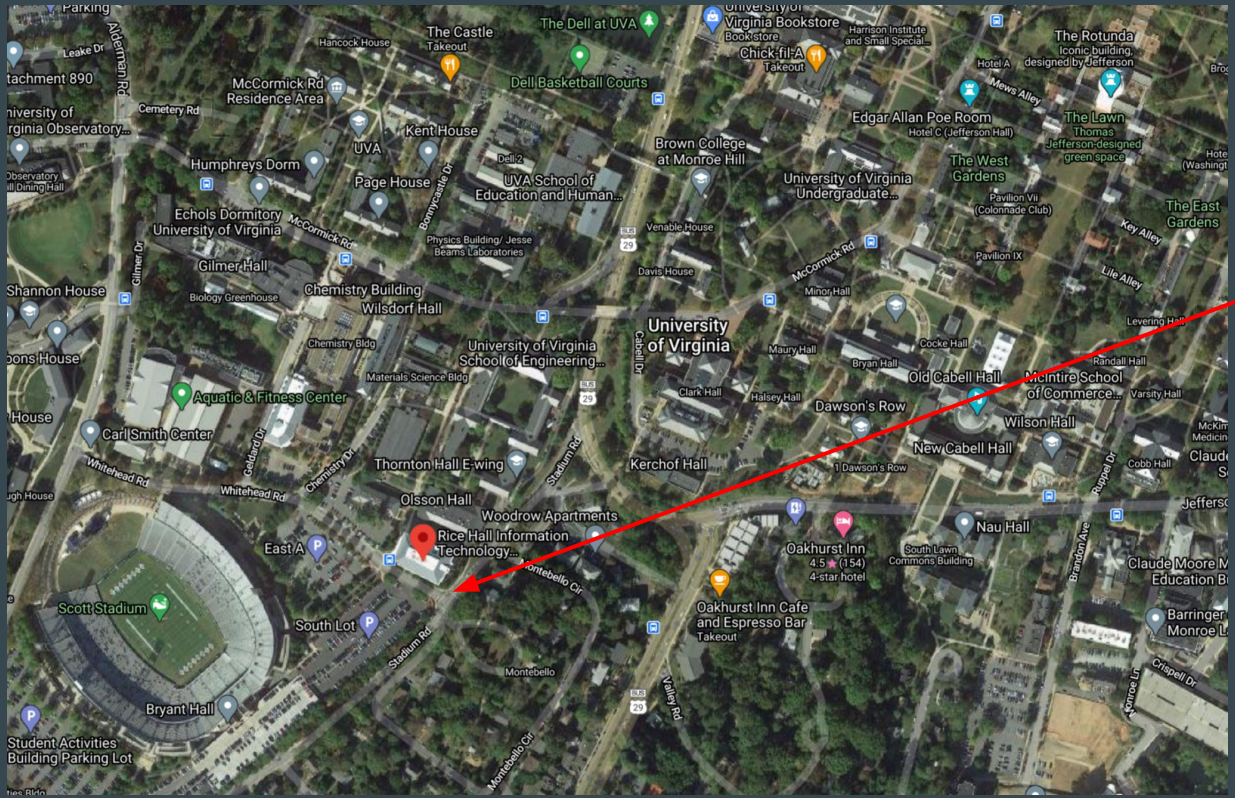
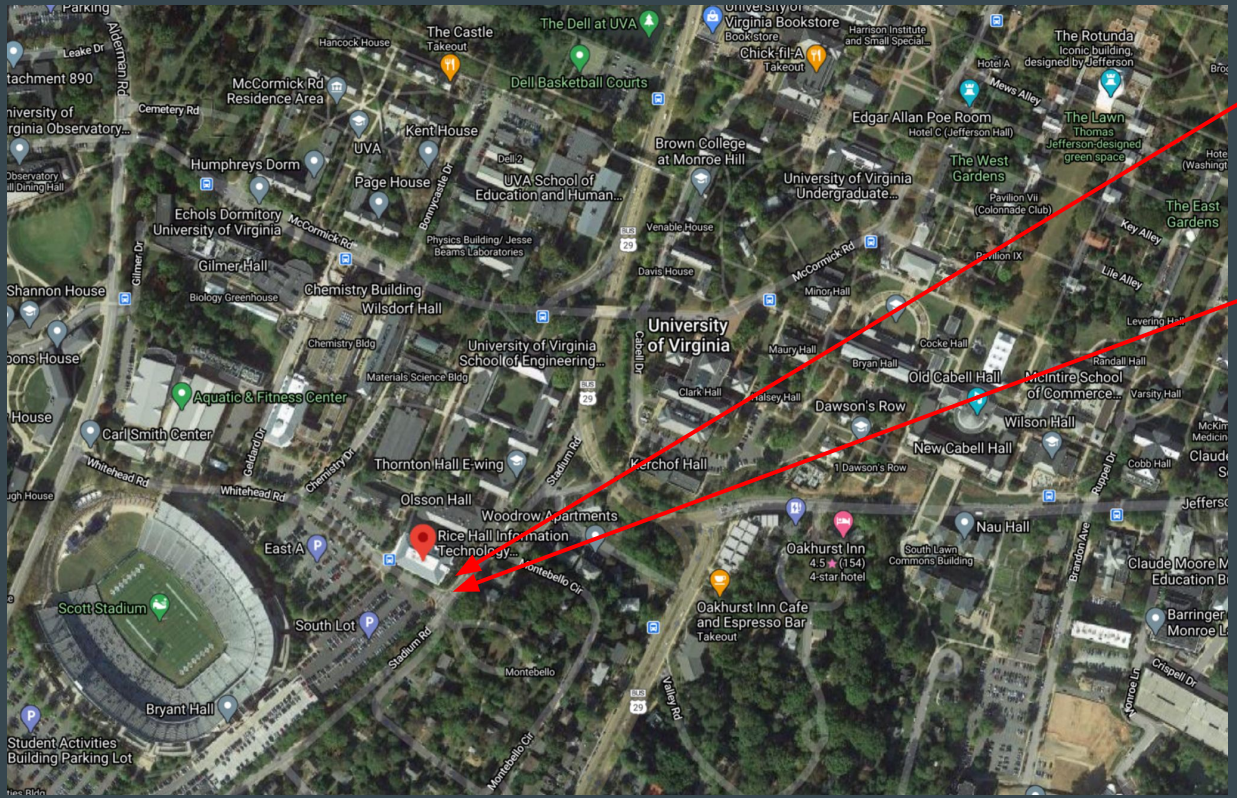
# CS4501
# Robotics for Soft Eng

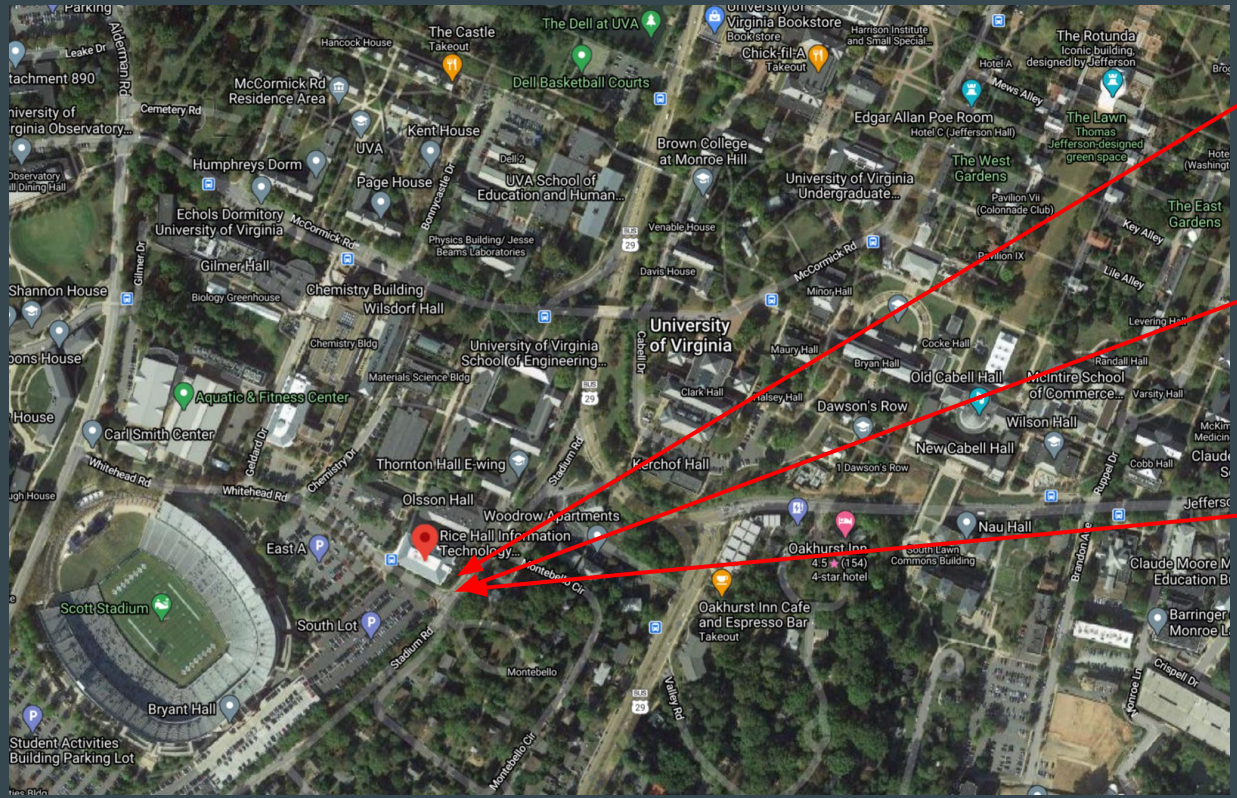
...

Coordinates and Transformations

Tell me, where are you?


Rice Hall Information Technology...
Lat: 38 01’ 52.6” N
Long: 78 30’ 38.7”  W


The Castle
Takeout
The Dell at UVA
Dell Basketball Courts
Chick-fil-A
Takeout
The Rotunda
Iconic building designed by Jefferson
McCormick Rd
Residence Area
Kent House
UVA
Humphreys Dorm
Page House
Brown College at Monroe Hill
Edgar Allan Poe Room
Hotel C (Jefferson Hall)
The Lawn
Thomas Jefferson-designed green space
The West Gardens
University of Virginia Undergraduate...
UVA School of Education and Human...
Echols Dormitory
University of Virginia
Gilmer Hall
Chemistry Building
Wilsdorf Hall
Shannon House
University of Virginia
University of Virginia School of Engineering...
Aquatic & Fitness Center
Carl Smith Center
Thornton Hall E-wing
Olsson Hall
Woodrow Apartments
Rice Hall Information Technology...
East A
Scott Stadium
South Lot
Bryant Hall
Student Activities Building Parking Lot
Oakhurst Inn
4.5 (154)
4-star hotel
Oakhurst Inn Cafe and Espresso Bar
Takeout
Nau Hall
South Lawn Commons Building
Old Cabell Hall
McIntire School of Commerce...
Dawson's Row
Wilson Hall
New Cabell Hall
Kerchof Hall
The East Gardens
Claude Moore Education B...
Barringer Monroe L...
Jefferson
Lat: 38.03178779534993
Long: -78.5108566305418
Lat: 38 01’ 52.6” N
Long: 78 30’ 38.7” W
Two Coordinate Systems
Latitude (North/South)
90°N
45°N
0°
Equator
45°S
90°S
Latitude varies from 0° at the equator to 90° North and South at the poles
Longitude (West/East)
W
E
Longitude varies from 0° at Greenwich to 180° East and West
https://en.wikipedia.org/wiki/Geographic_coordinate_system

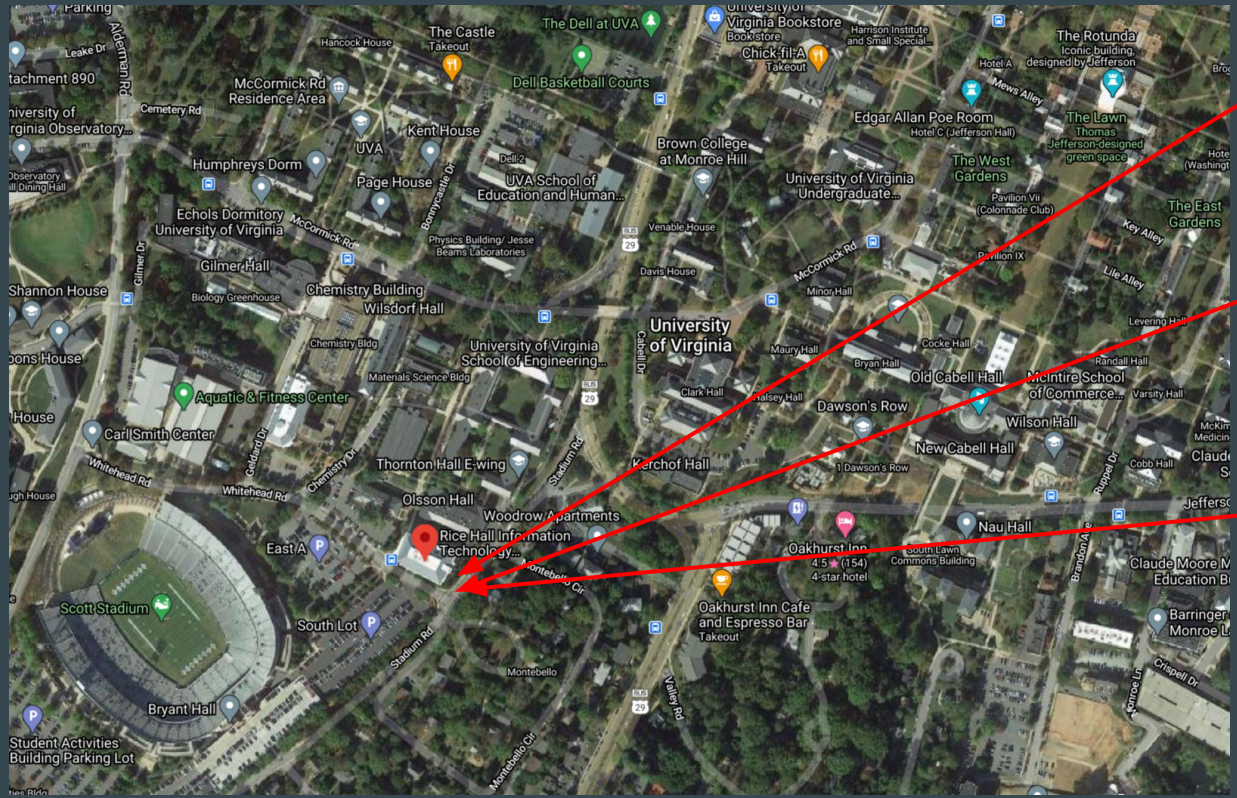



Lat: 38.03178779534993
Long: -78.5108566305418

Lat: 38 01’ 52.6” N
Long: 78 30’ 38.7” W

Plus Codes

2FJQ+JM




Lat: 38.03178779534993
Long: -78.5108566305418

Lat: 38 01’ 52.6” N
Long: 78 30’ 38.7”  W

Plus Codes

2FJQ+JM

Missing?

