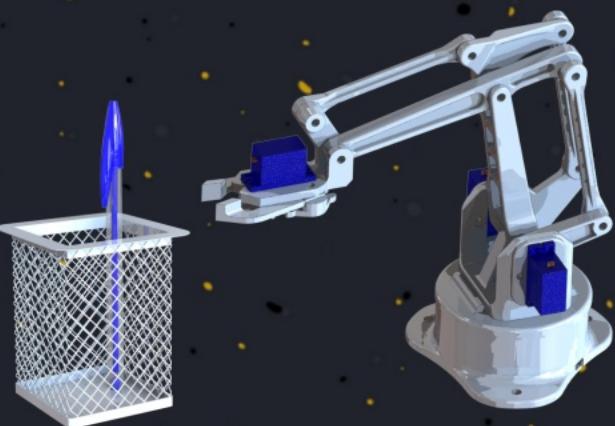
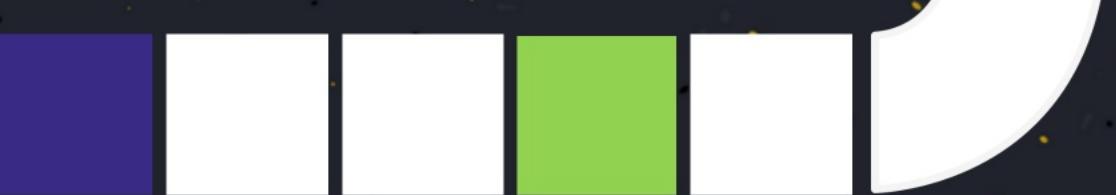
