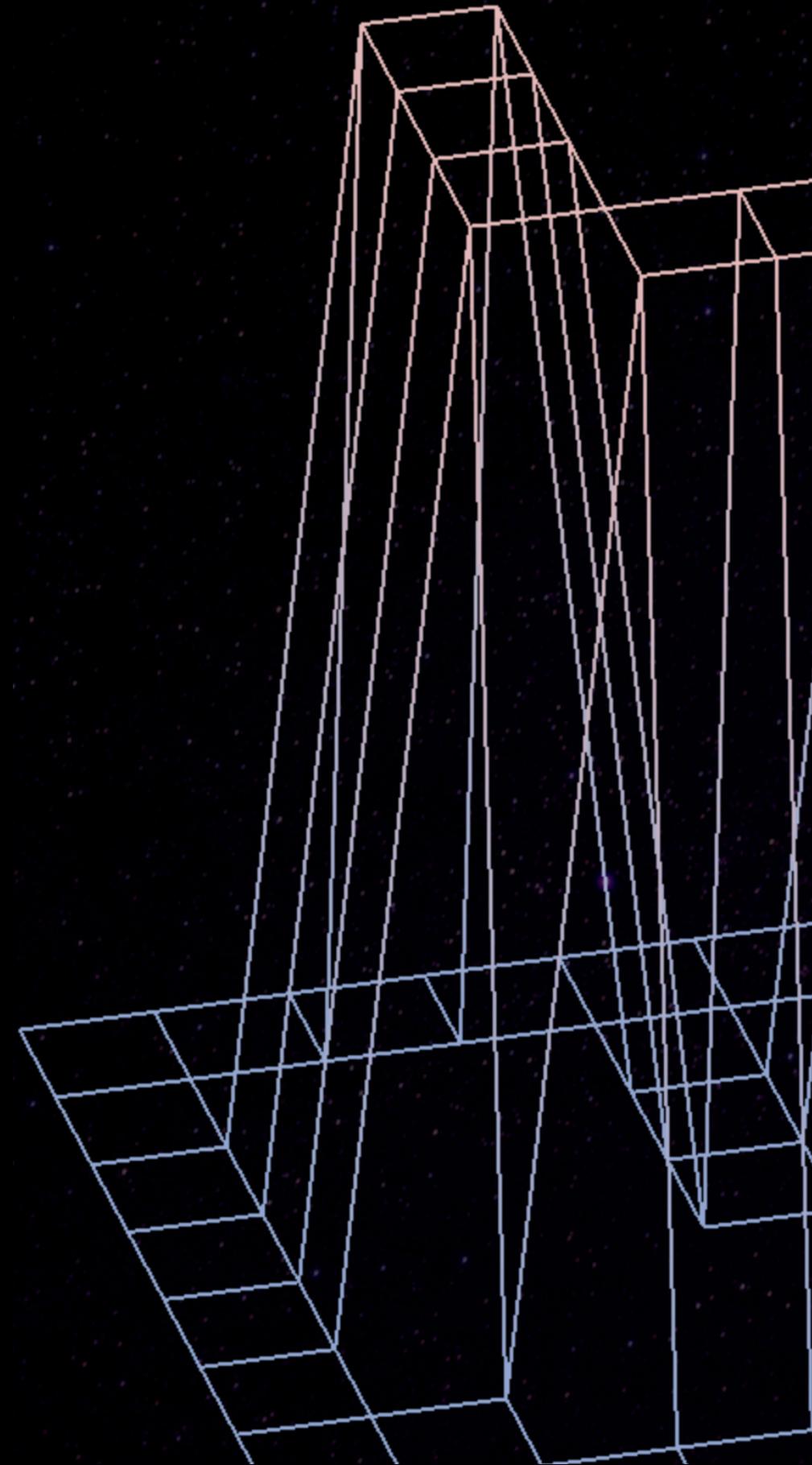


- $width : height = 2 : 1$
- $angle a = \arctan(\frac{1}{2})$
- 수학적으로 봤을 때 isometric projection이 아니라 dimetric projection임.
- 컴퓨터 그래픽에서는 isometric projection이 주로 dimetric projection을 뜻함!

Color Gradient

Color Gradient

About color gradient



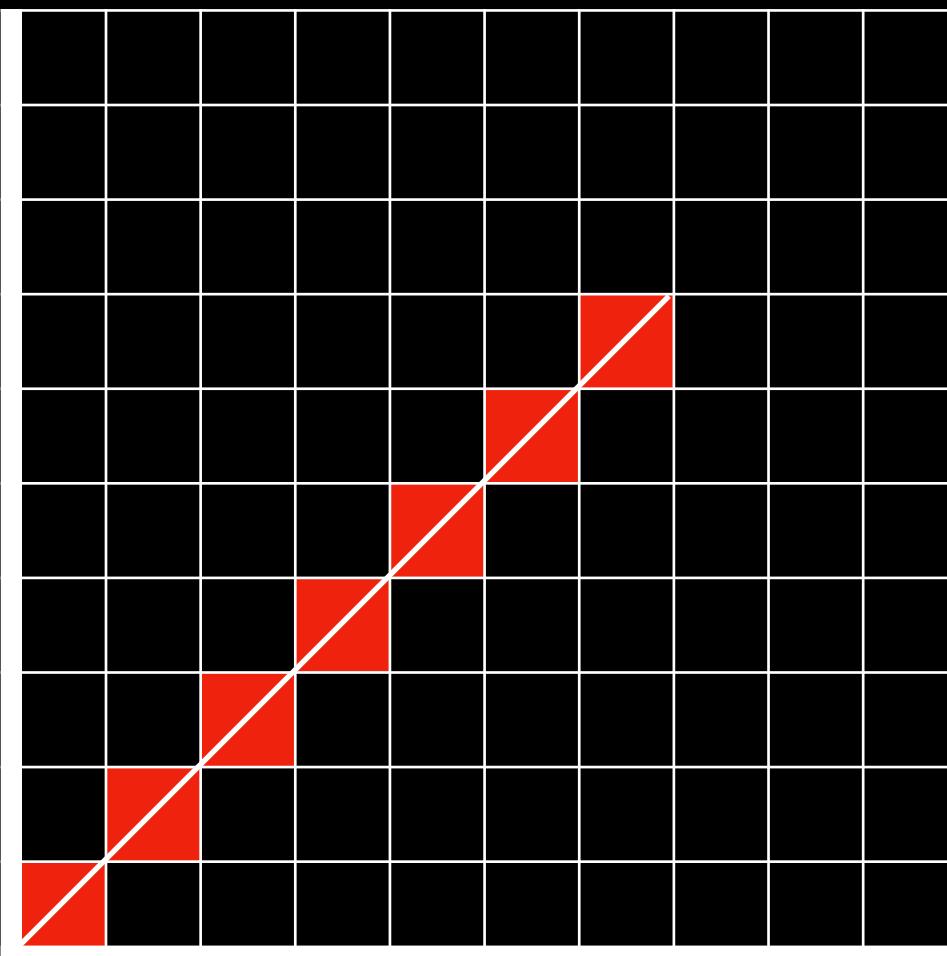
- 각 픽셀을 찍을 때 색깔을 입히면 됨.
- 이때 그라데이션으로 입혀야 하므로 시작점부터 끝점까지 색깔을 서서히 바꿔가면 됨.
- $(0,0)$ 에서 $(0,1)$ 까지 직선을 그린다면, $(0, 0.5)$ 좌표에서는 $[(0,0)\text{의 색상값} + (0,1)\text{의 색상값}] * 0.5(50\%)$ 을 해주는 방식
- 각 색상값은 RGB의 16진수로 이뤄져 있어, 비트 연산을 통해 계산 가능.