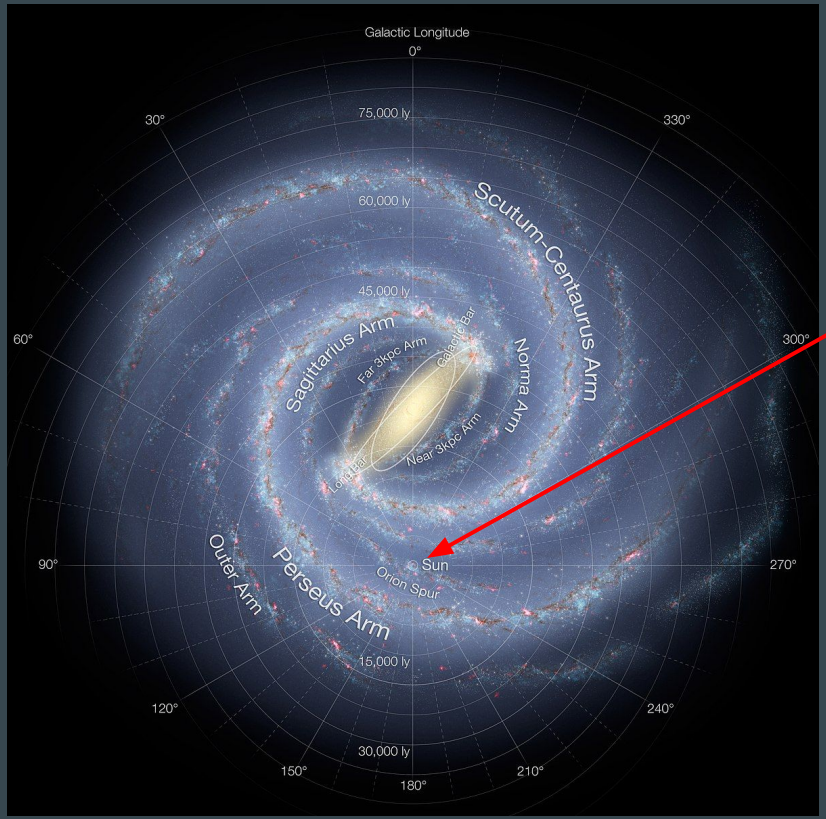
# ROS Support

- **Specialized Message types**

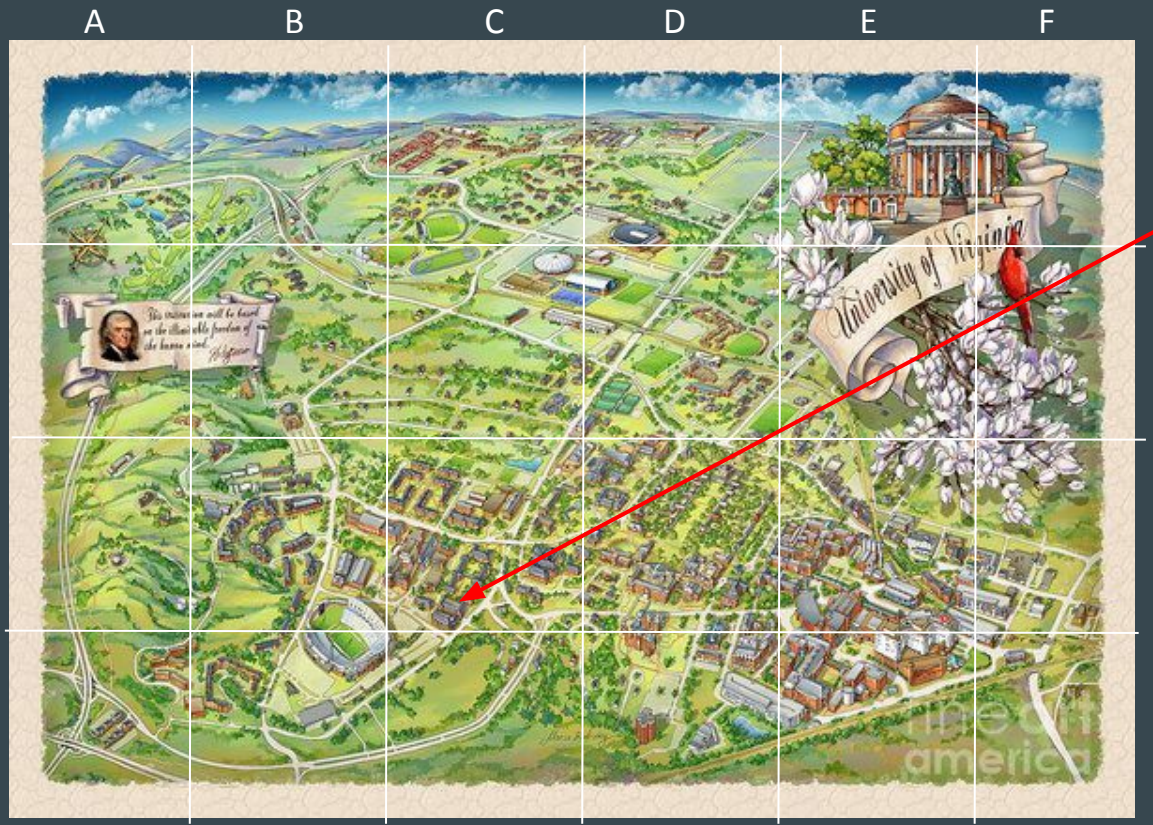
  **sensor_msgs/NavSatFix.msg**

```
std_msgs/Header header
sensor_msgs/NavSatStatus status
float64 latitude
float64 longitude
float64 altitude
float64[9] position_covariance
uint8 position_covariance_type
```

```
# Navigation Satellite fix status for any Global Navigation Satellite System

# Whether to output an augmented fix is determined by both the fix
# type and the last time differential corrections were received.  A
# fix is valid when status >= STATUS_FIX.

int8 STATUS_NO_FIX =  -1        # unable to fix position
int8 STATUS_FIX =      0        # unaugmented fix
int8 STATUS_SBAS_FIX = 1        # with satellite-based augmentation
int8 STATUS_GBAS_FIX = 2        # with ground-based augmentation

int8 status

# Bits defining which Global Navigation Satellite System signals were
# used by the receiver.

uint16 SERVICE_GPS =     1
uint16 SERVICE_GLONASS = 2
uint16 SERVICE_COMPASS = 4      # includes BeiDou.
uint16 SERVICE_GALILEO = 8

uint16 service
```

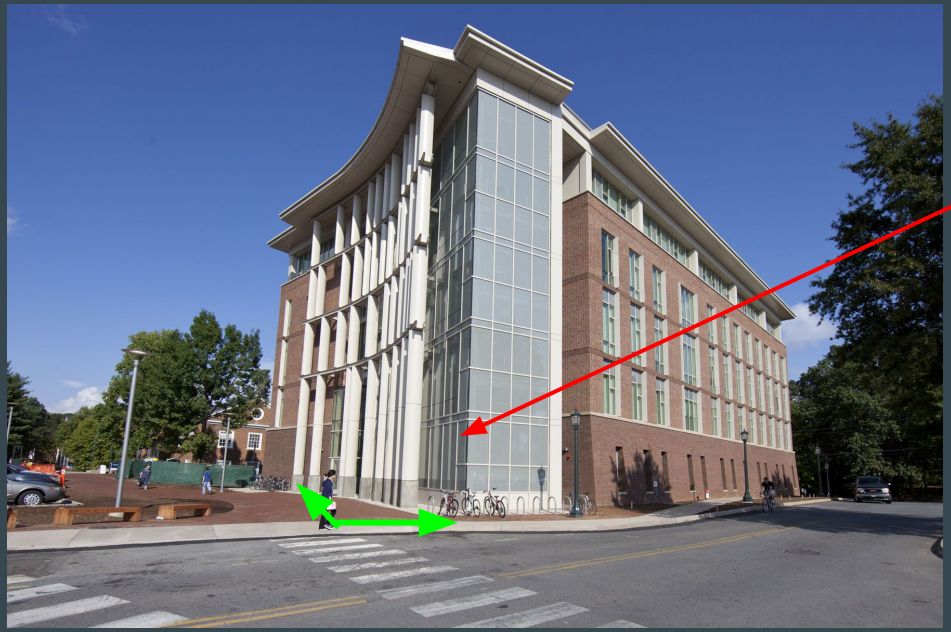



Another Coordinate System



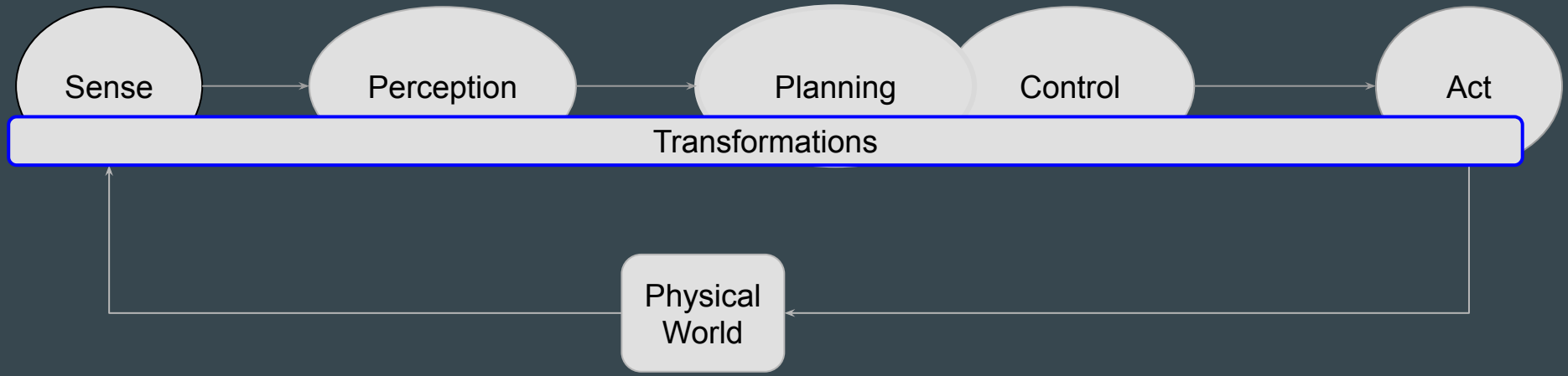

## Another Coordinate System

B3



## Yet Another Coordinate System

10 yards, 45 degrees


Sense
Perception
Planning
Control
Act
Transformations
Physical World

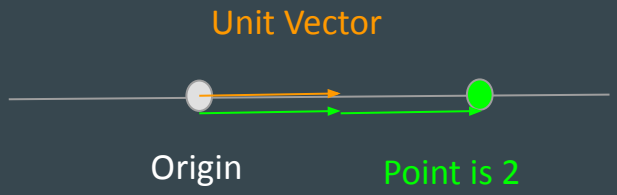
# System of coordinates

- Method to associate unique numbers to a point
- Requires
  - Origin
  - Basis unit vector (positive)

# System of coordinates

- Method to associate unique numbers to a point
- Requires
  - Origin
  - Basis unit vector (positive)

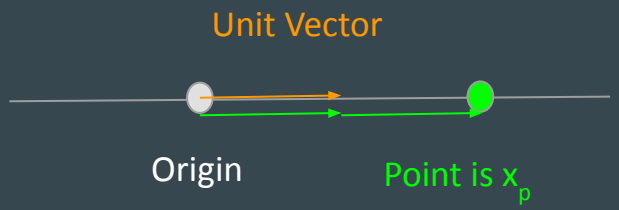
1D System - point in a line



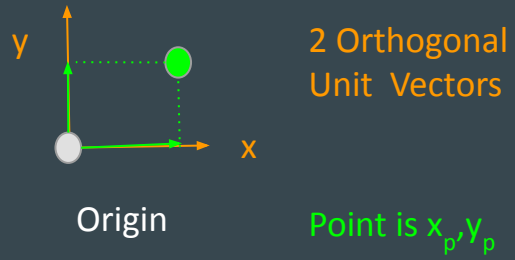
# System of coordinates

- Method to associate unique numbers to a point
- Requires
  - Origin
  - Basis unit vector (positive)