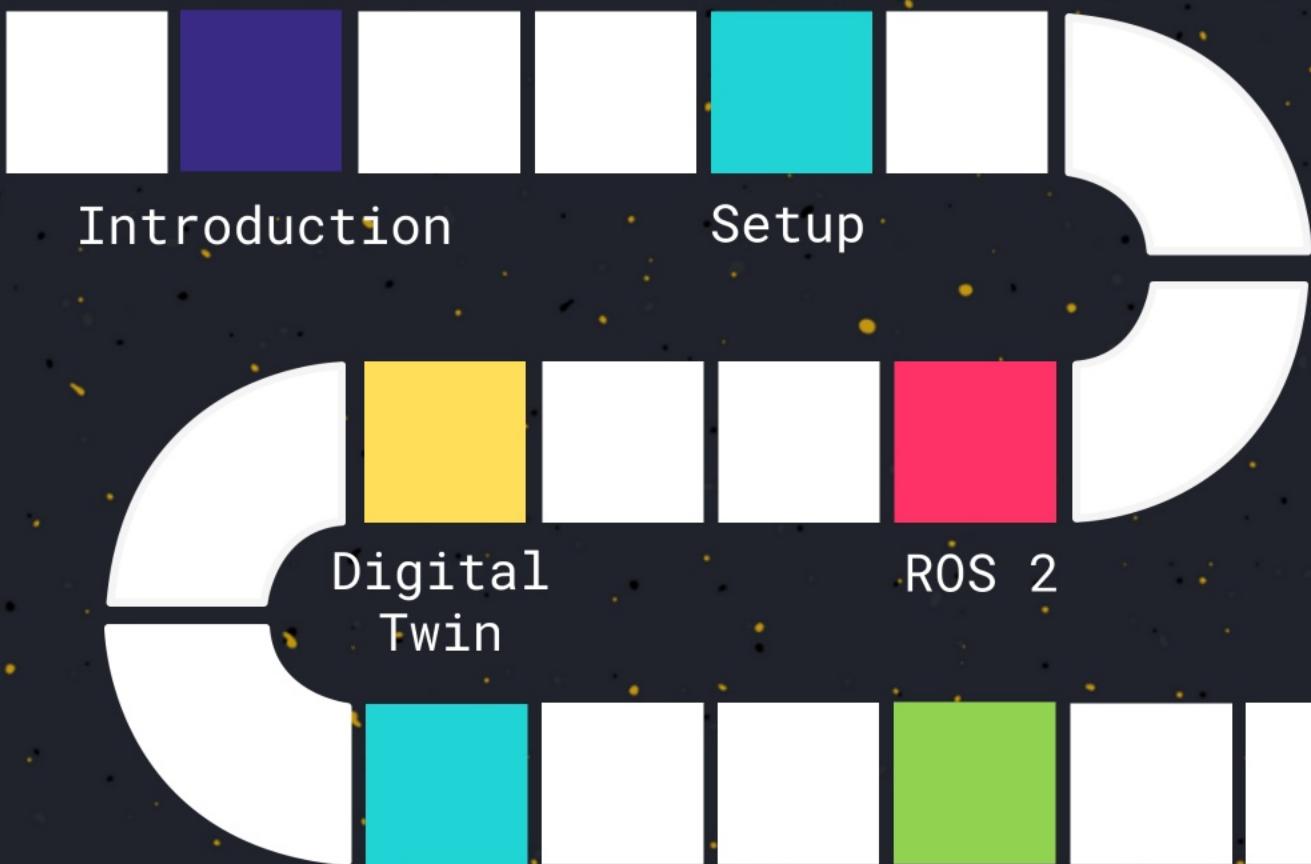
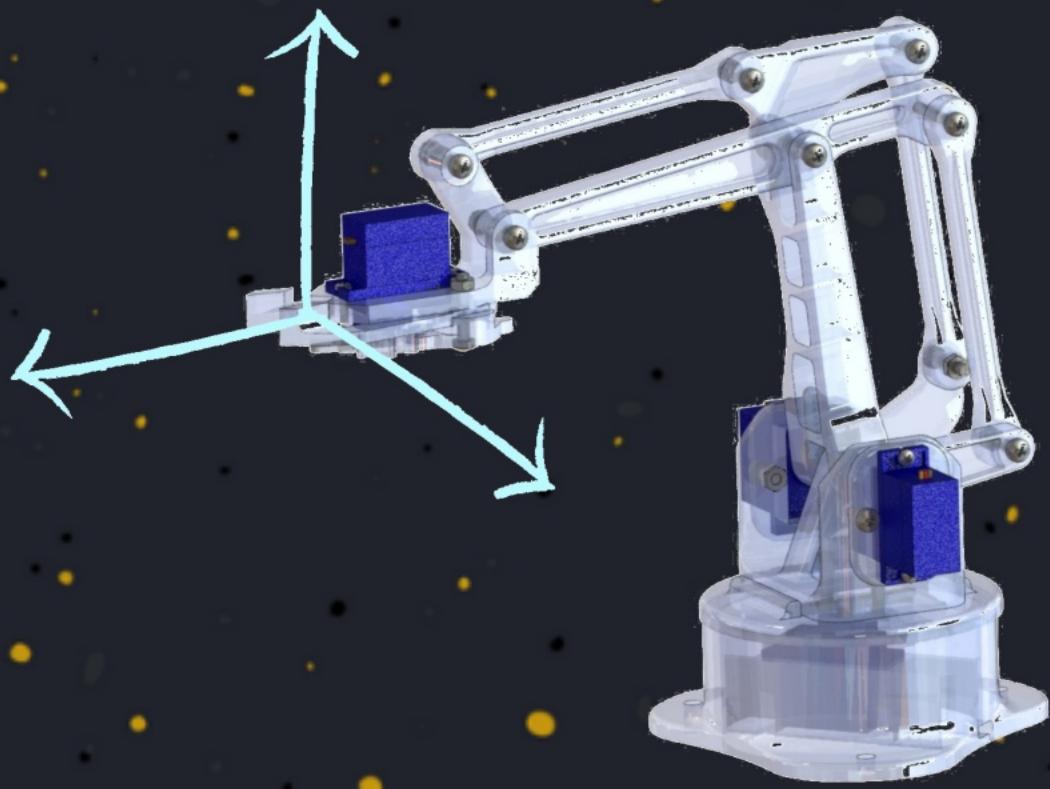