Line Drawing Algorithm

Line drawing algorithm

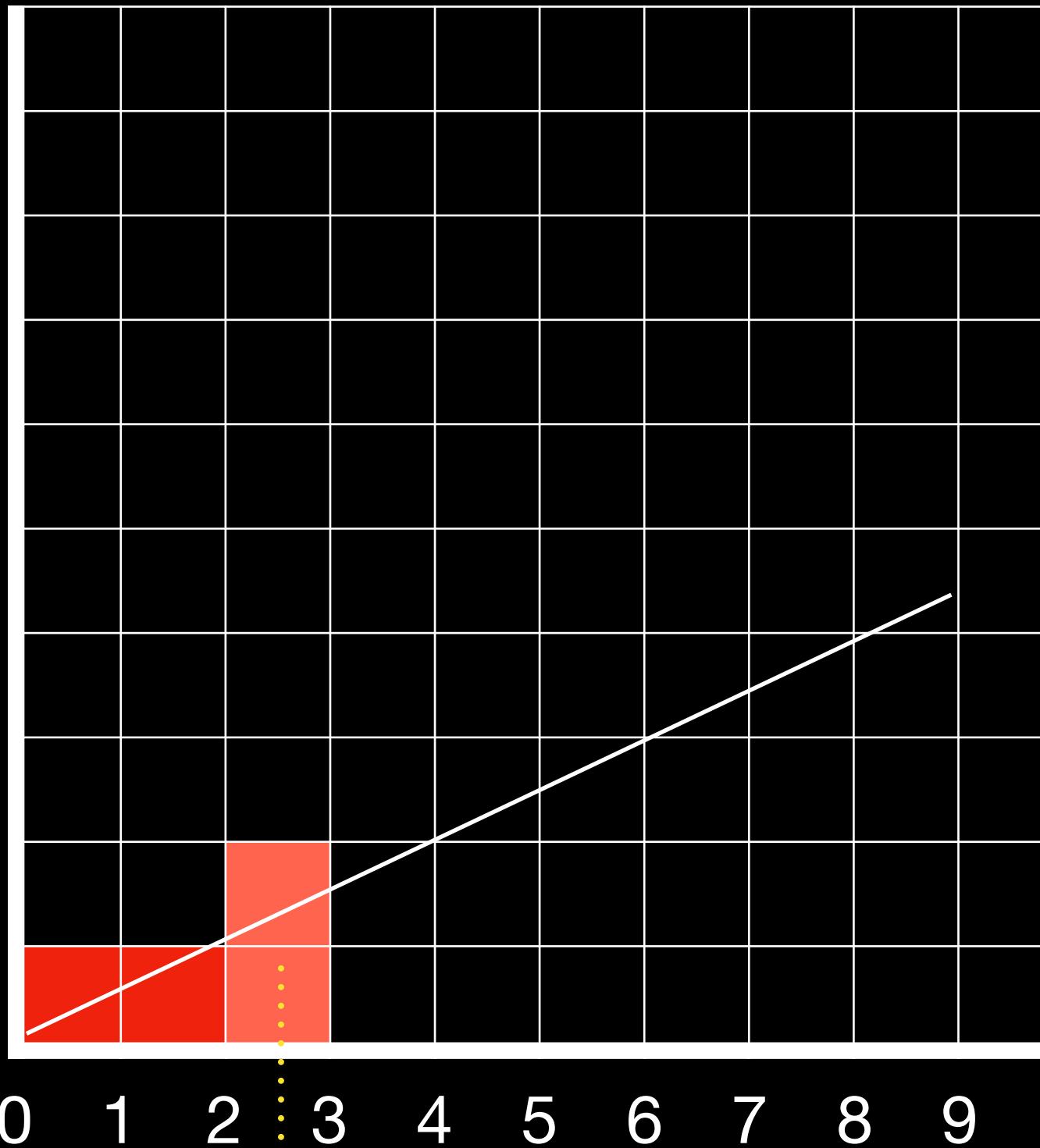
Bresenhem's line algorithm

- 컴퓨터에서 그리는 직선은 결국 픽셀의 모음으로 연속적이지 않음
- 따라서 필요한 정수 계산만 해서 픽셀을 찍어 선을 그리는 알고리즘
- x축 (or y축)의 한 점을 시작점으로 잡고 1씩 증가 시키면서 y축(or x축) 좌표를 증가 시킬 것인지, 유지 할 것인지 판별하여 픽셀을 입력
 $y = x$