1D System - point in a line



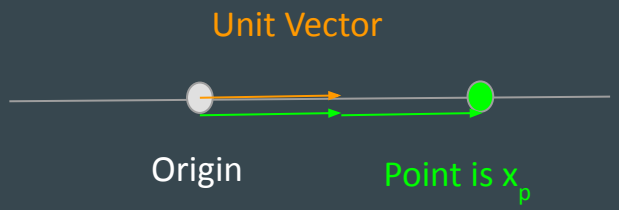

2D System - point in a plane



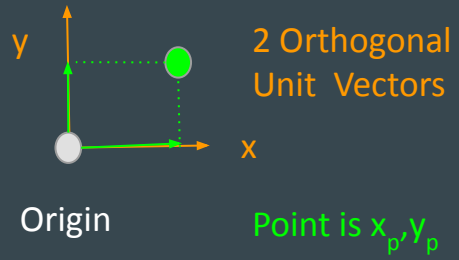
# System of coordinates

- Method to associate unique numbers to a point
- Requires
  - Origin
  - Basis unit vector (positive)

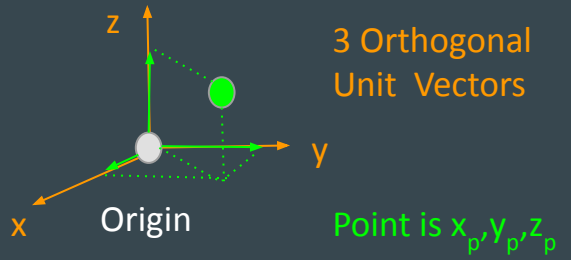
1D System - point in a line




2D System - point in a plane



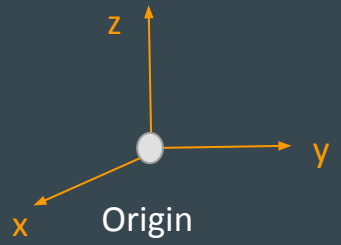

3D System - point in a space

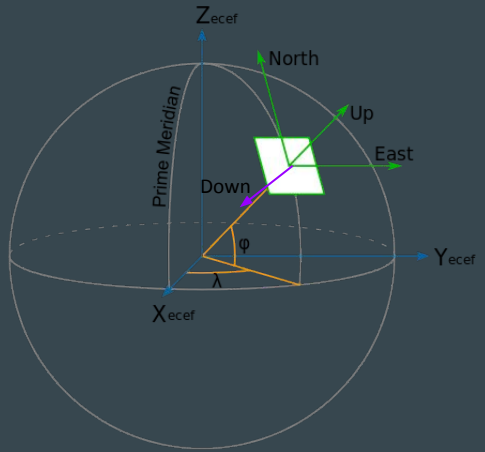


# Multiple Coordinate Systems

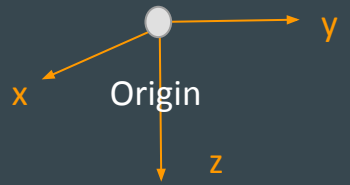
- 3D World reference frames
- Multiple conventions