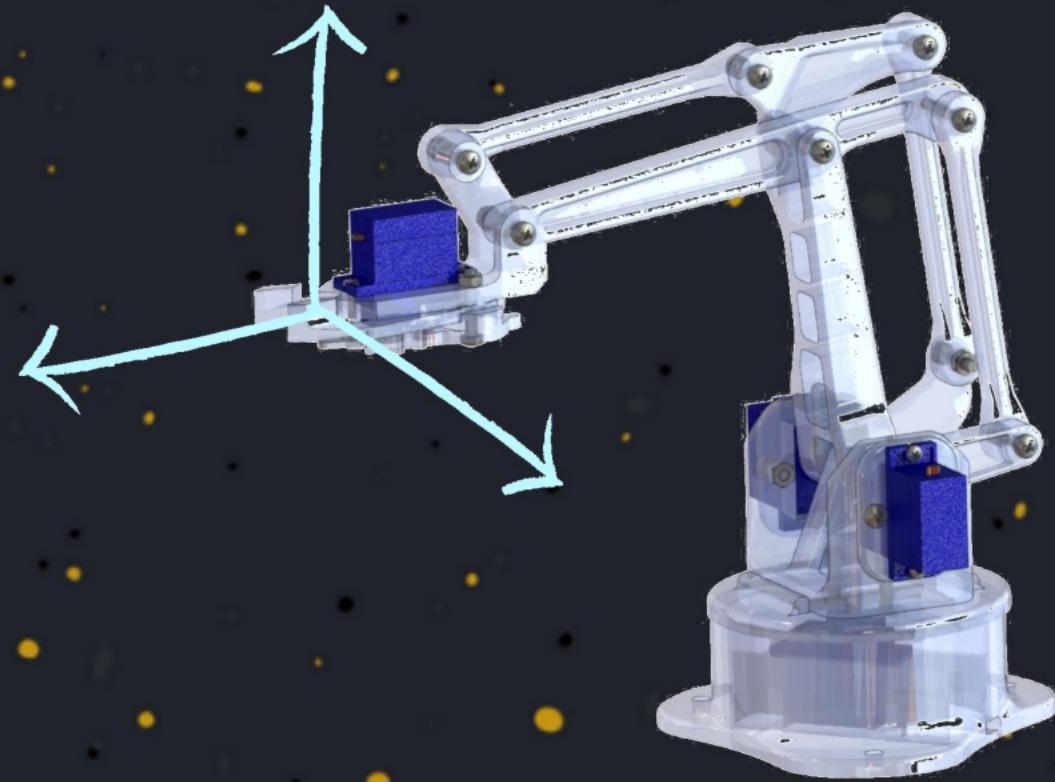