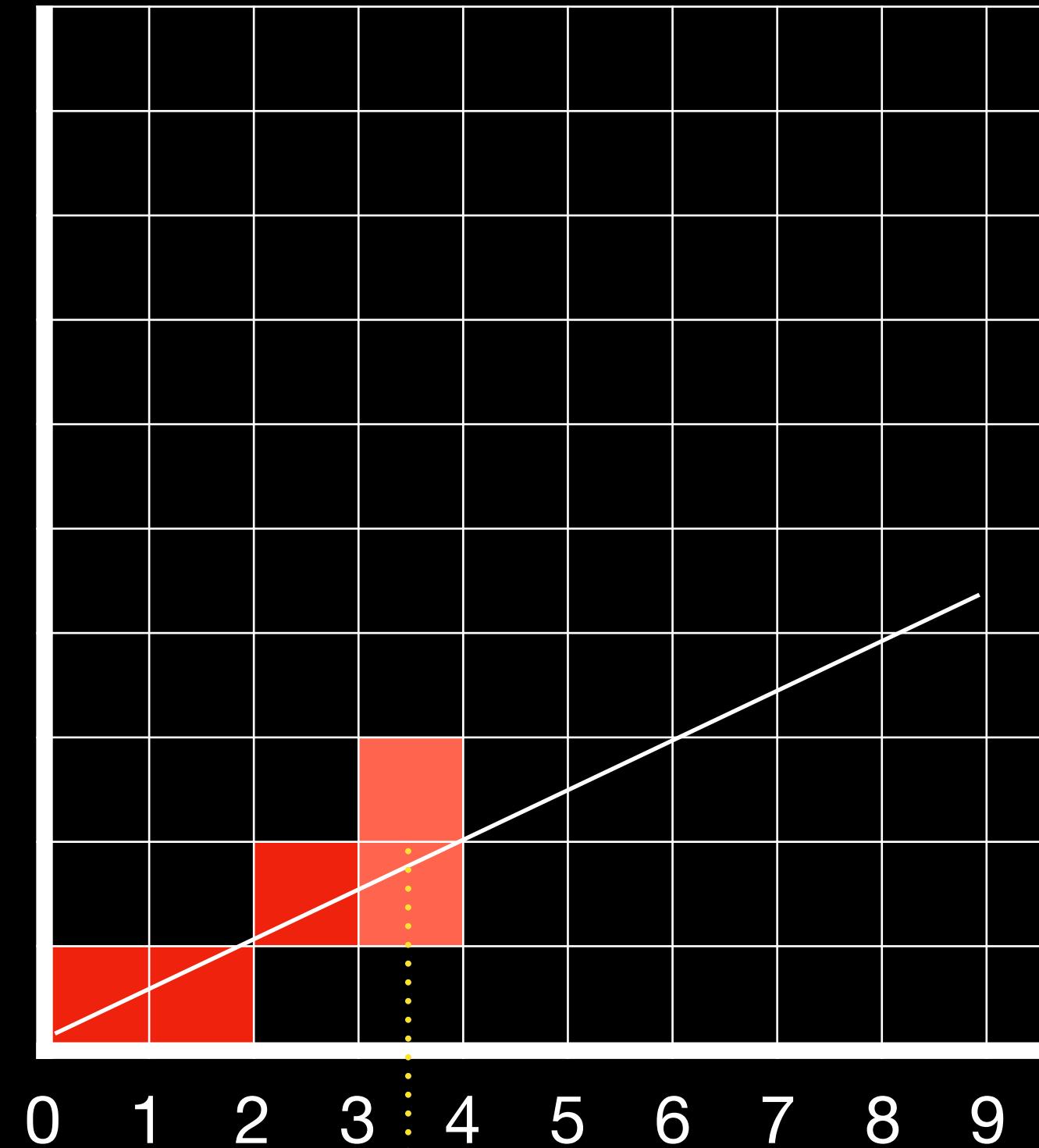


Line drawing algorithm

Bresenhem's line algorithm



중간점의 y좌표와 직선
의 y좌표를 비교하여
어디에 픽셀을 입력할지
판별



중간점의 y좌표와 직선
의 y좌표를 비교하여
어디에 픽셀을 입력할지
판별

Line drawing algorithm

Bresenhem's line algorithm

- 이전 예시에서 기울기가 1보다 작고, x축이 증가하는 방향으로 선을 그림.
- 기울기가 1보다 크면 $y = x$ 대칭이동, 즉 x와 y를 스왑해서 픽셀을 찍으면 됨.
- projection을 하면 시작점의 x좌표(x_0)가 도착점의 x좌표(x_1)보다 커질 수 있는데, x_0 과 x_1 을 스왑한 뒤 같은 논리로 진행하면 됨.