ENU - East, North, UP

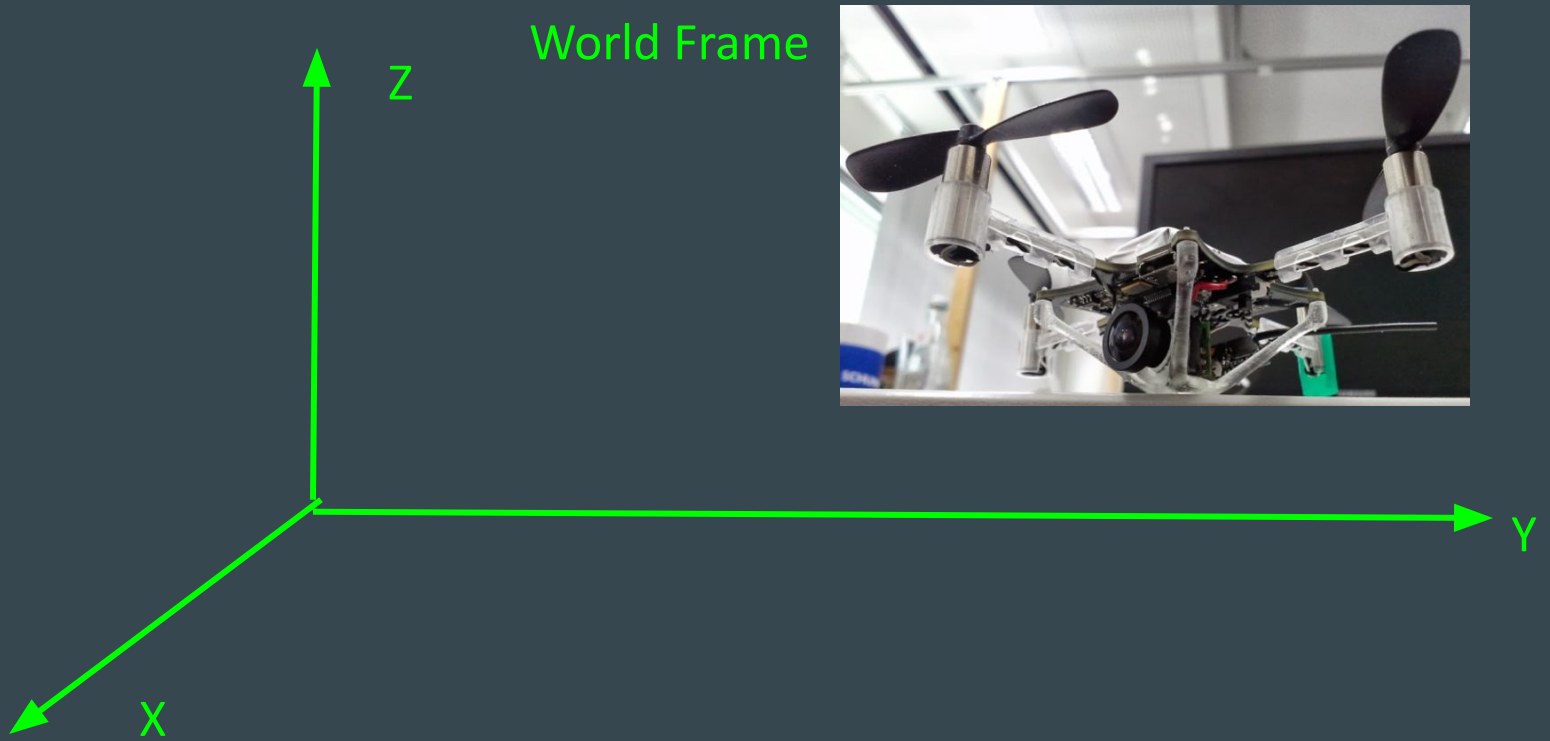


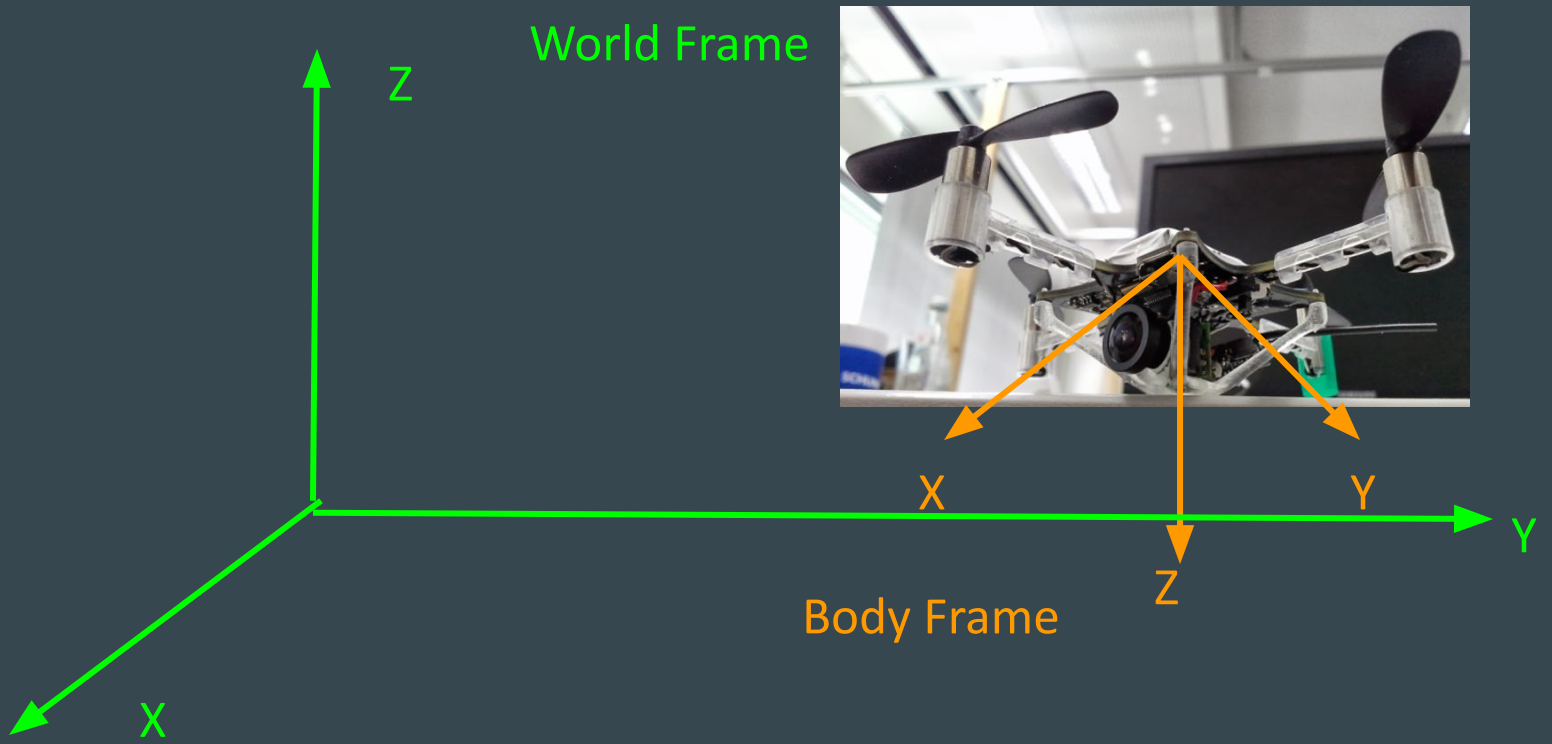

NED - North, East, Down



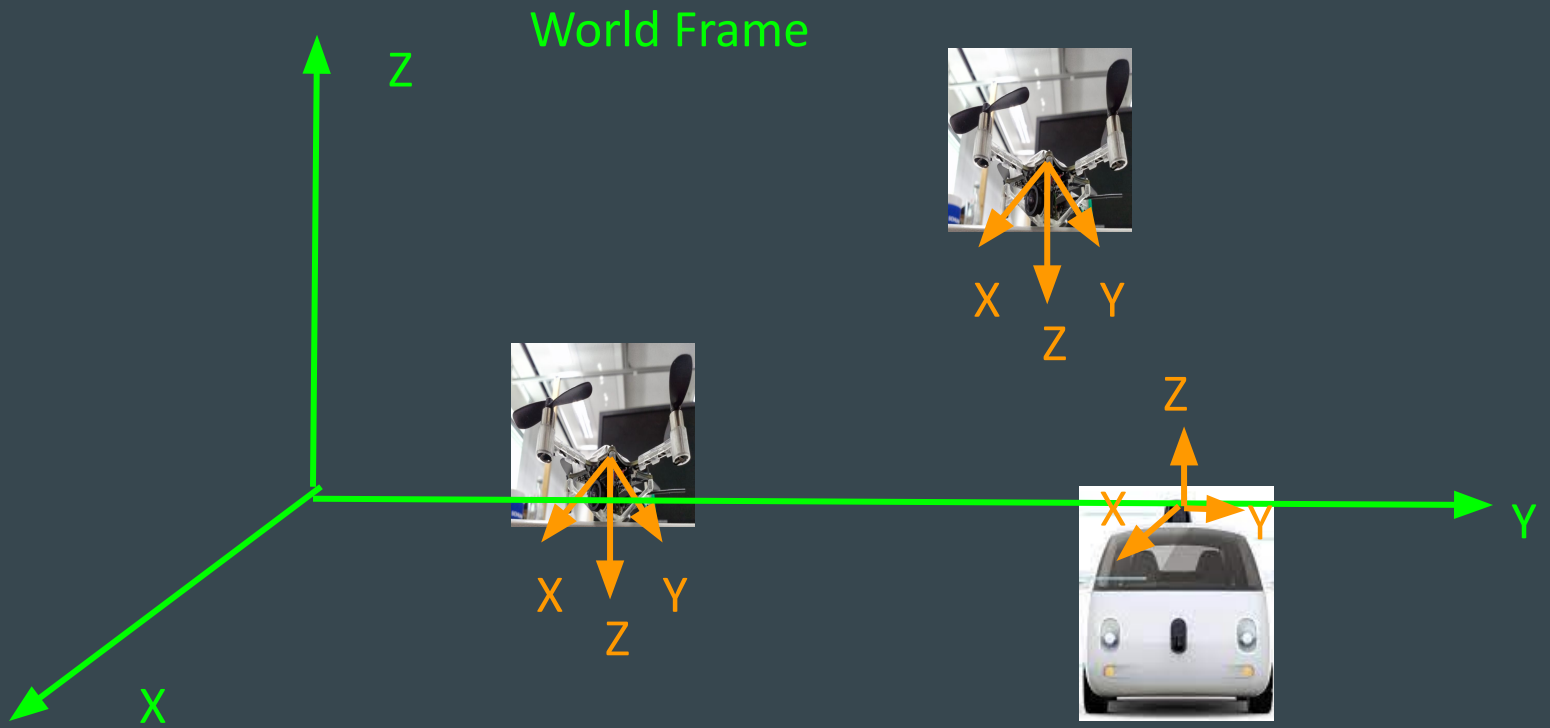
# Multiple Coordinate Systems



# Multiple Coordinate Systems



# Multiple Coordinate Systems



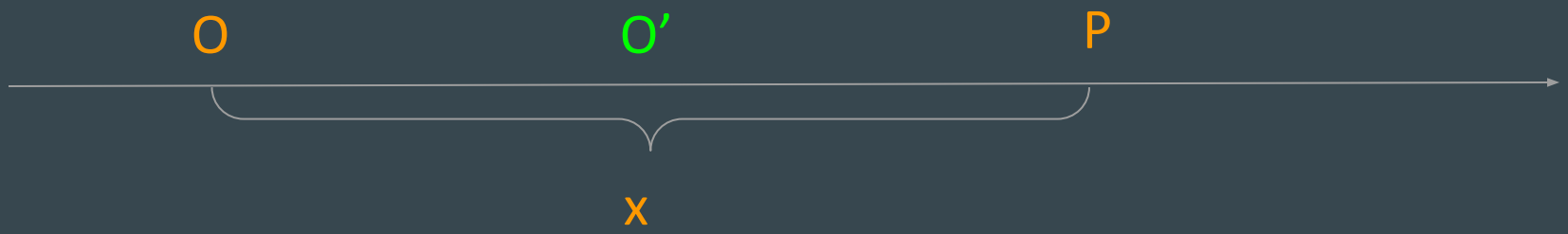
# Transform

- Function
  - Input: point/vector P in Frame A, target Frame B
  - Output:  point/vector P in Frame B
- Pseudocode
  - Translate
  - Rotate (trigonometry)

# 1D Transform

# 1D Transform

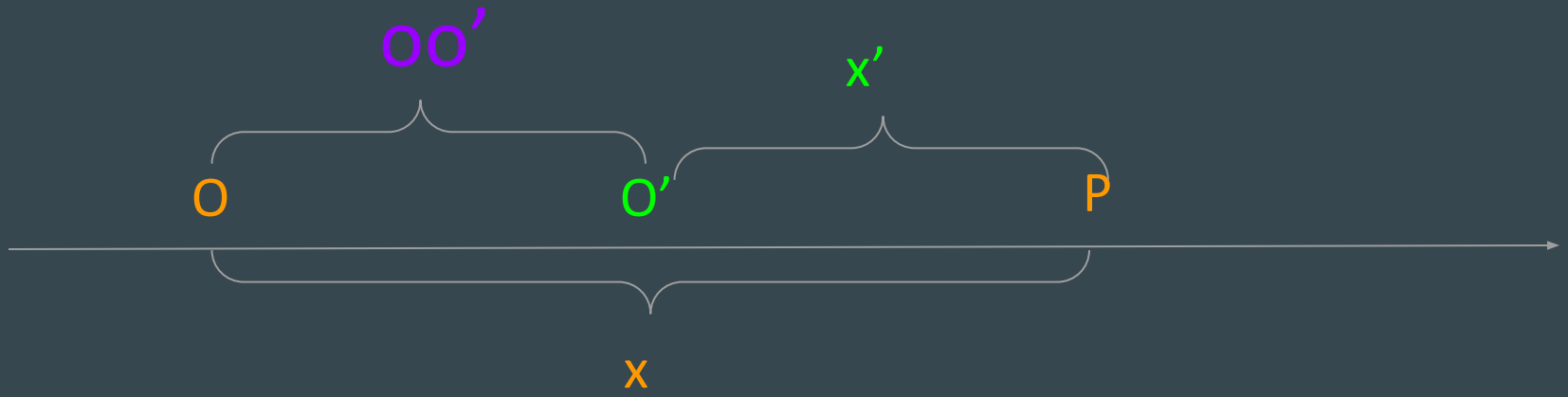
Where is P in O’?



# 1D Transform

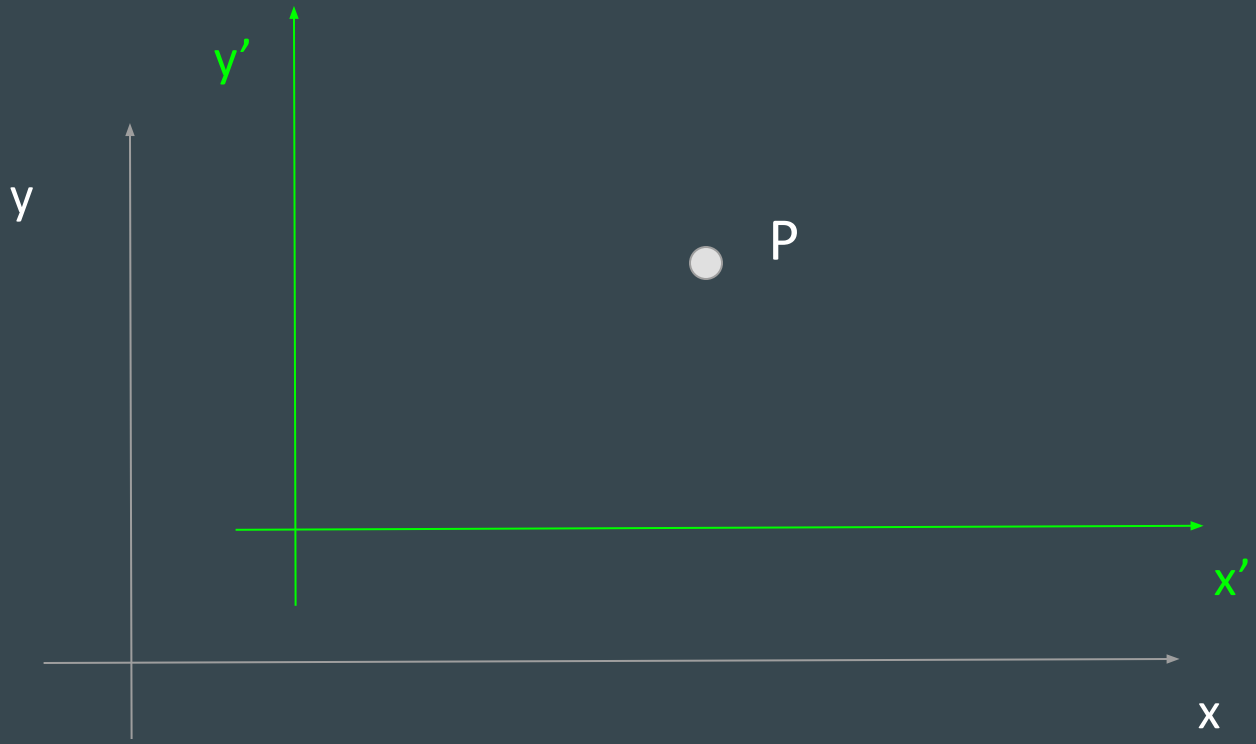
Where is P in O’?

$x' = x - oo'$



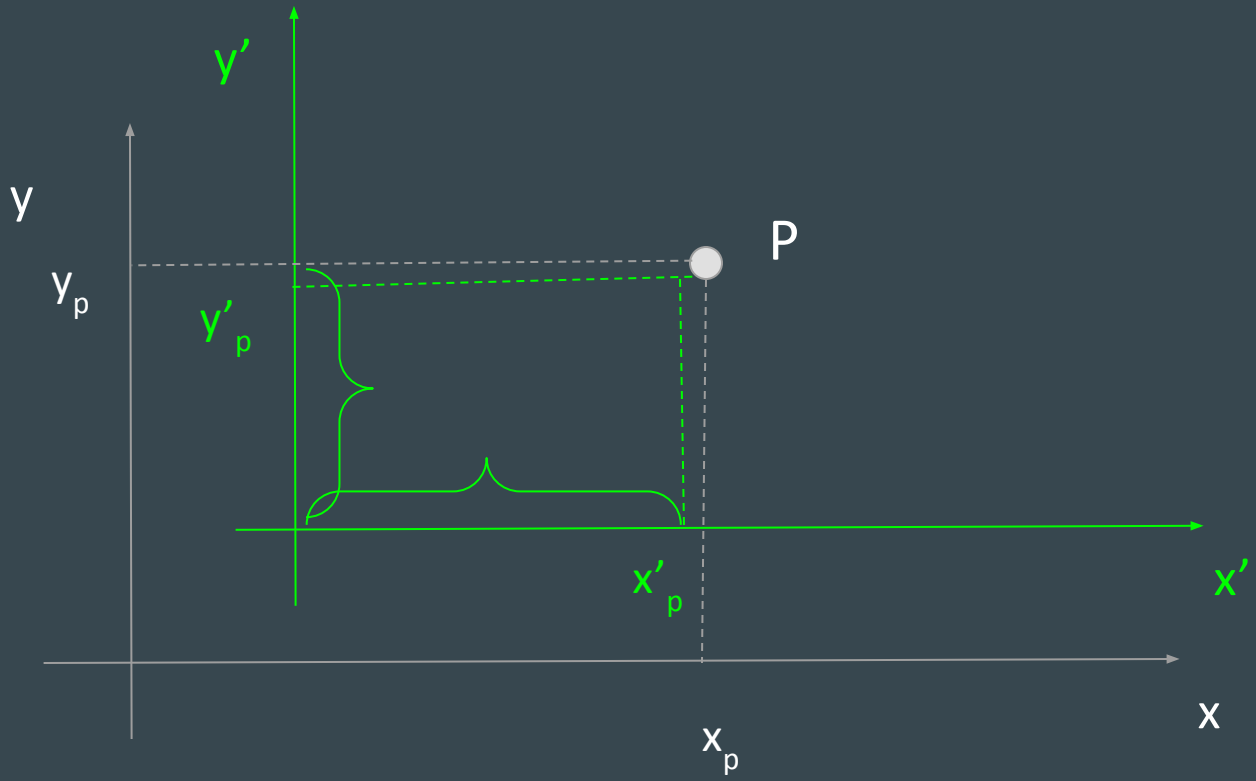
# 2D Transform - translation

Where is P in O’?



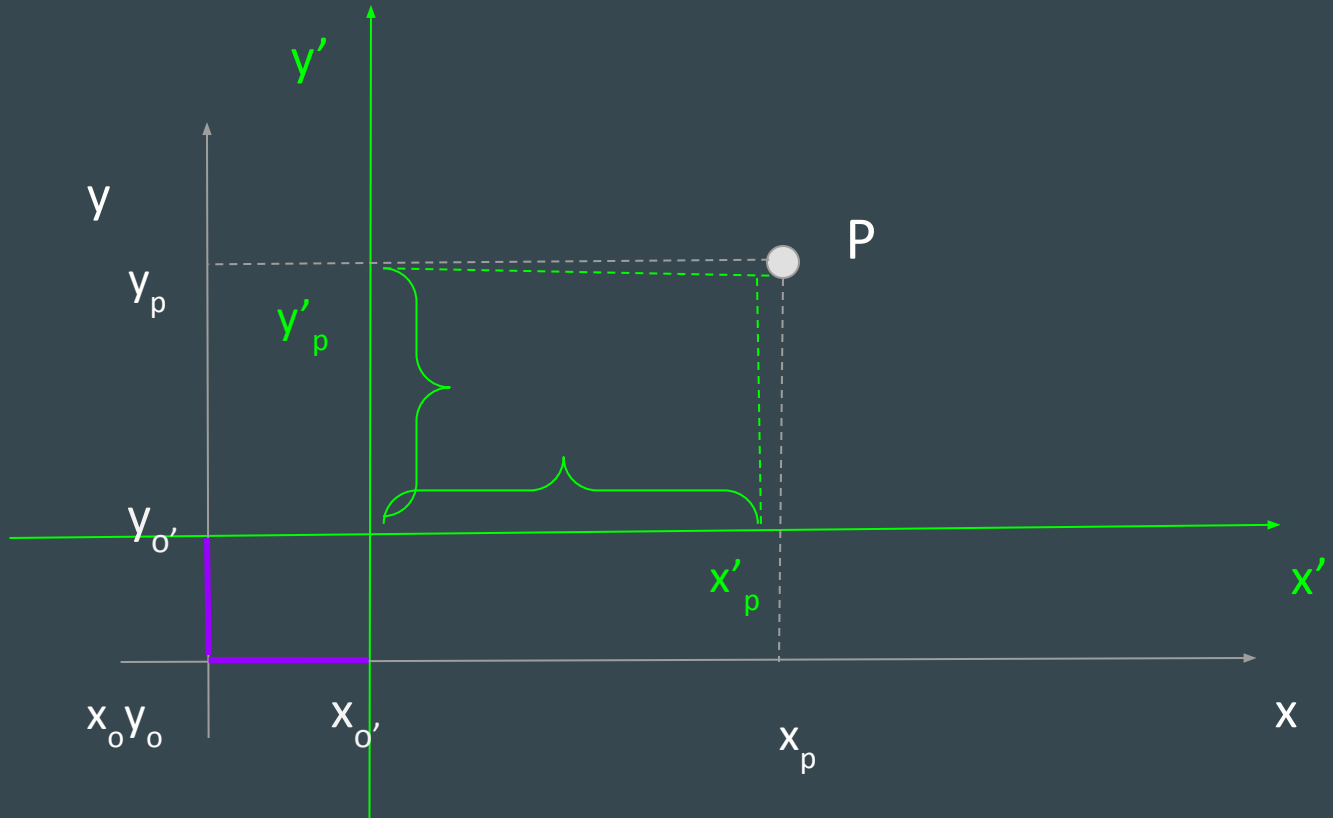
# 2D Transform - translation

Where is P in O’?