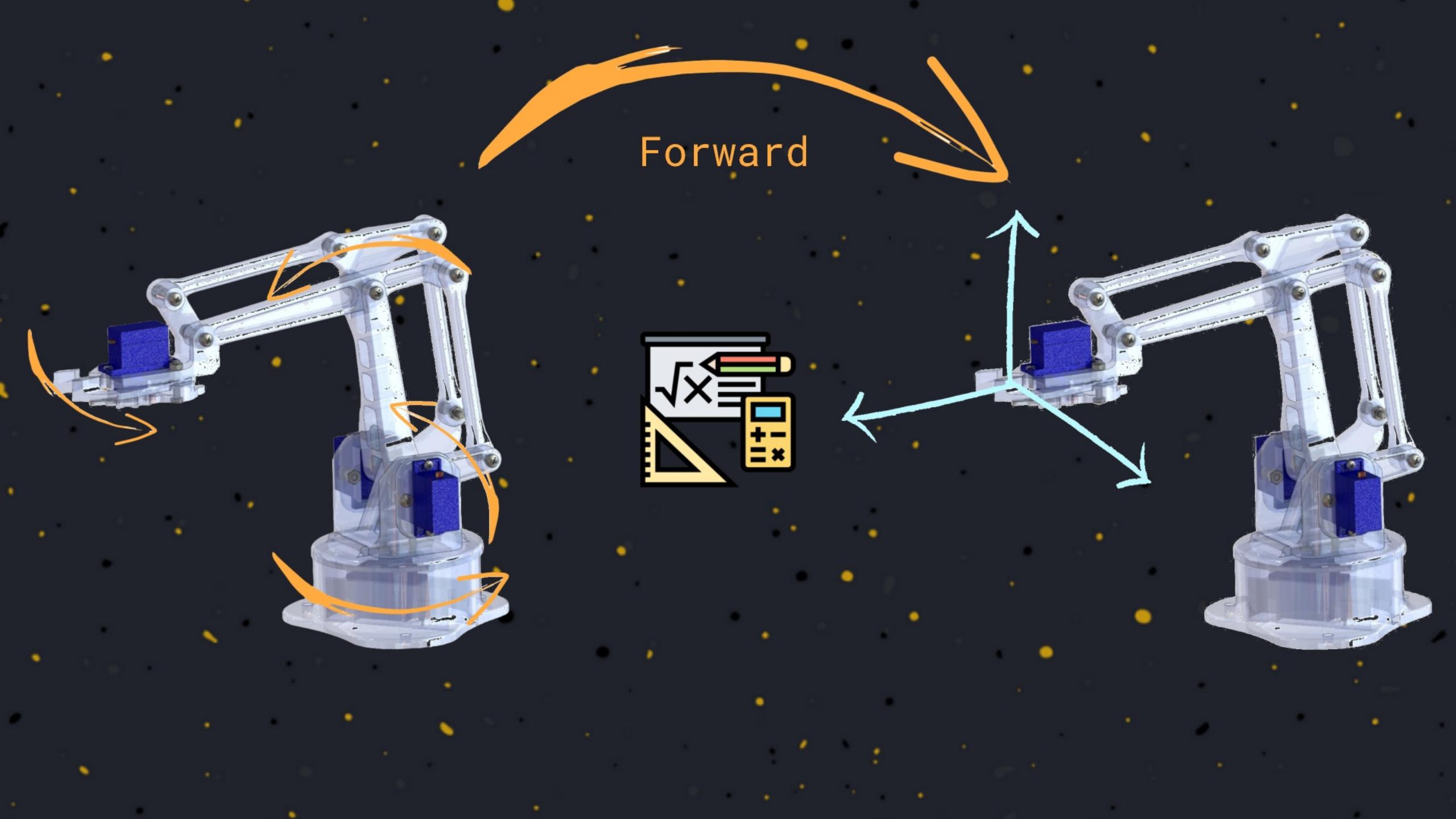


Arduinobot





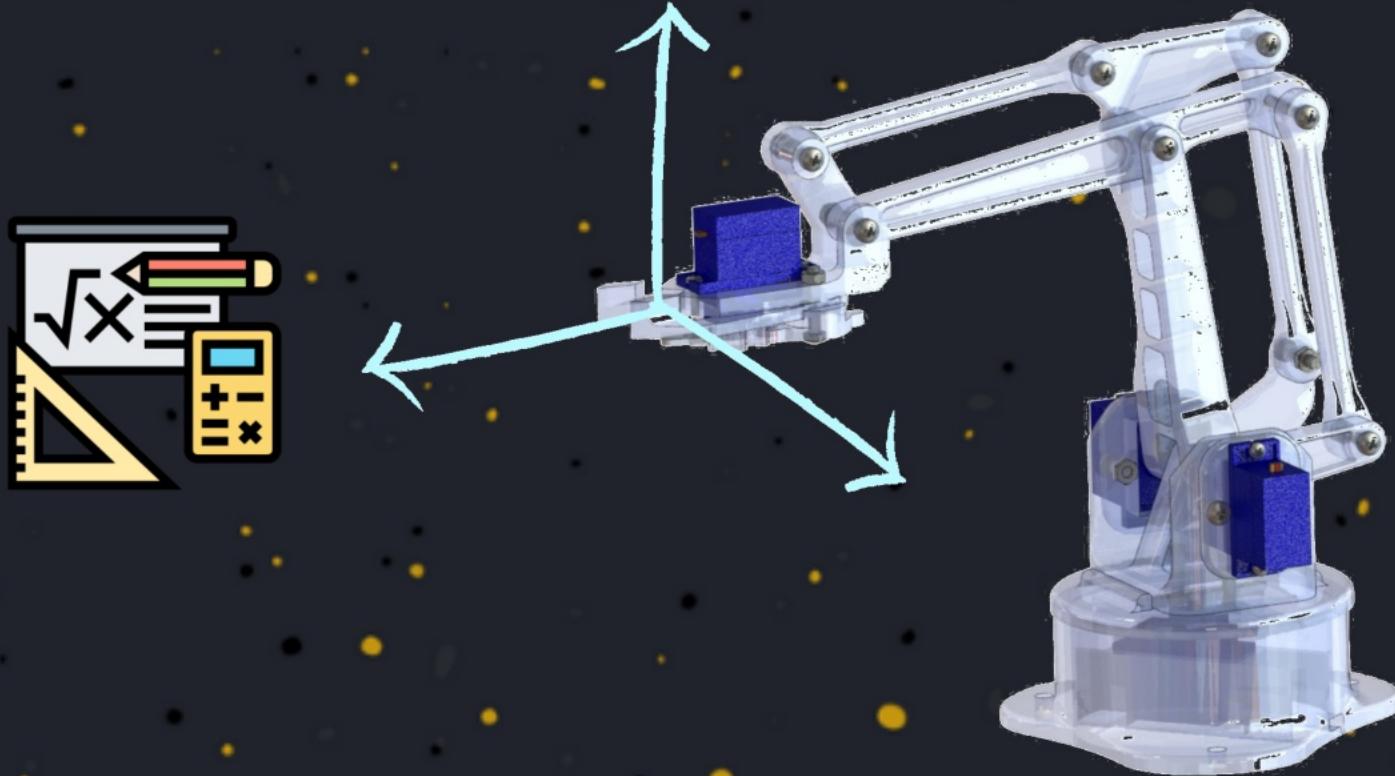