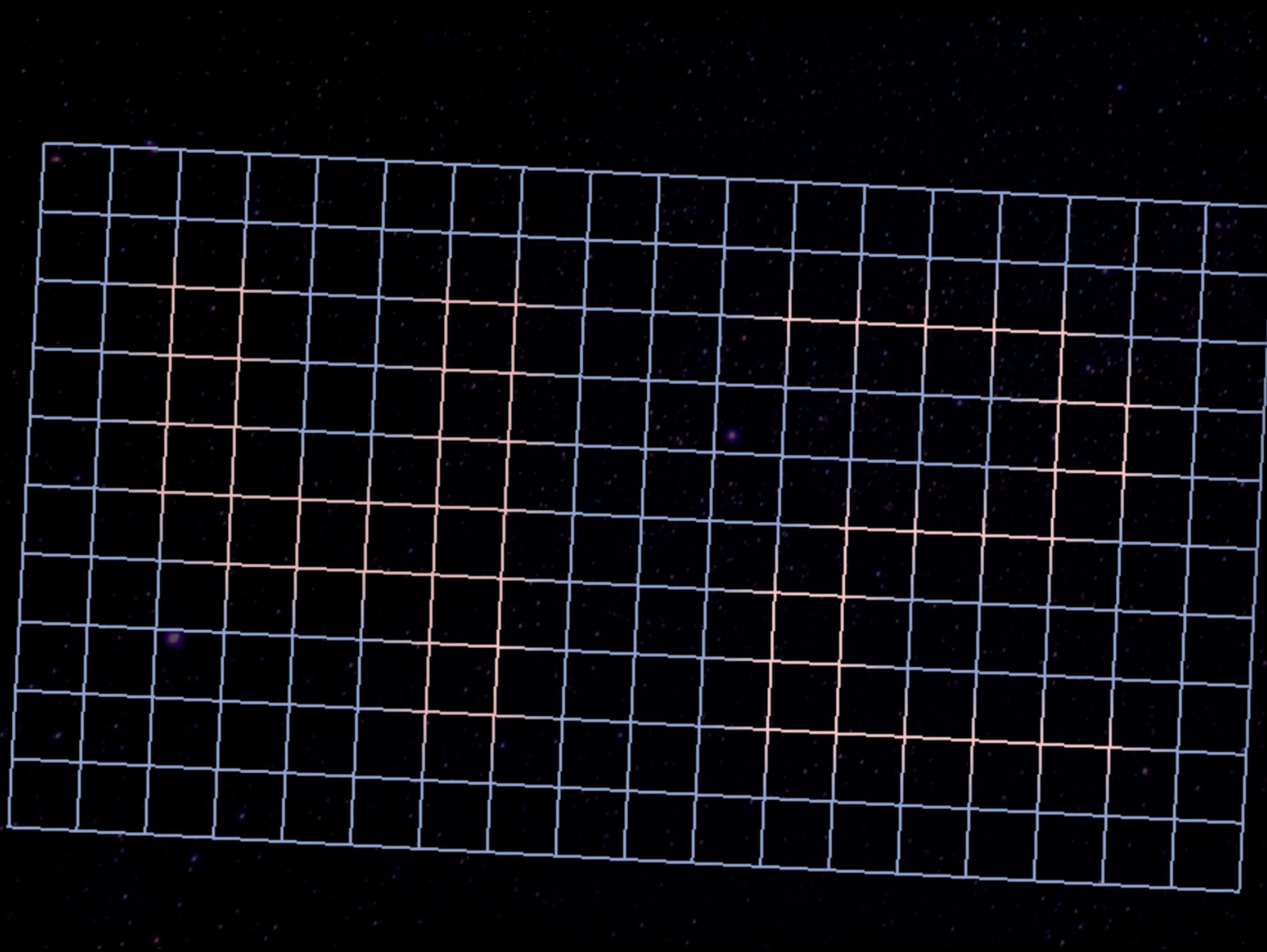
Anti-aliased Line

Anti-aliased Line

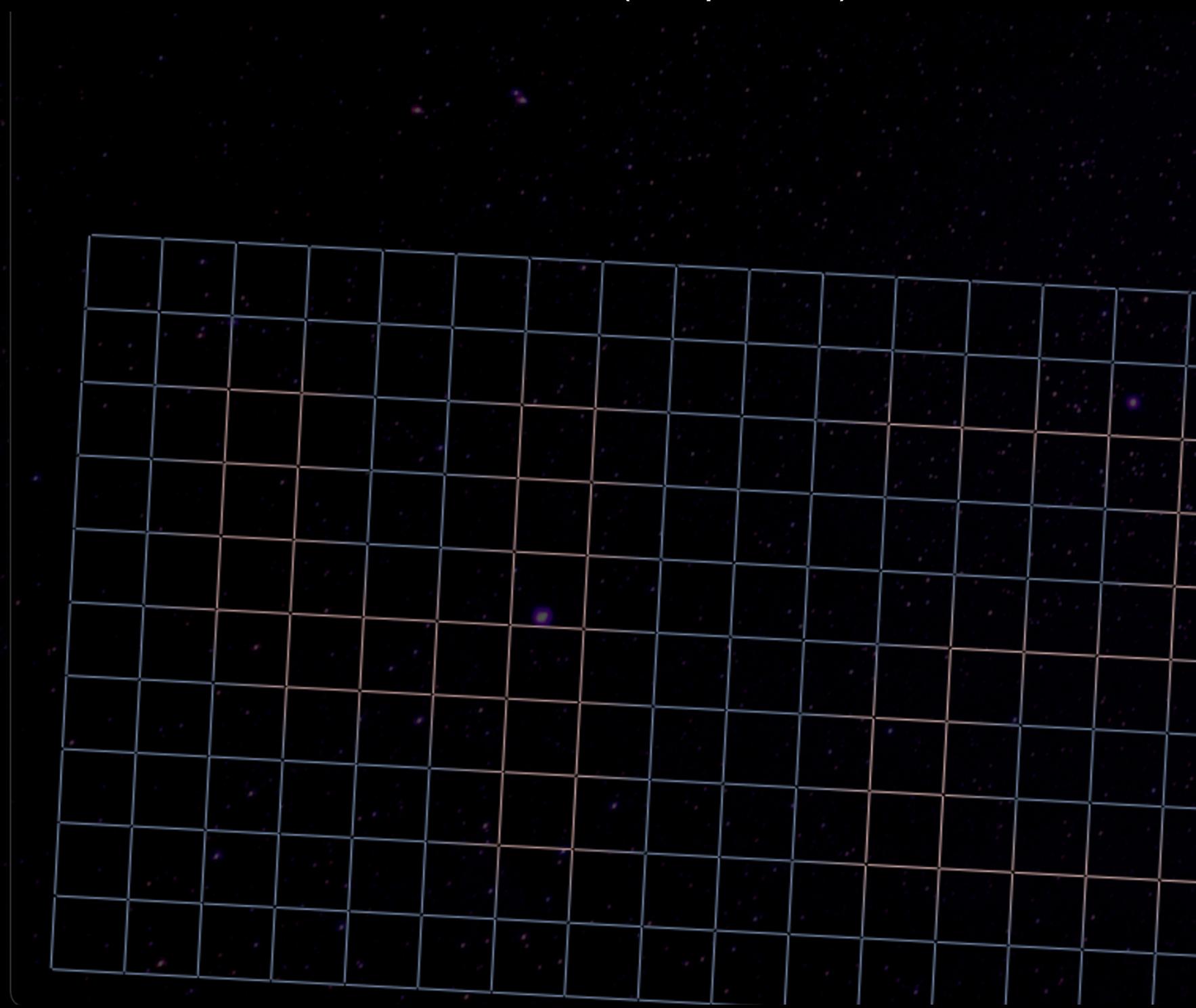
About aliasing

- 픽셀로 선을 그리다 보니 선이 끊어져 보이는 계단현상(aliasing) 발생(참고)

Aliased



anti-aliased(not perfect)