# 2D Transform - translation

Where is P in O'?

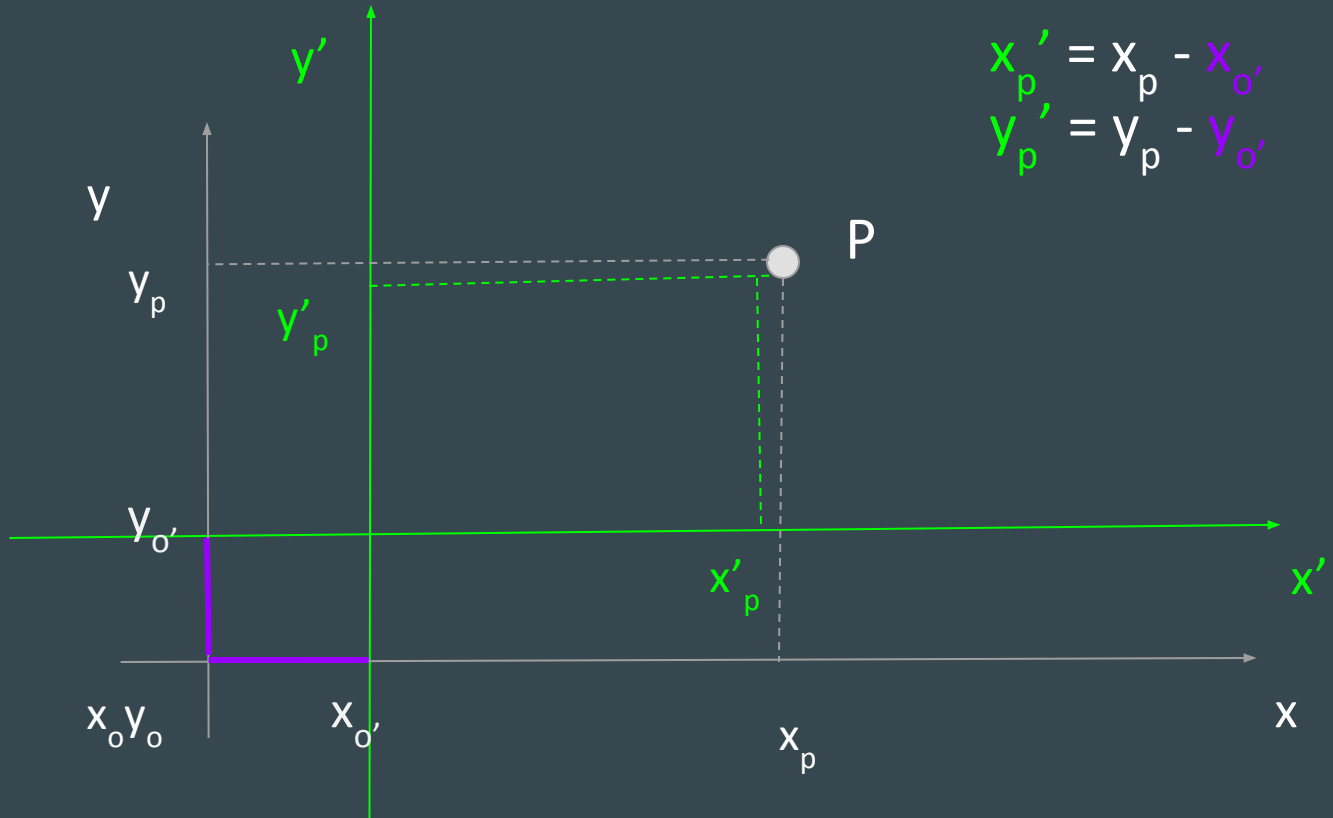


# 2D Transform - translation

Where is P in O'?

$x_p' = x_p - x_{o'}$

$y_p' = y_p - y_{o'}$



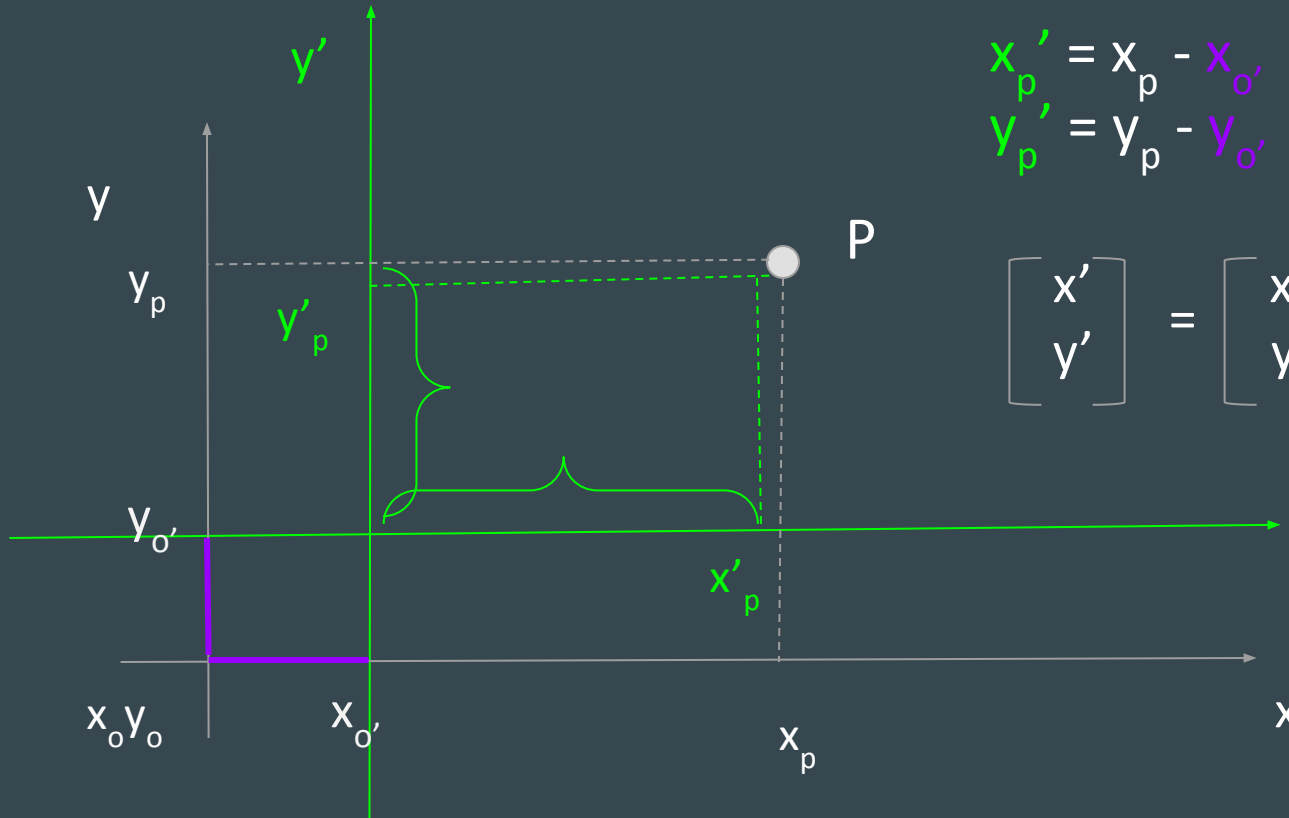
# 2D Transform - translation

Where is P in O'?

$x_p' = x_p - x_{o'}$

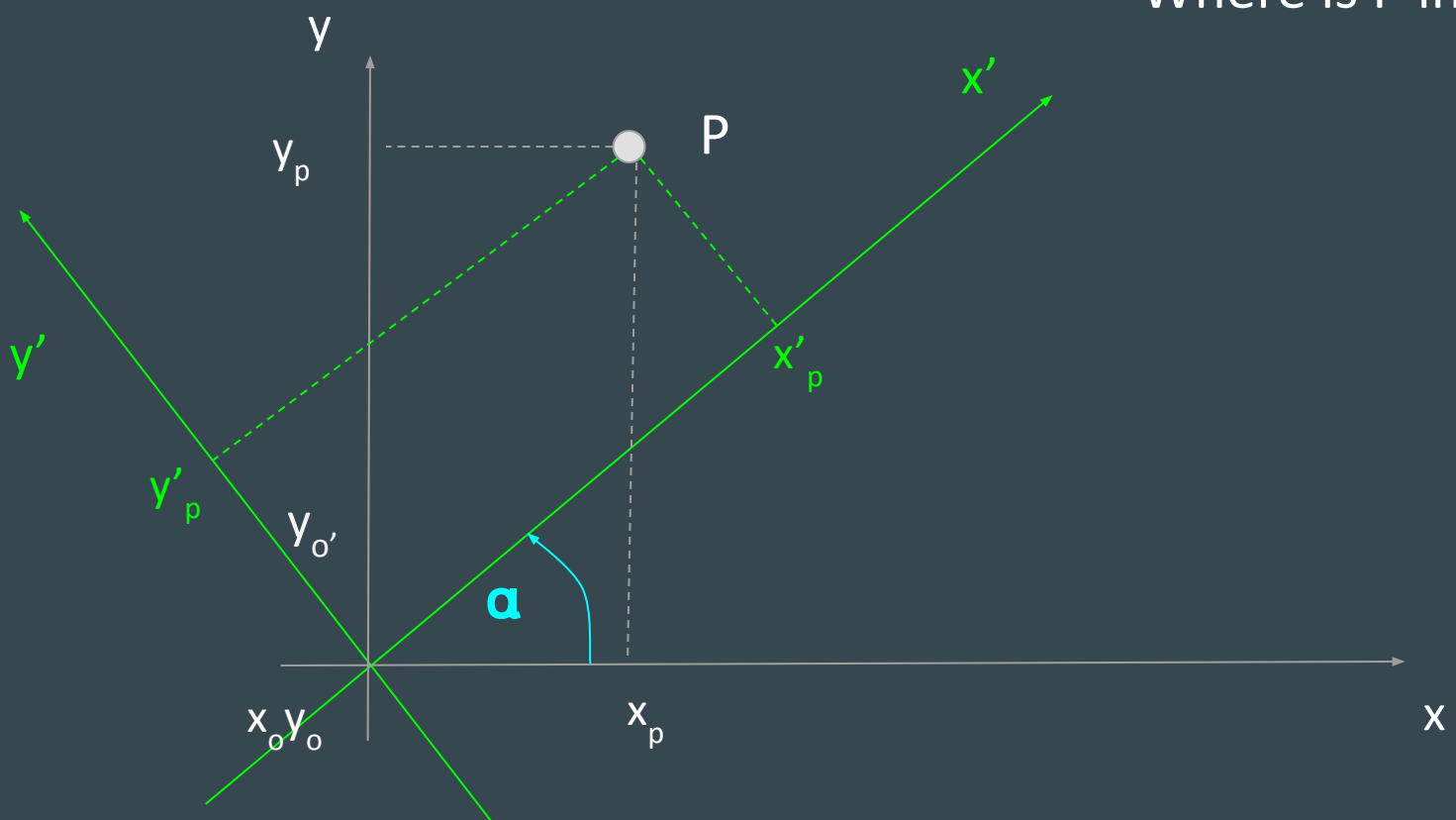
$y_p' = y_p - y_{o'}$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_{0'} \\ y_{0'} \end{bmatrix}$$

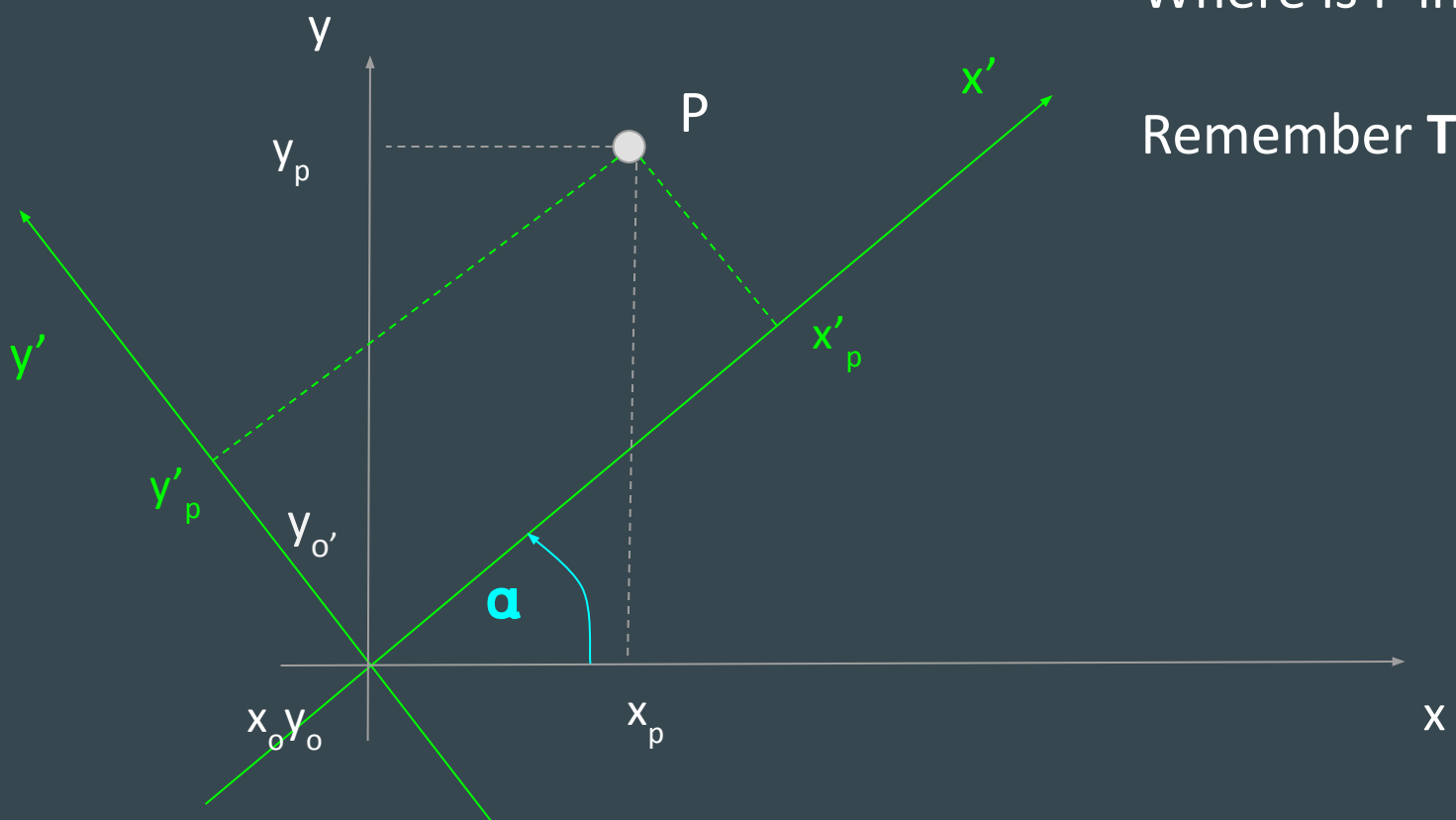


# 2D Transform - rotation

Where is P in O?