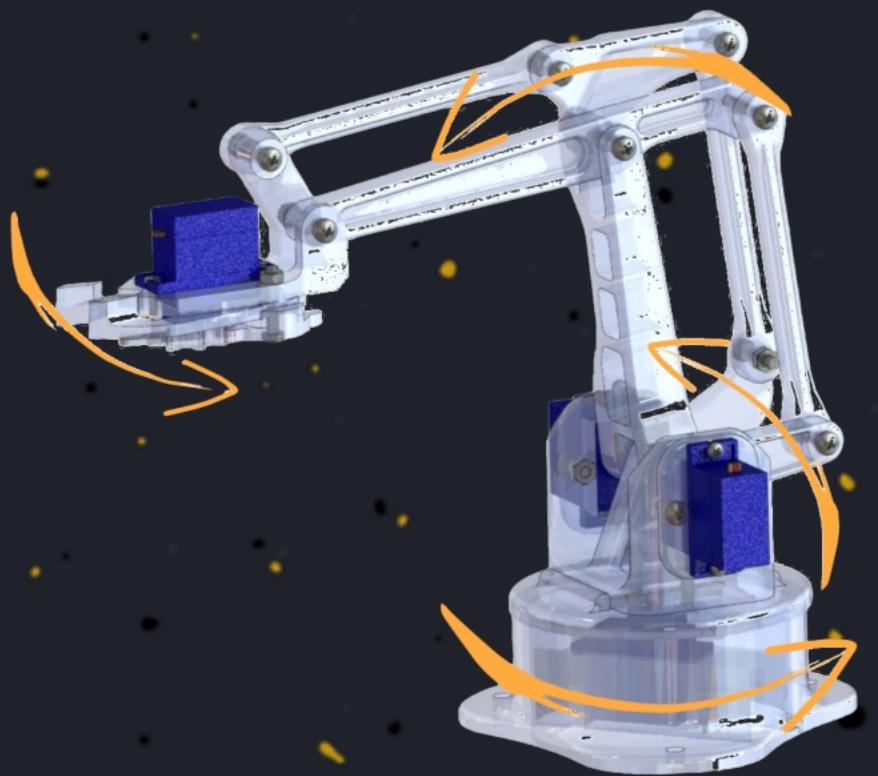




Forward