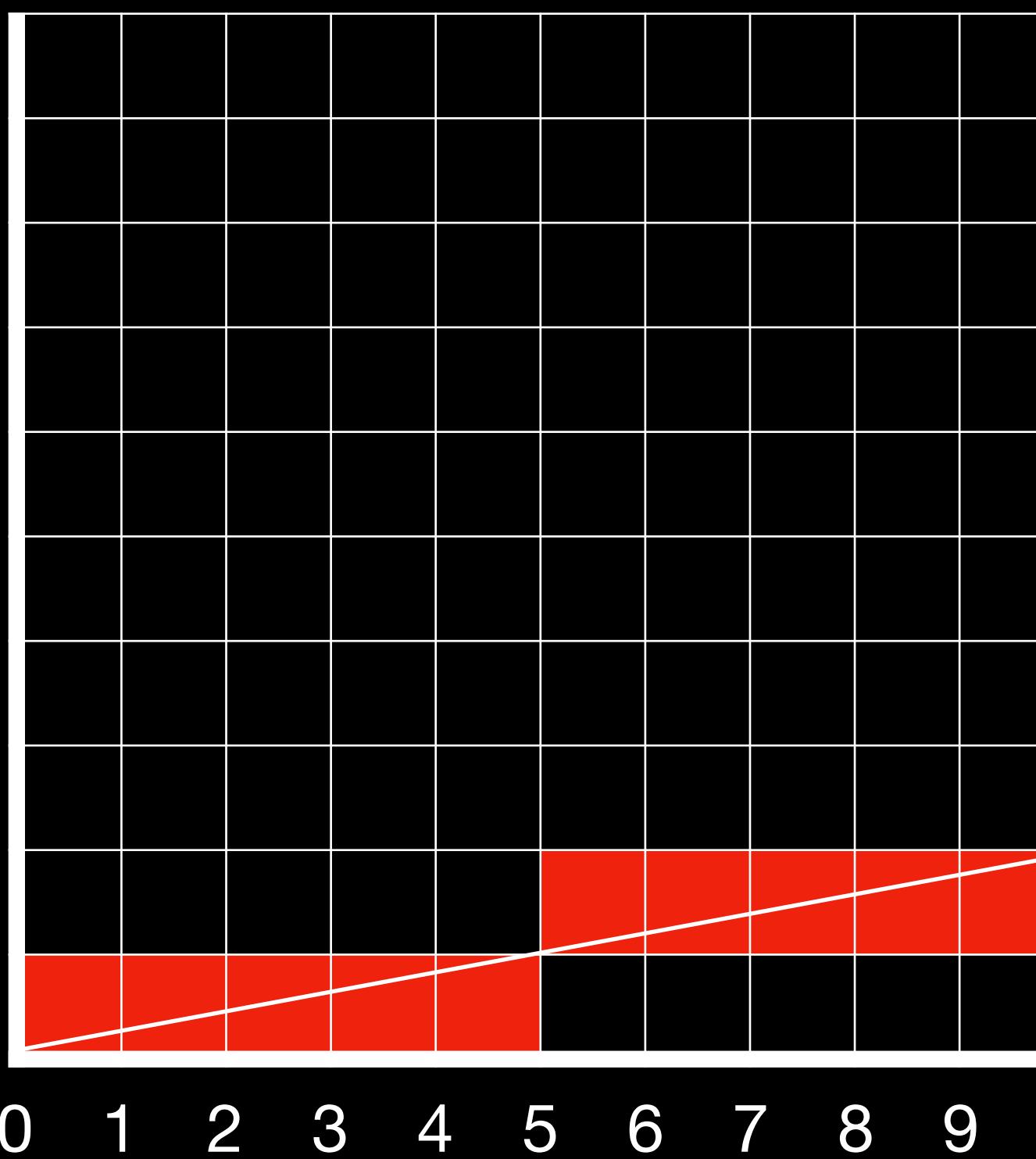


Anti-aliased Line

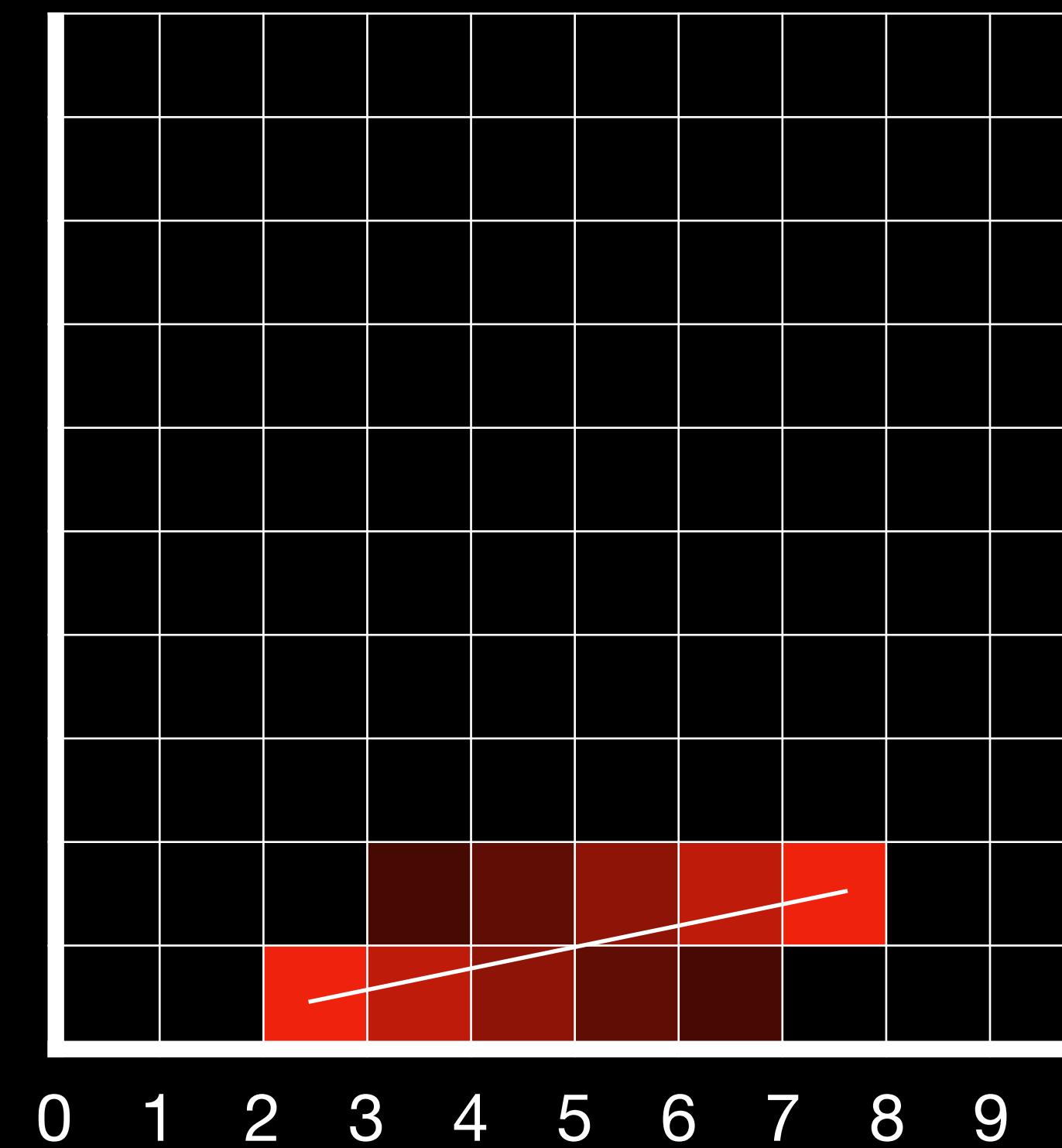
About Xiaolin Wu' line algorithm

- Anti-aliased line을 그리는 알고리즘 -> xiaolin wu's line algorithm

Bresenham's

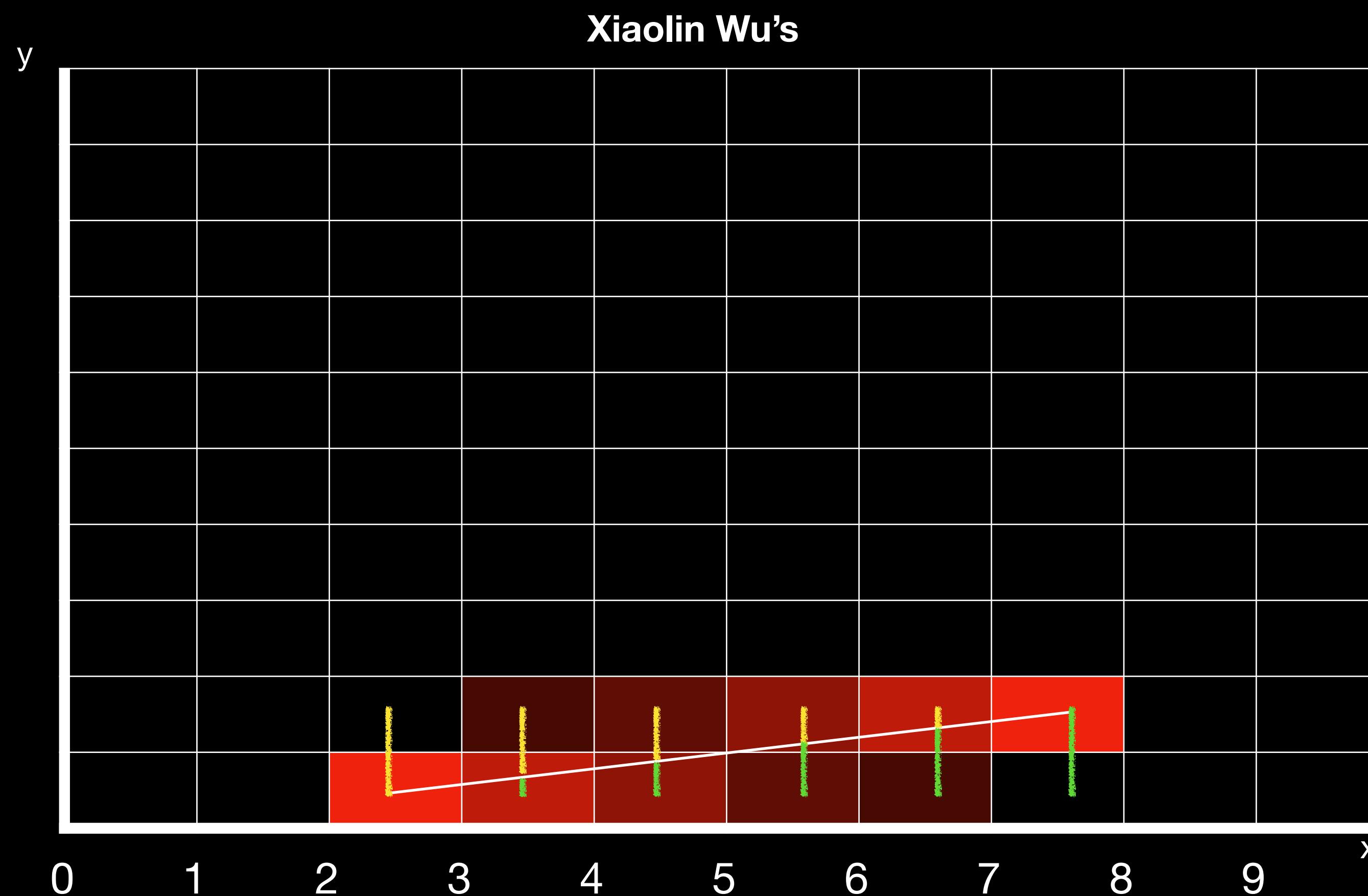


Xiaolin Wu's



Anti-aliased Line

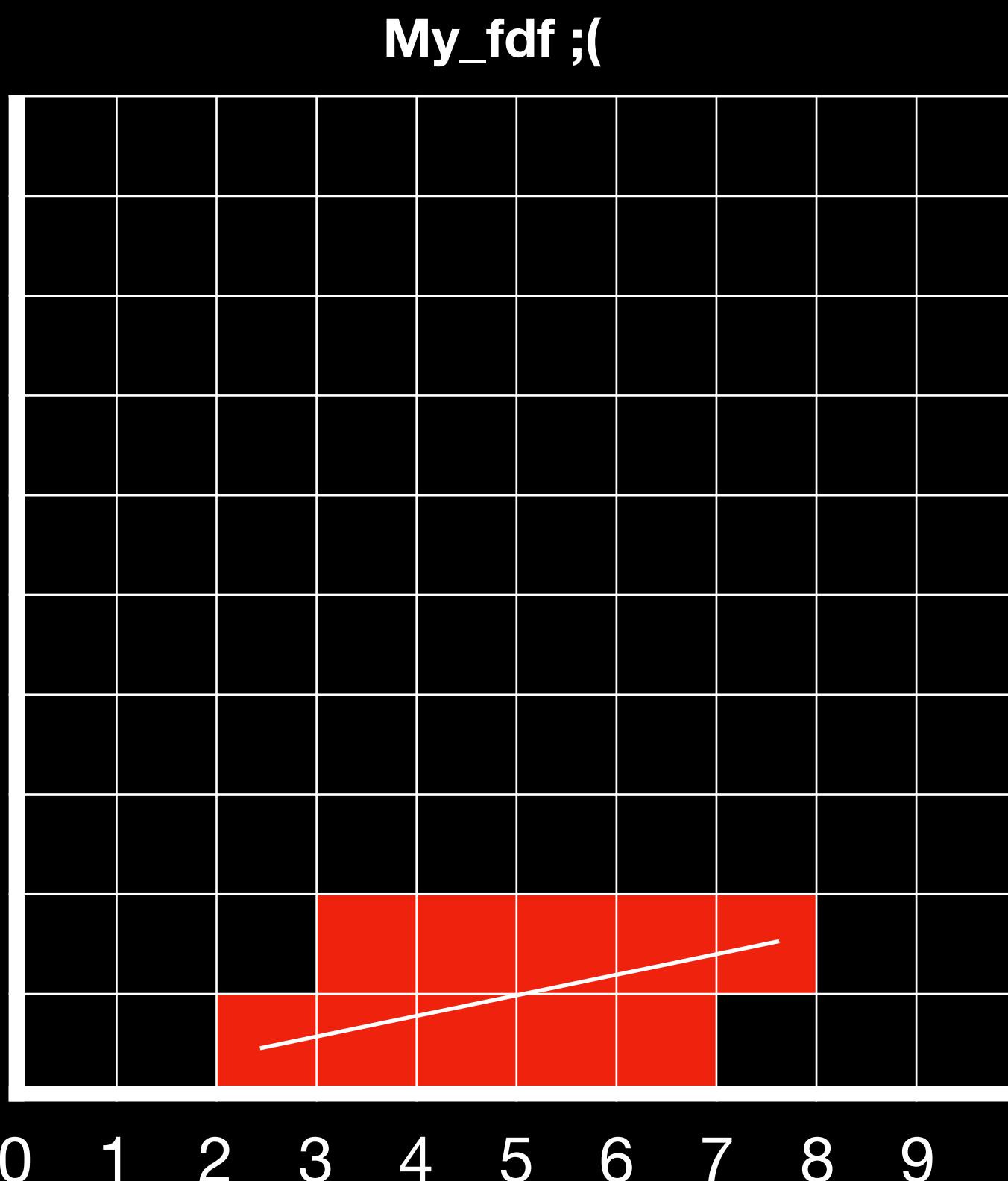
About Xiaolin Wu' line algorithm



- 각 픽셀의 위치를 실수값으로 계산하면 y좌표의 중점(y_m)을 얻을 수 있다.
- y_m 과 $y_m + 1$ 사이에 직선이 위치하는데, 이 직선의 좌표를 y_{cur} 이라고 하자.
- y_m 과 y_{cur} 좌표의 차이를 y_d 라고 하면, y_d 와 $1 - y_d$ 값을 구하여 색깔을 어느정도 투명하게 해야 할지 퍼센테이지를 구할 수 있다.
- 이를 기반으로 색깔을 조절하여 구현

Anti-aliased Line

My Line drawing in fdf