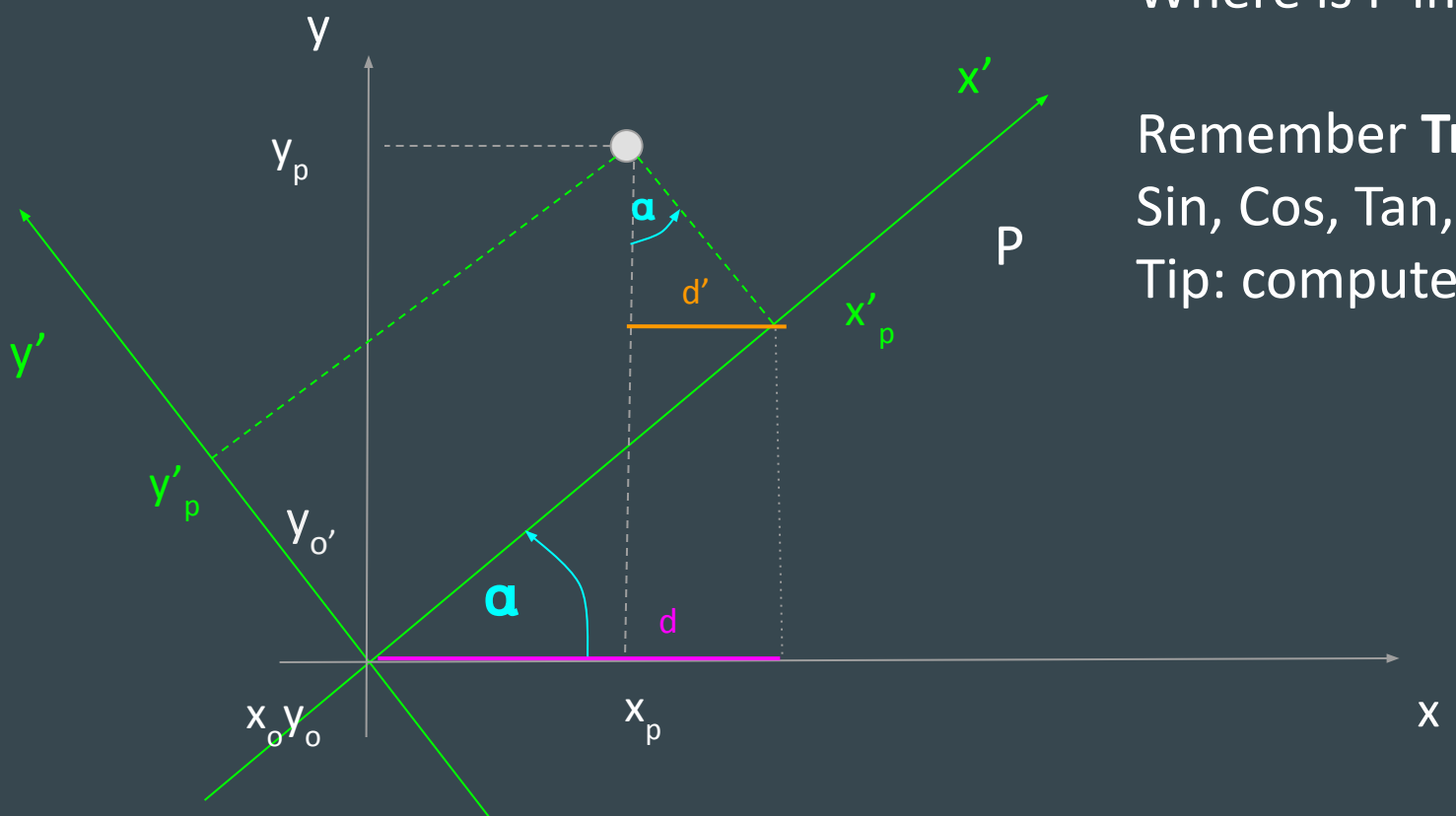
# 2D Transform - rotation




Where is P in O?

Remember **Trig**?

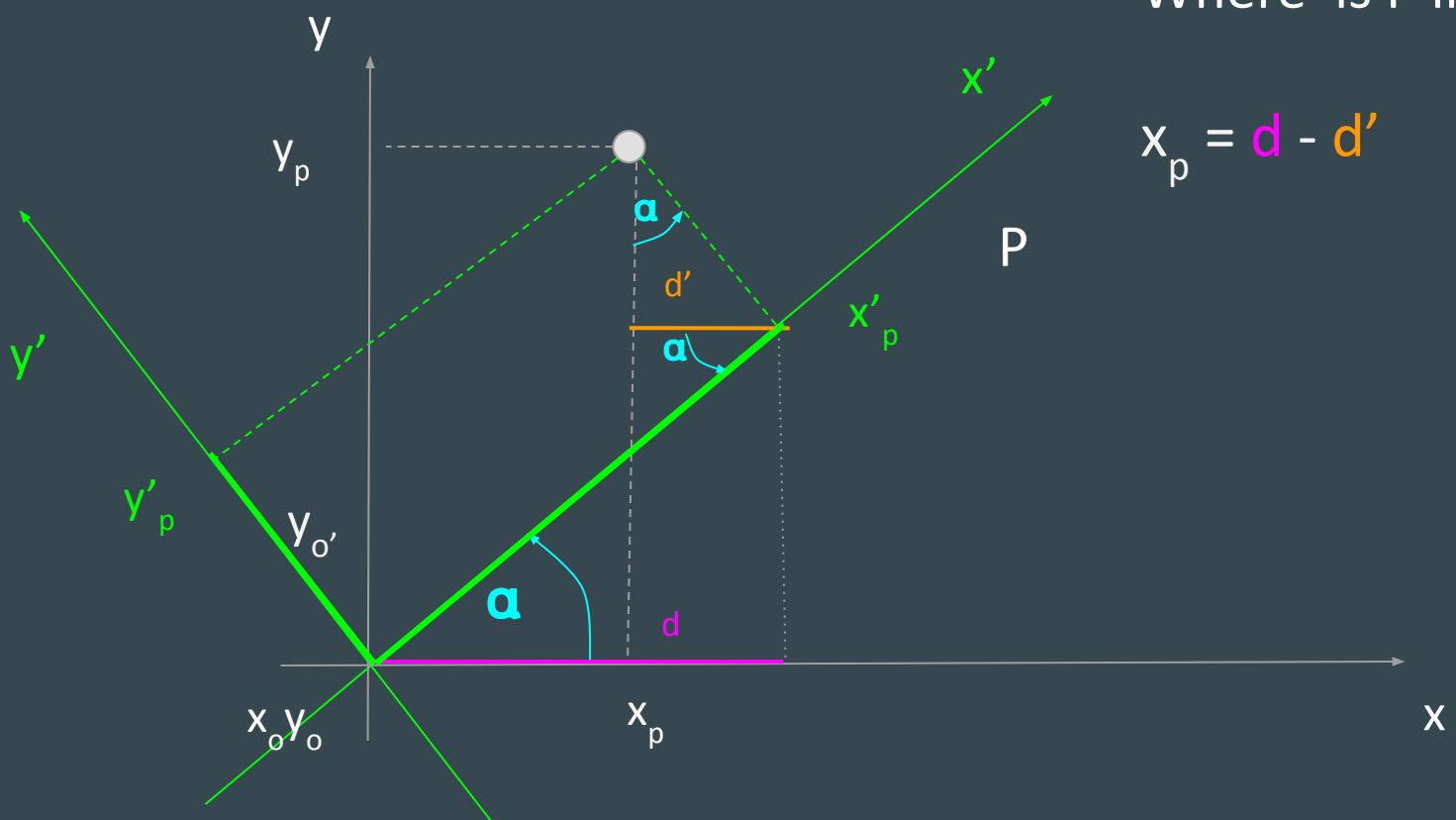
# 2D Transform - rotation




Where is P in O?

Remember **Trig**?
Sin, Cos, Tan, ...
Tip: compute d and d'

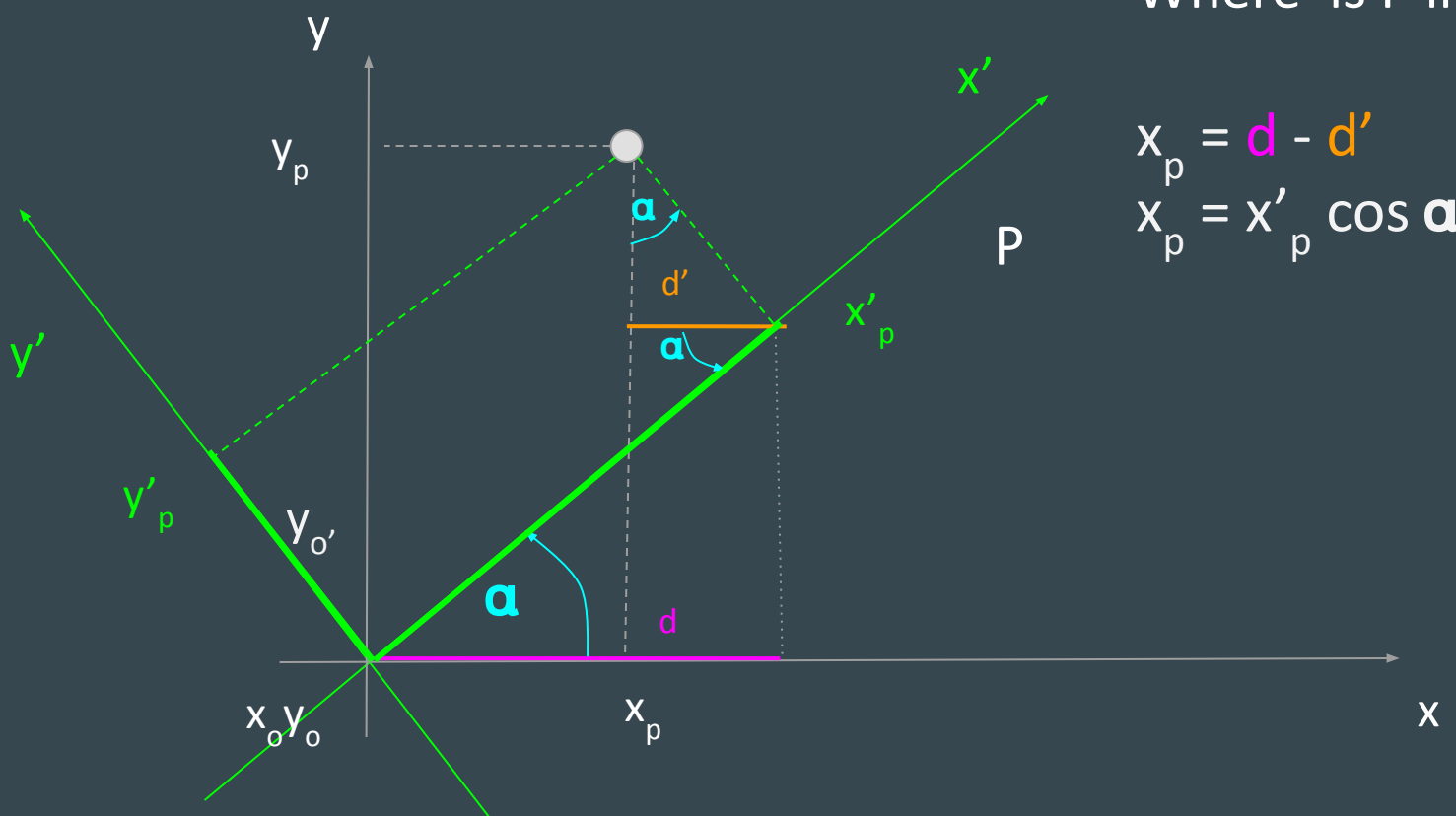
# 2D Transform - rotation




Where is P in O?