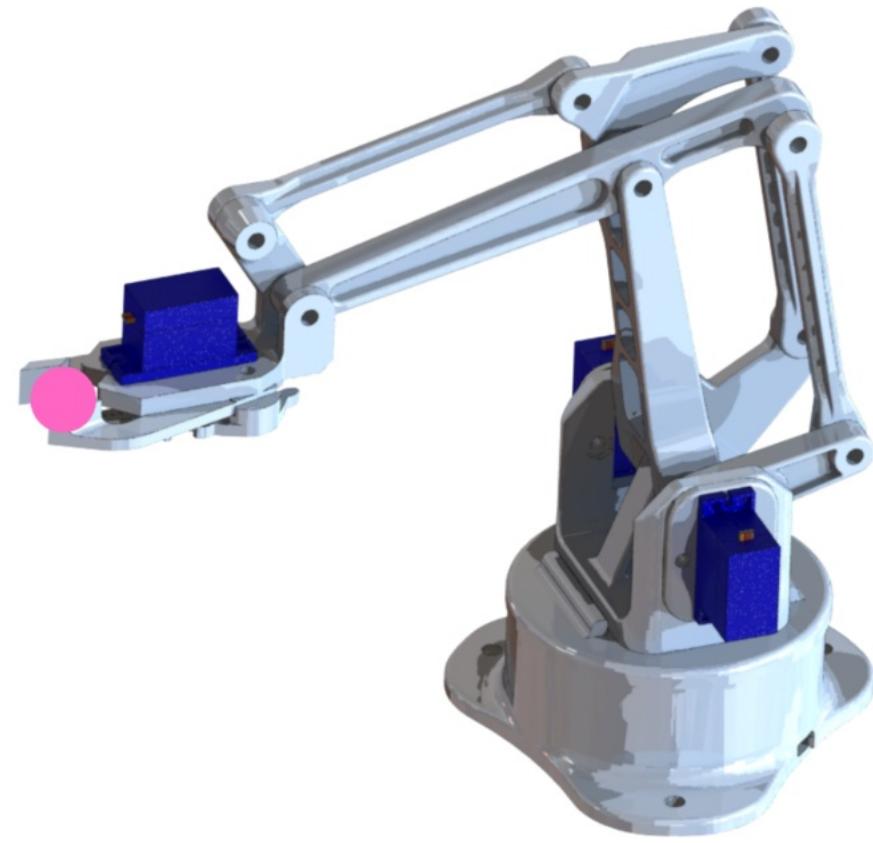


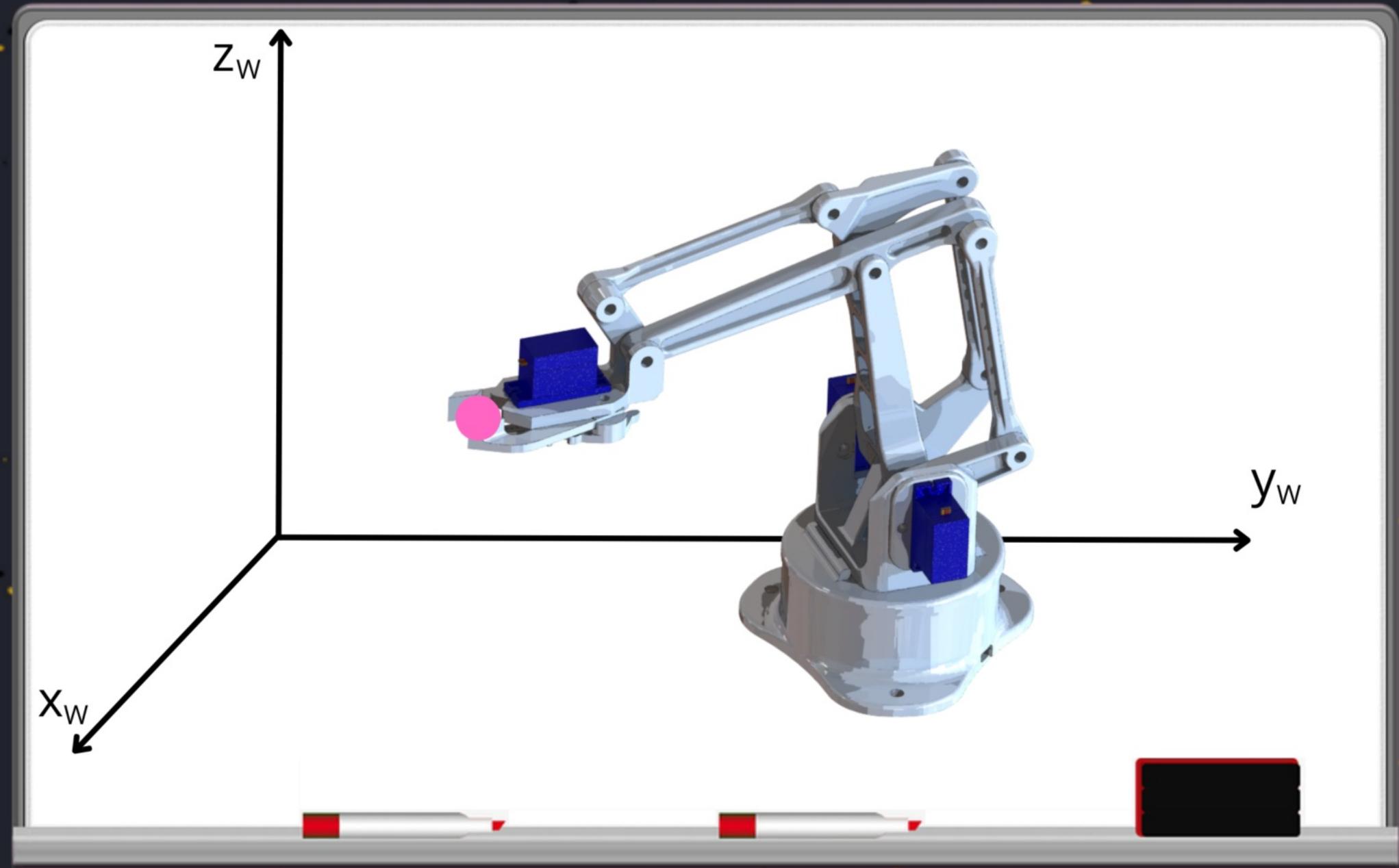