- Anti-aliasing와 color gradient를 동시에 적용하려면, 그라데이션이 적용된 색깔을 기반으로 anti-aliasing을 해야 하는데 이 과정이 너무 복잡함.
- 투명도를 조절하는 방식으로 구현 해봤으나, 이미지 퀄리티가 상당히 떨어짐.
- 왼쪽 그림처럼 라인을 그린 꼴이 되었으나, aliasing 현상이 있지만 해상도가 좋아지는 긍정적인 효과가 있음 😊

BONUS

zoom

Zoom

Called zoom, it's scaling though

- 줌인 / 줌아웃은 프로젝션된 x, y좌표에 같은 숫자를 곱해주는 방법으로 구현할 수 있음.
- 예를 들어, $(1, 1)$ 에서 $(2, 5)$ 로 이어지는 직선이 있고 10배 줌을 한다면 각 x, y값에 10씩 곱한 뒤 직선을 그리면 됨!
- scale이란 변수를 만들어 입력을 받을 때마다 증가 / 감소 시켜주면 잘 작동함.

Translation

Translation

About Translation(평행이동)

- fdf 를 상/하/좌/우로 이동시켜야 함.
- (x, y, z) 좌표를 projection하여 (x, y) 좌표계로 옮긴 뒤 이동시킴.
- 키 입력을 받을 때마다 모든 x, y 좌표를 증가 / 감소 시켜주는 방식으로 구현.

Rotation

Rotation

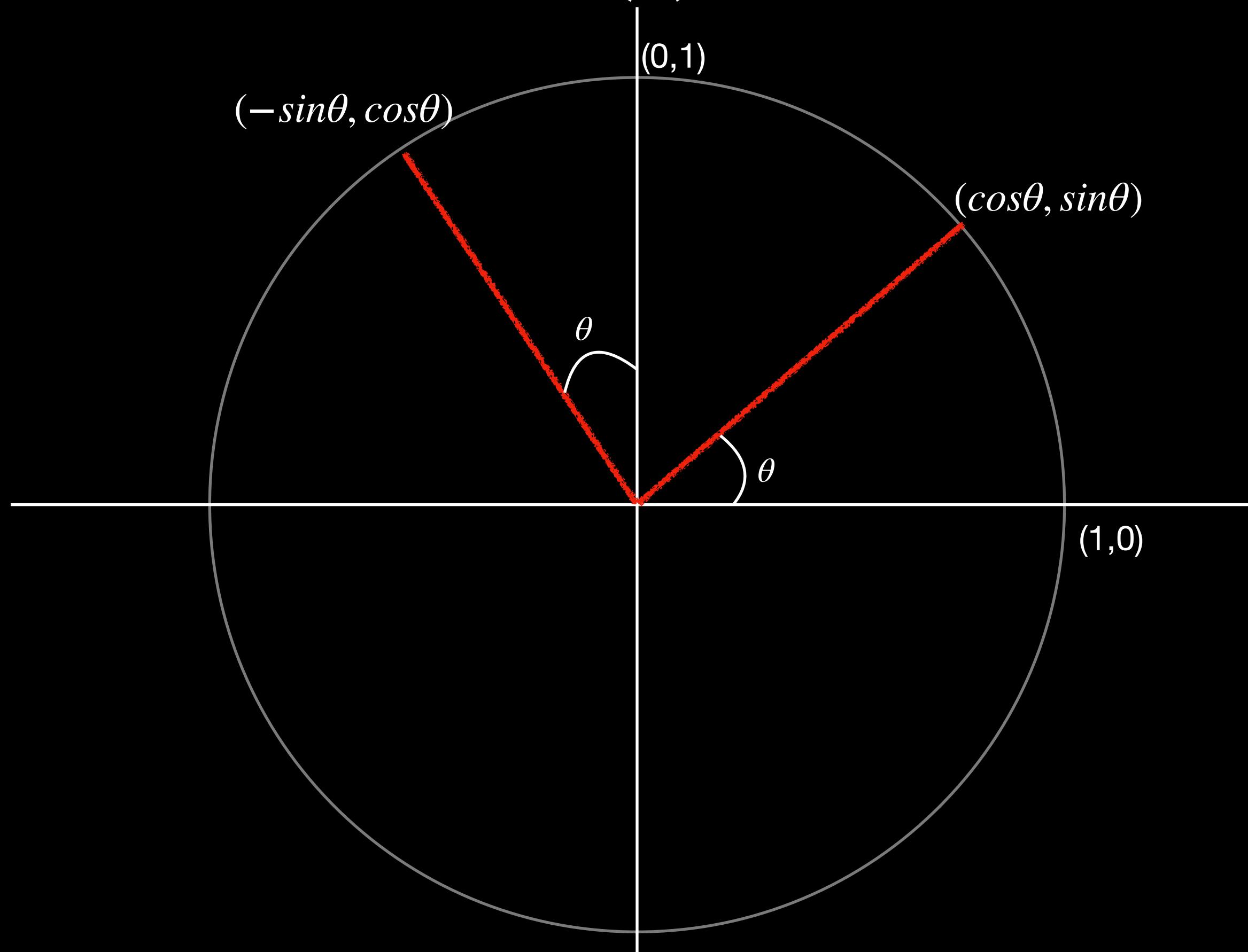
About Rotation (회전변환)