$x_p = d - d'$

# 2D Transform - rotation

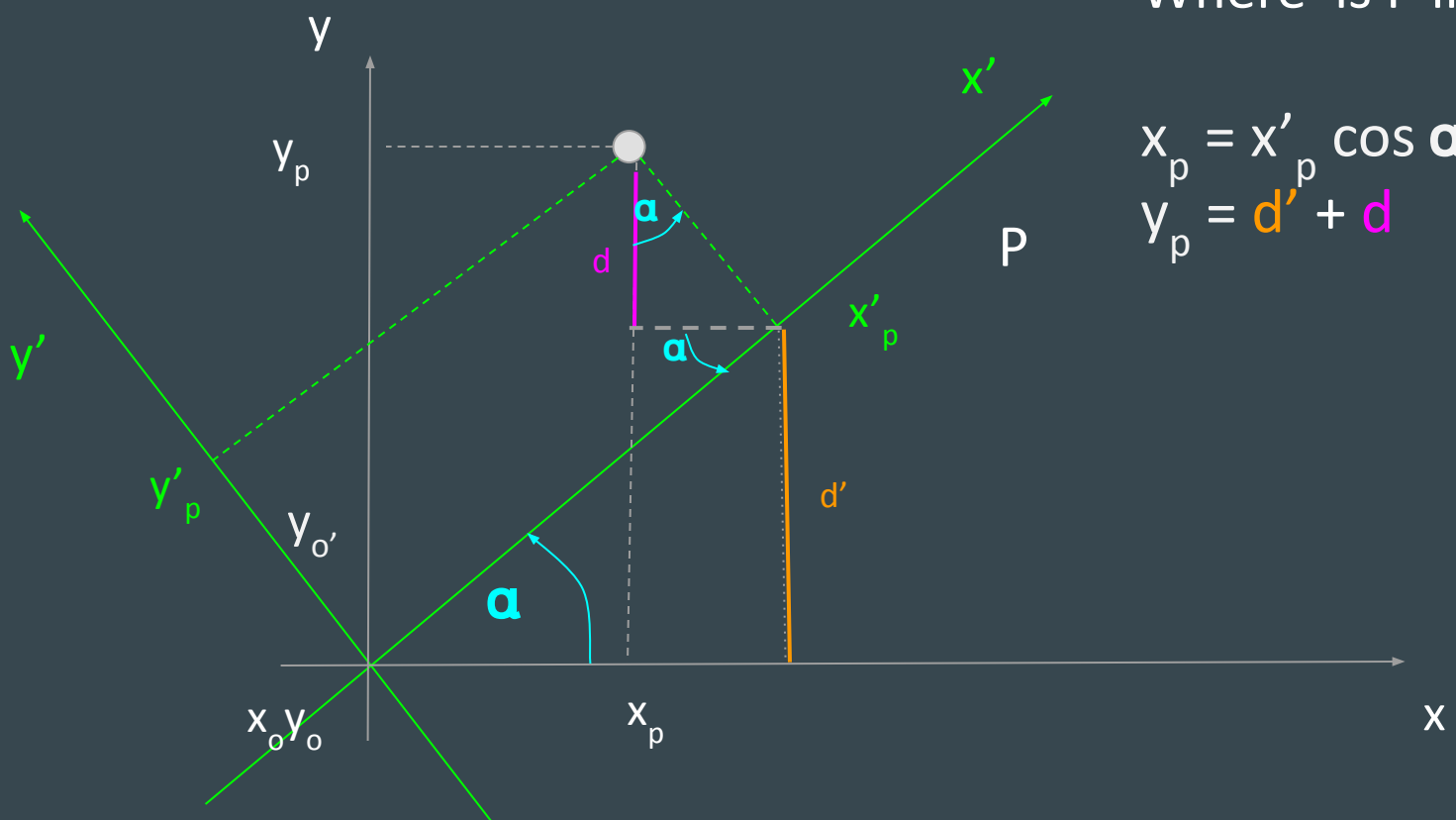



Where is P in O?

$x_p = d - d'$

$x_p = x'_p \cos \alpha - y'_p \sin \alpha$

# 2D Transform - rotation

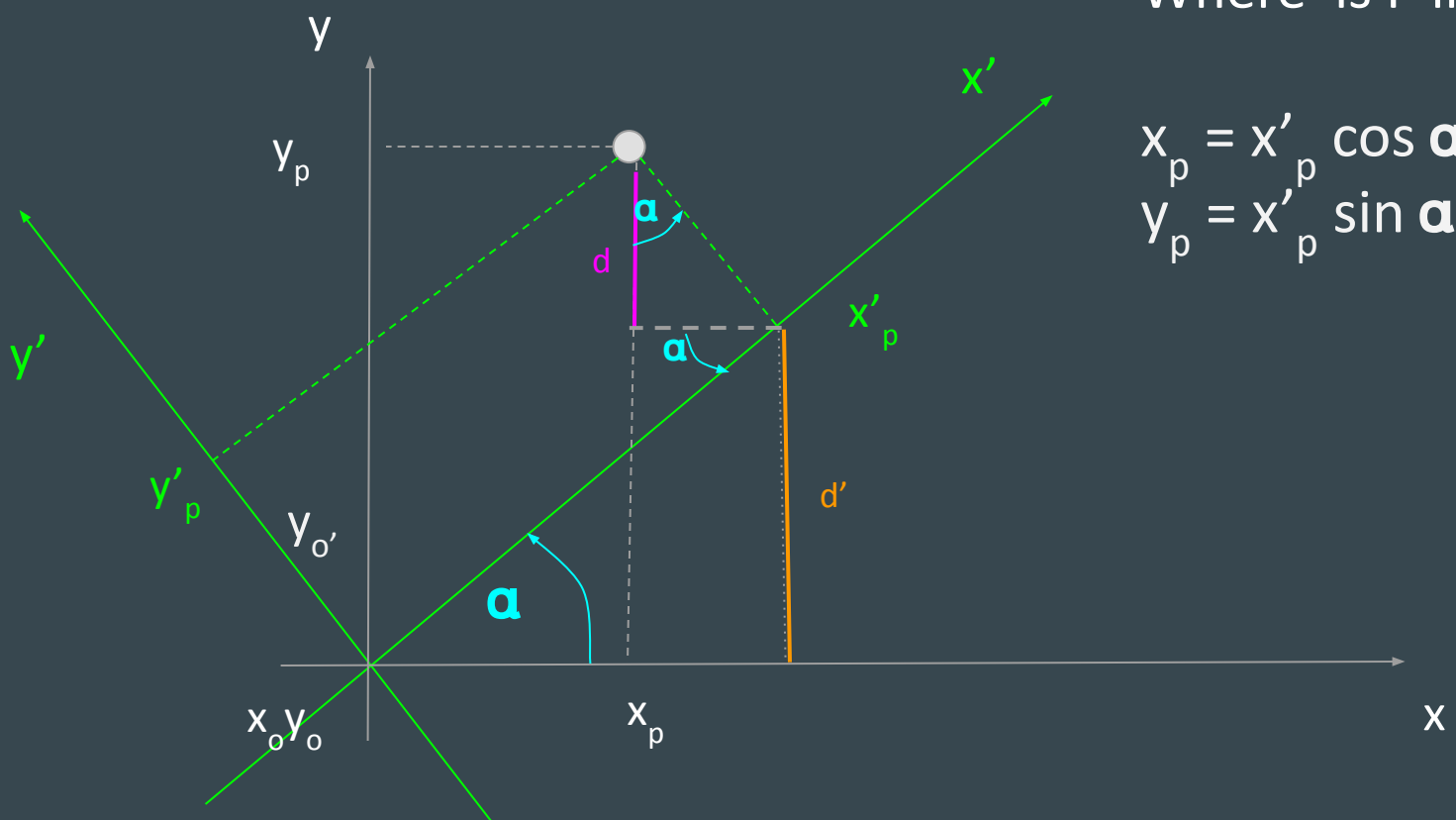



Where is P in O?

$x_p = x'_p \cos \alpha - y'_p \sin \alpha$

$y_p = d' + d$

# 2D Transform - rotation

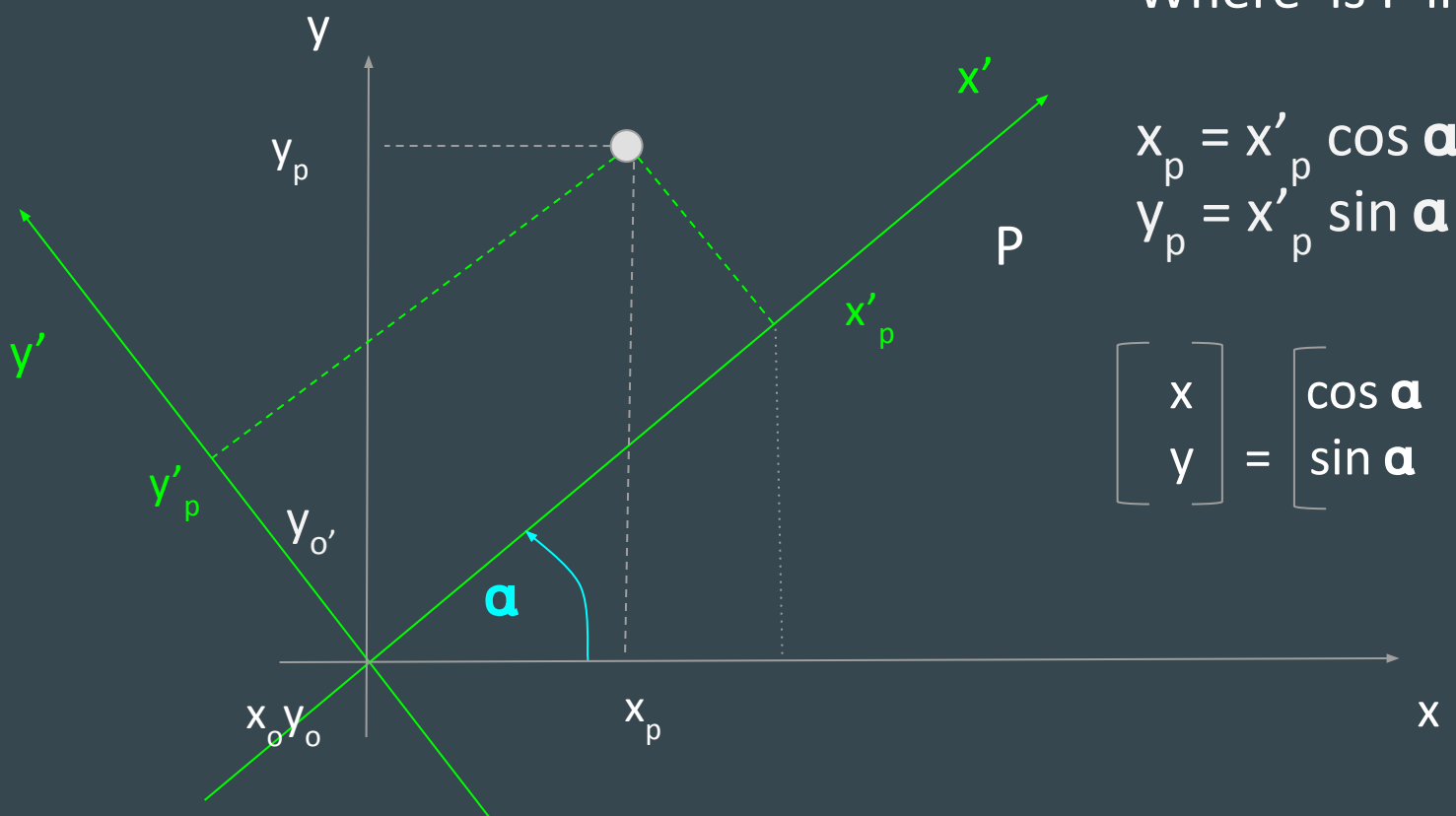



Where is P in O?