Forward

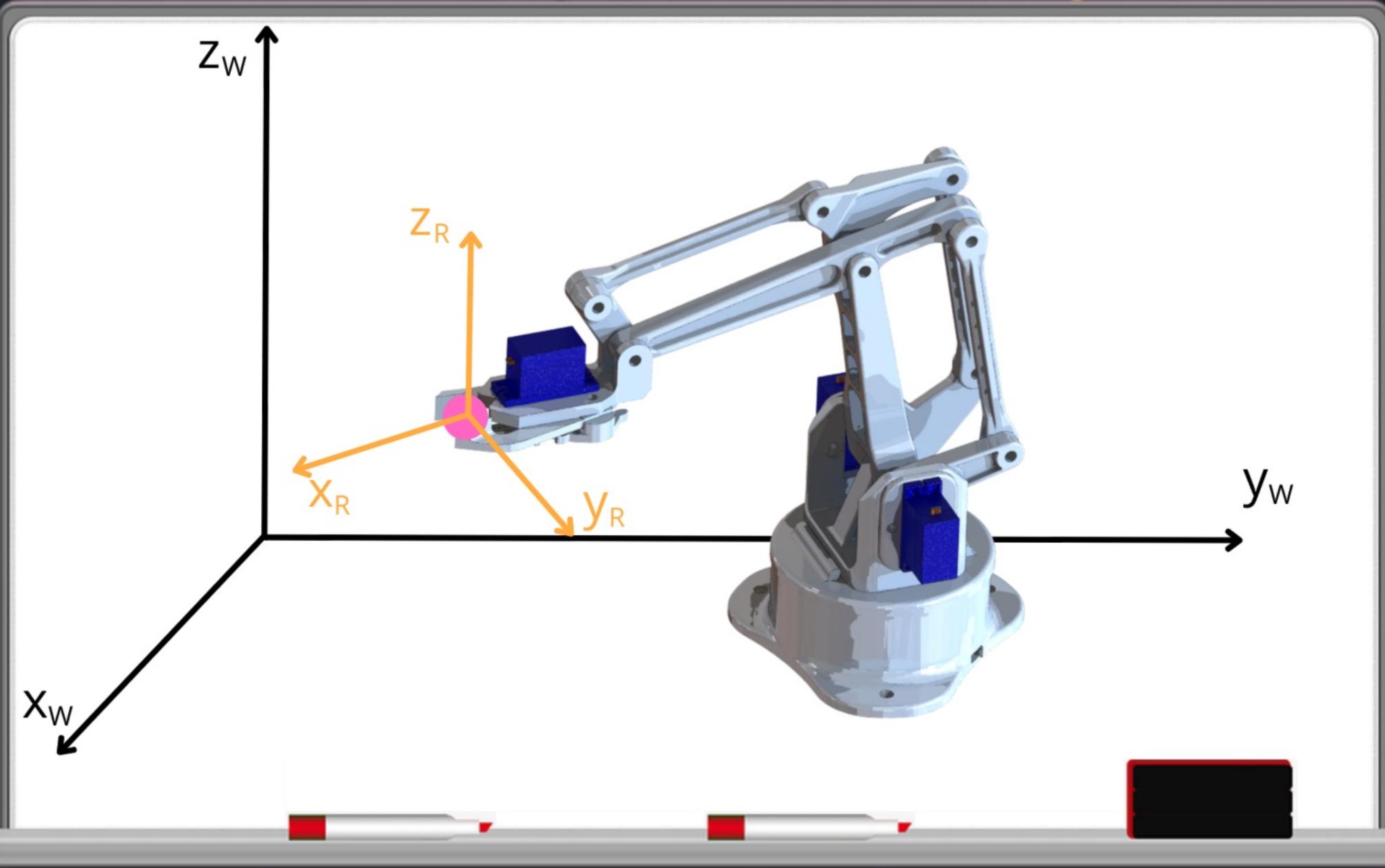