- (x, y, z) 좌표에 회전 변환을 한 뒤 projection을 하면 더 다채로운 변환이 가능.
- (x, y, z) 좌표에 회전 행렬을 곱해주면 됨.

Rotation Matrix

About Rotation Matrix(회전행렬)

- 2차원 회전 행렬 $R(\theta)$

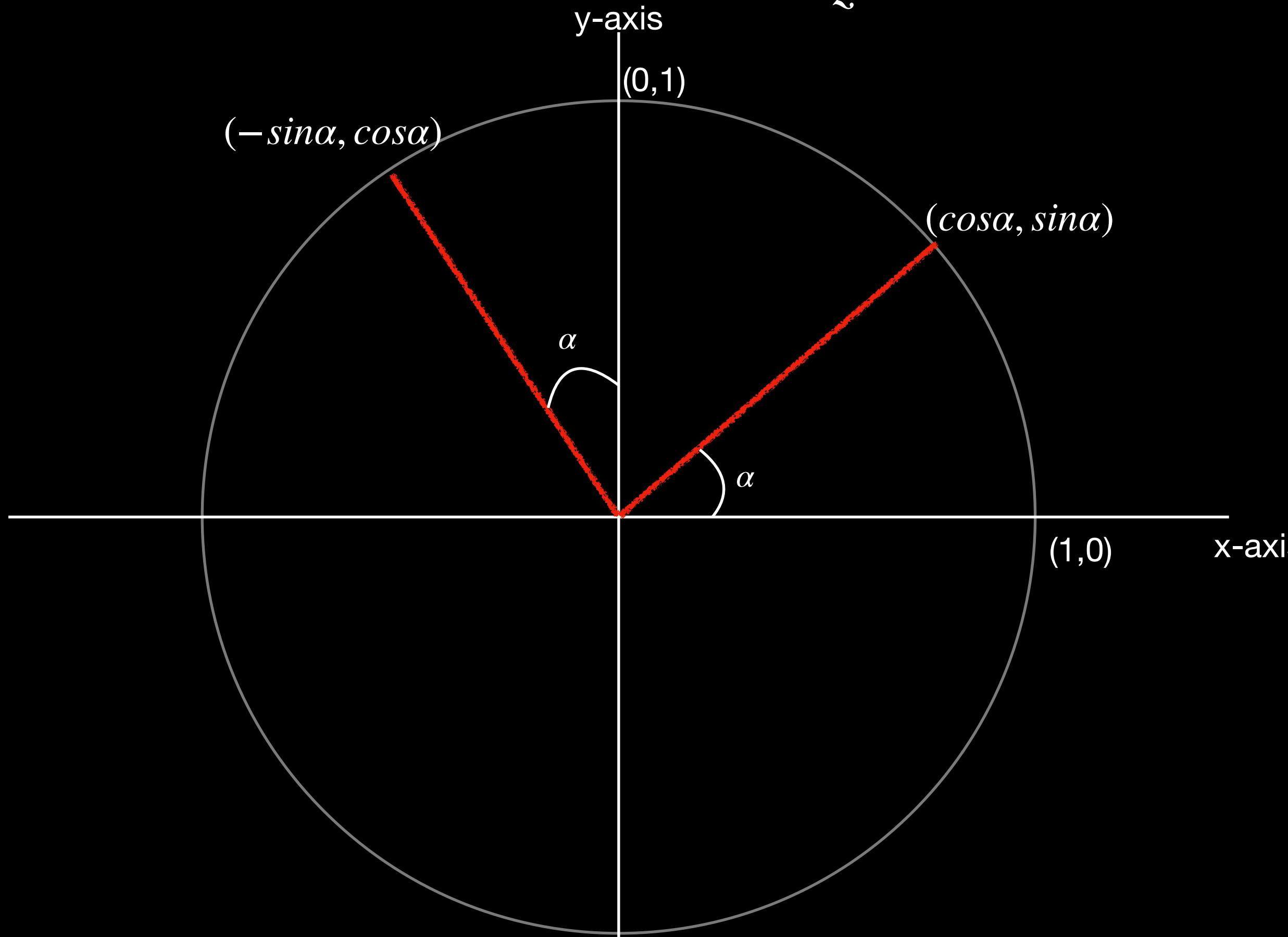


$$\begin{pmatrix} x_{rotated} \\ y_{rotated} \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} * \begin{pmatrix} x \\ y \end{pmatrix}$$

Rotation Matrix

About Rotation Matrix(회전행렬)