$x_p = x'_p \cos \alpha - y'_p \sin \alpha$

$y_p = x'_p \sin \alpha + y'_p \cos \alpha$

# 2D Transform - rotation

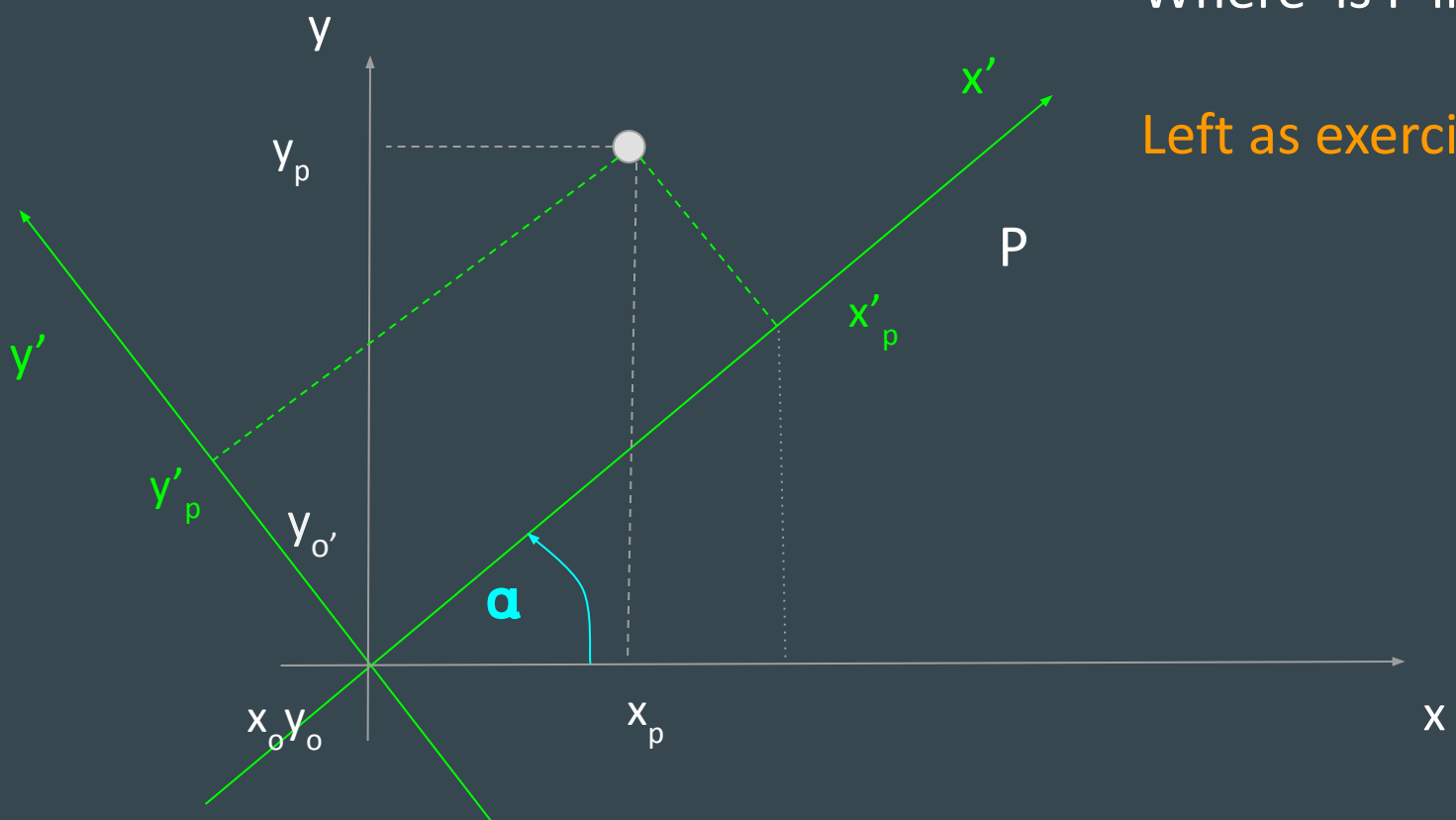



Where is P in O?

$$x_p = x'_p \cos \alpha - y'_p \sin \alpha$$
$$y_p = x'_p \sin \alpha + y'_p \cos \alpha$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix}$$

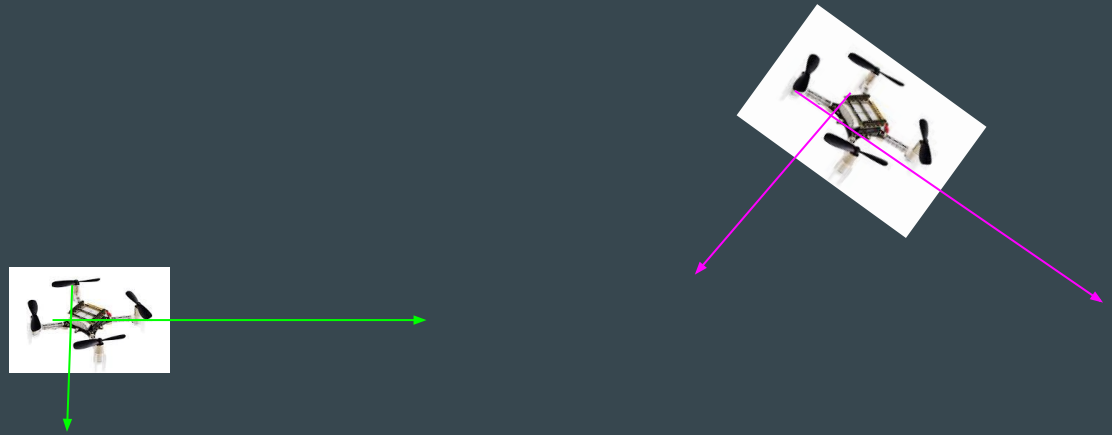
# 2D Transform - rotation



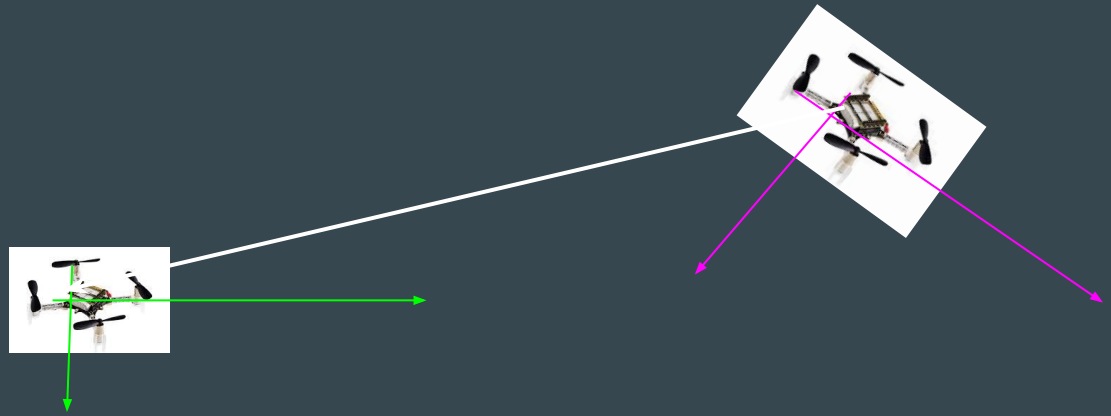

Where is P in O'?

Left as exercise

# Full 2D Transform: Translation then Rotation

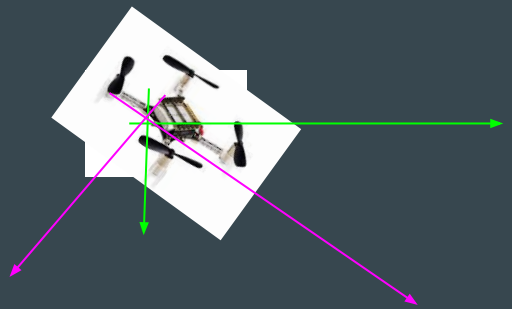


# Full 2D Transform: Translation then Rotation



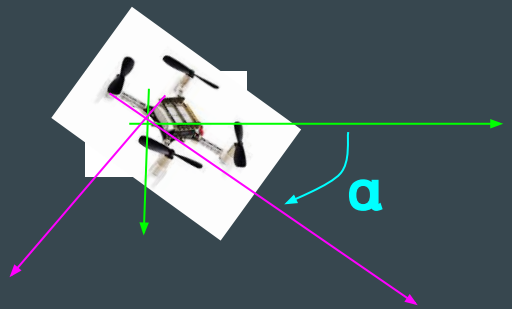
Compute Translation from O to O’

# Full 2D Transform: Translation then Rotation



Translate and now O = O’

# Full 2D Transform: Translation then Rotation



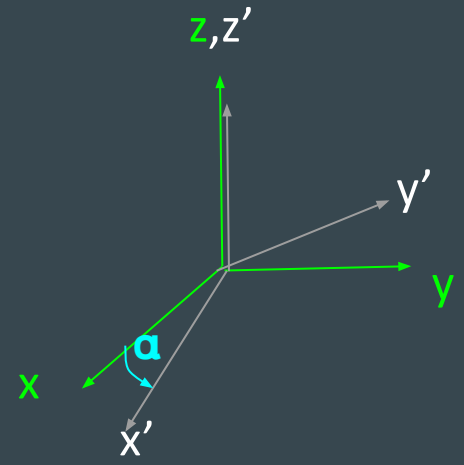

Compute Rotation based on α to have axis aligned

# Full 2D Transform: Translation then Rotation

Transformation complete: same origins, same axis

# 3D Transform: Rotation

Rotation around Z



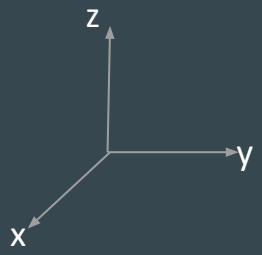

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

same z and z’, so all is z except for last element

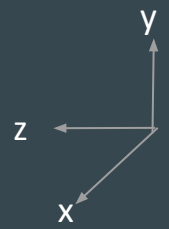
# 3D Transform: Rotation

- Rotation around X,Y,Z
  - Composition of rotations
  - Multiplication of matrices is non-commutative
  - Must agree on the order

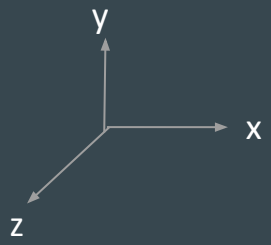
90 degrees
counter-clockwise




Rx



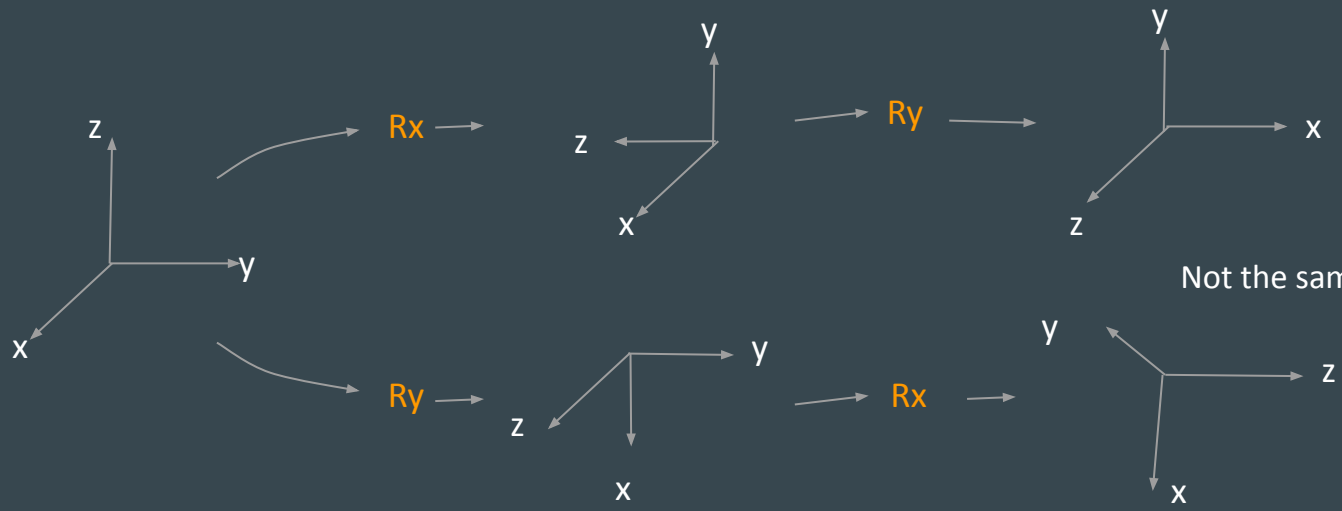

Ry



# 3D Transform: Rotation

- Rotation around X,Y,Z
  - Composition of rotations
  - Multiplication of matrices is non-commutative
  - Must agree on the order

90 degrees
counter-clockwise

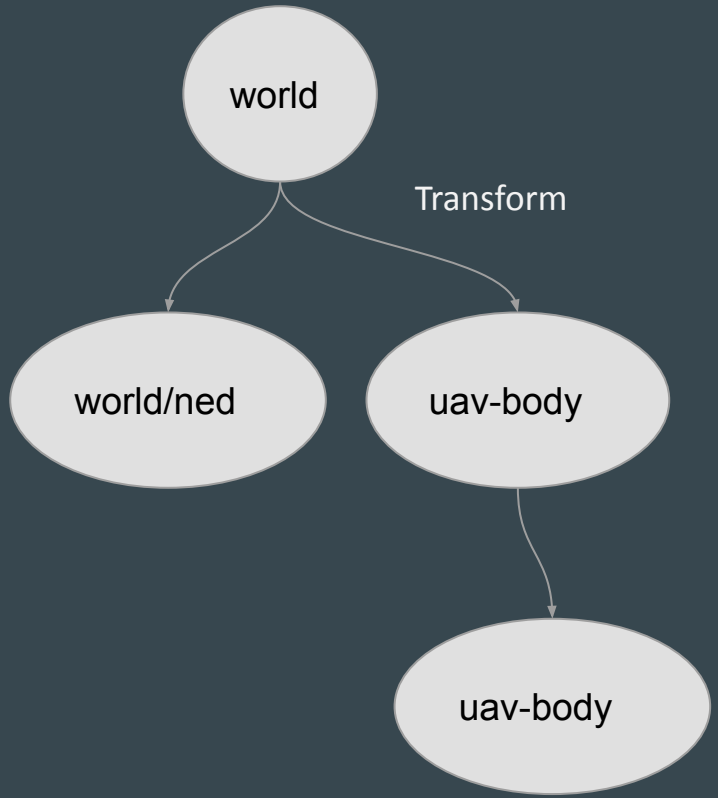


# Transform

- Function
  - Input: point/vector P in Frame A, target Frame B
    - Pose, Velocity, Acceleration
  - Output:  point/vector P in Frame B
    - Pose, Velocity, Acceleration
- Pseudocode
  - Translate
  - Rotate (trigonometry)

# Frames in ROS

- Tf API
- Support for definition and management of frames and transforms across a system
- Frames and transforms organized as a Tree
  - Define a transform (between parent and child)
    - Static
    - Dynamic
  - Publish a transform
  - Lookup transform
  - Listen for a transform
- Tf utilities



Physical data without a
Coordinate Frame
is meaningless

Frame is part of the physical data Type