- 3차원 z축 기준 회전 행렬 $R_z(\alpha)$

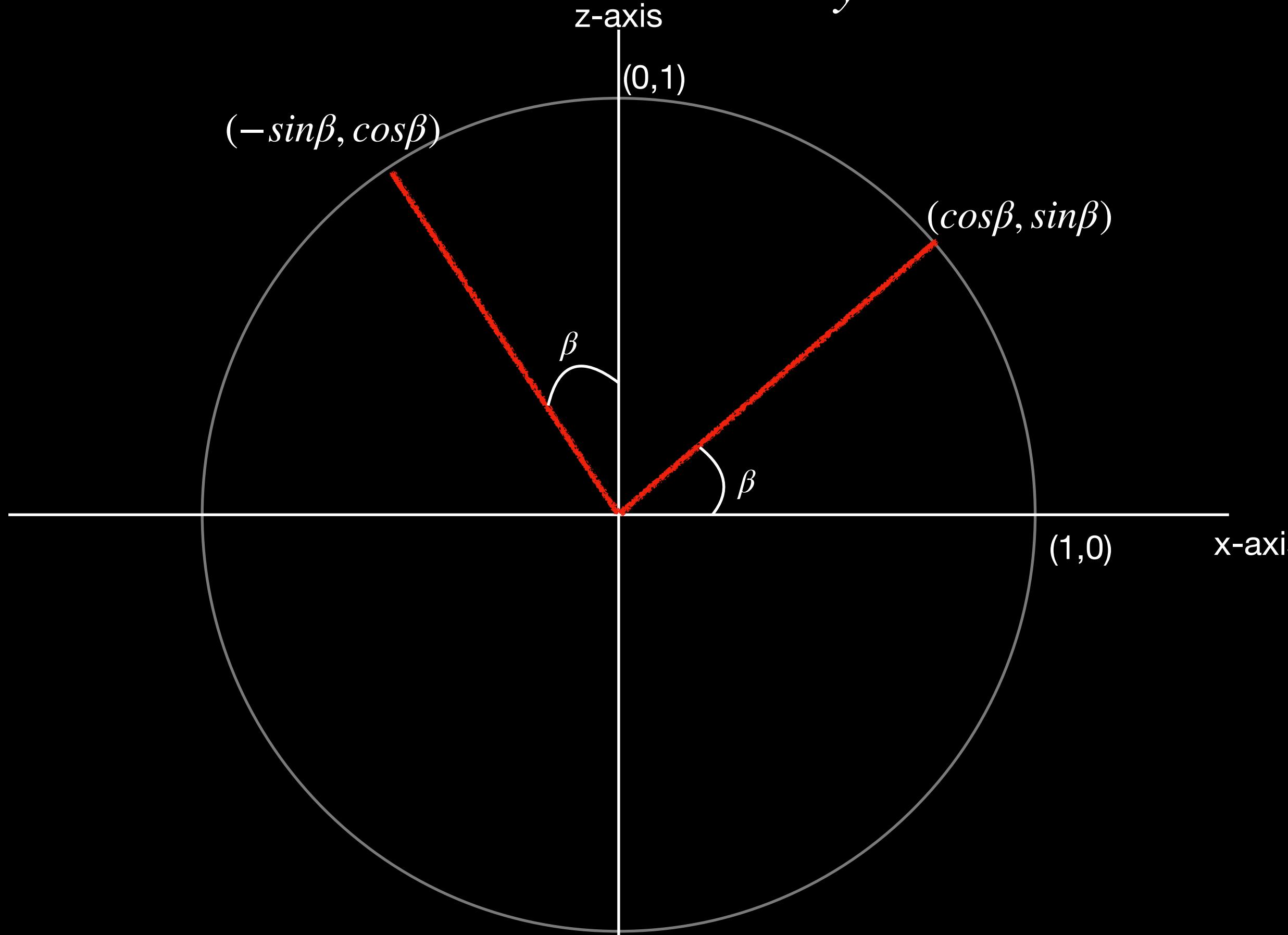


$$\begin{pmatrix} x_{rotated} \\ y_{rotated} \\ z_{rotated} \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Rotation Matrix

About Rotation Matrix(회전행렬)

- 3차원 y축 기준 회전 행렬 $R_y(\beta)$

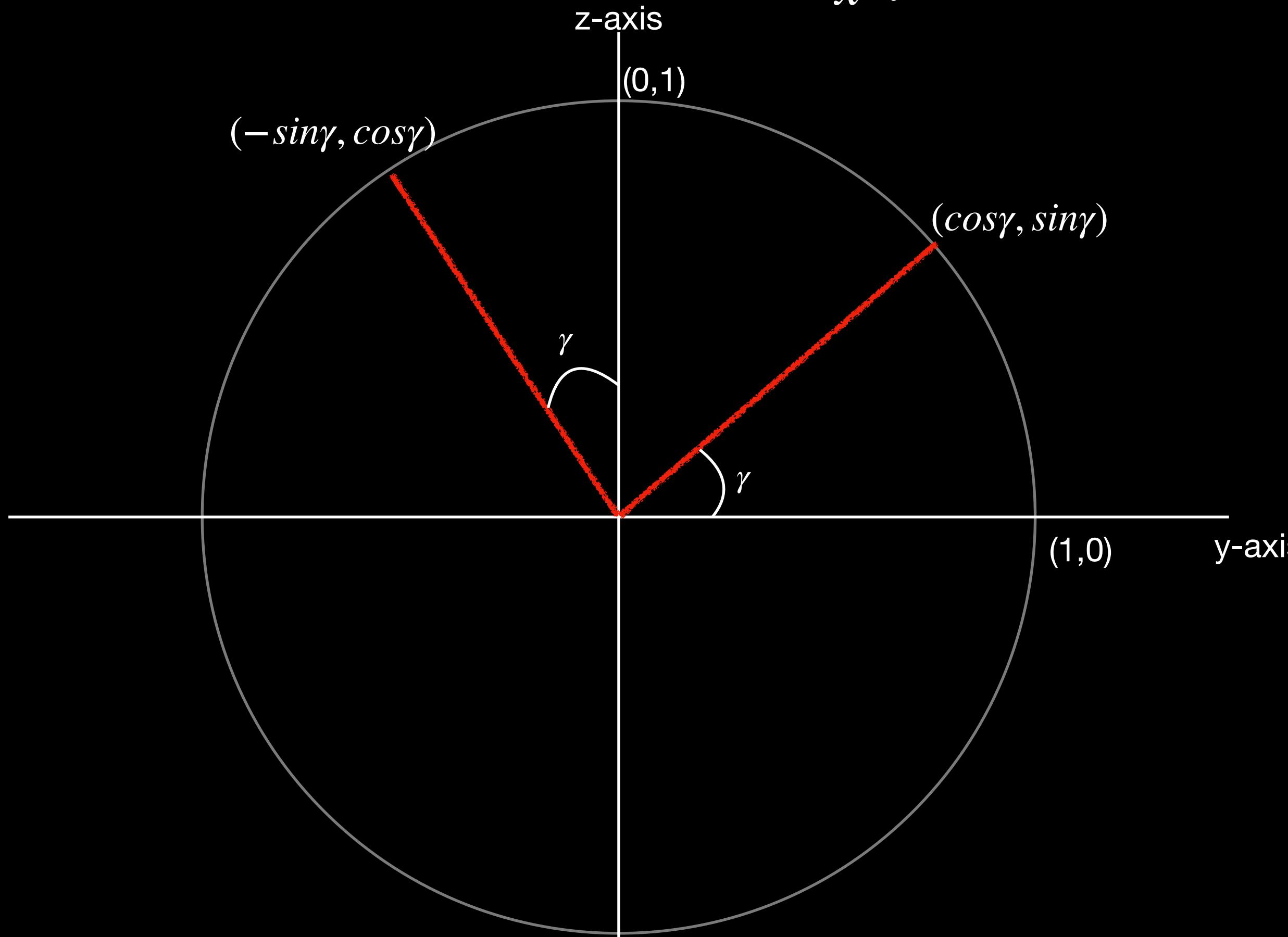


$$\begin{pmatrix} x_{rotated} \\ y_{rotated} \\ z_{rotated} \end{pmatrix} = \begin{pmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} * \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Rotation Matrix

About Rotation Matrix(회전행렬)

- 3차원 x축 기준 회전 행렬 $R_x(\gamma)$



$$\begin{pmatrix} x_{rotated} \\ y_{rotated} \\ z_{rotated} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{pmatrix} * \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Rotation Matrix

About Rotation Matrix(회전행렬)

- 3차원 회전 행렬 $R = R_z(\alpha)R_y(\beta)R_x(\gamma)$

$$\begin{pmatrix} x_{rotated} \\ y_{rotated} \\ z_{rotated} \end{pmatrix} = \begin{pmatrix} \cos(\alpha)\cos(\beta) & \cos(\alpha)\sin(\beta)\sin(\gamma) - \sin(\alpha)\cos(\gamma) & \cos(\alpha)\sin(\beta)\cos(\gamma) + \sin(\alpha)\sin(\gamma) \\ \sin(\alpha)\cos(\beta) & \sin(\alpha)\sin(\beta)\sin(\gamma) + \cos(\alpha)\cos(\gamma) & \sin(\alpha)\sin(\beta)\cos(\gamma) - \cos(\alpha)\sin(\gamma) \\ -\sin(\beta) & \cos(\beta)\sin(\gamma) & \cos(\beta)\cos(\gamma) \end{pmatrix} * \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$