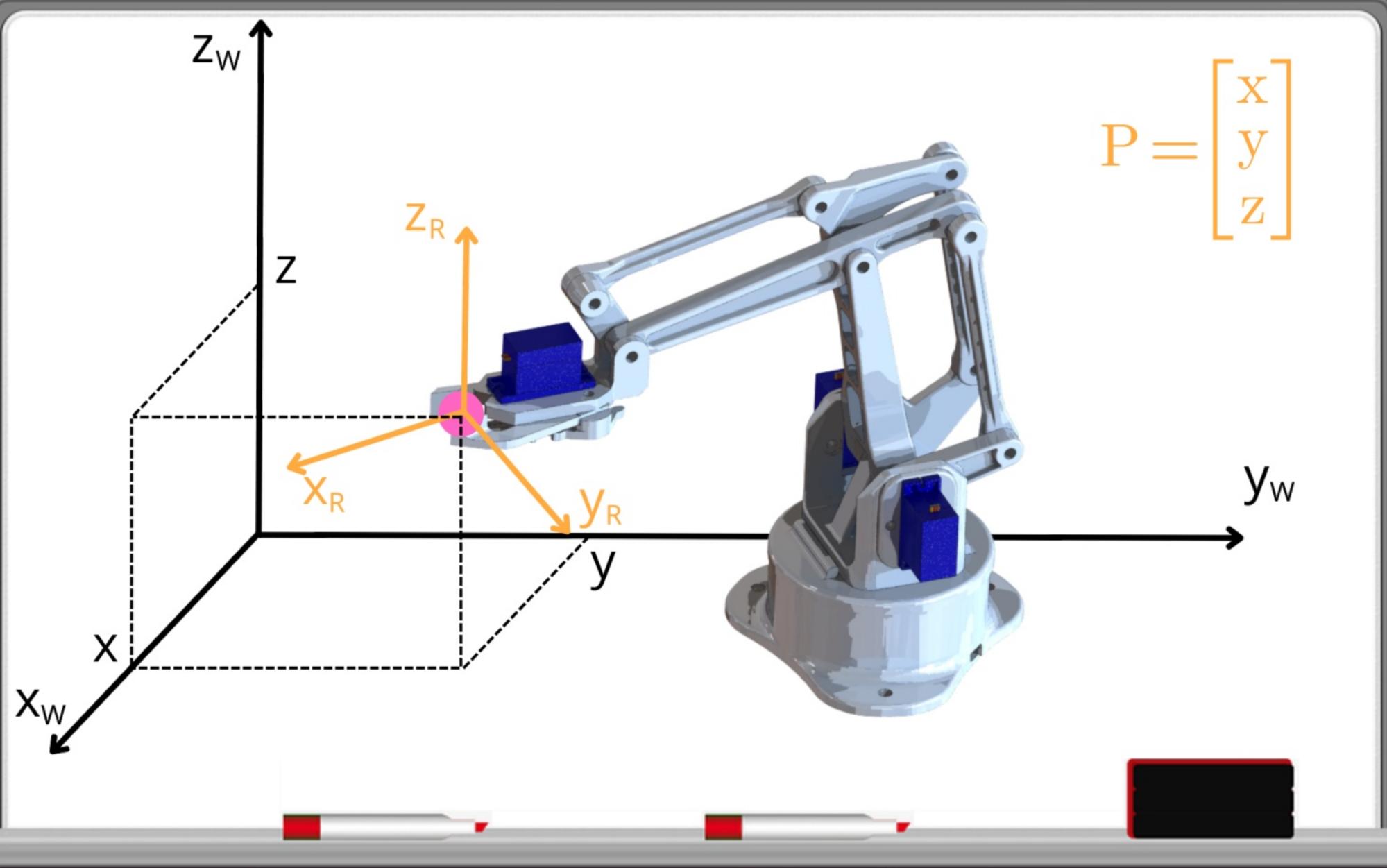
Inverse

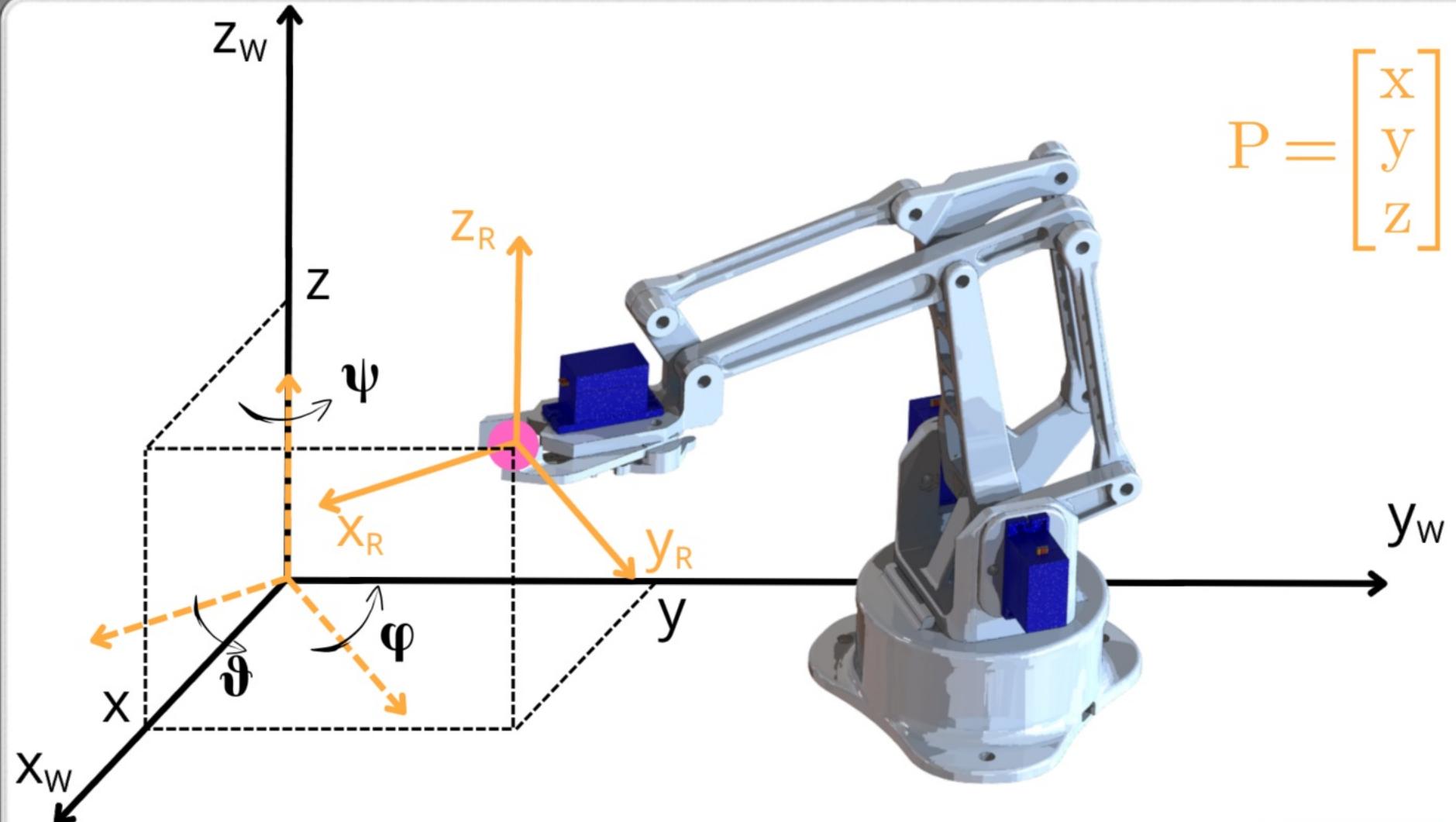


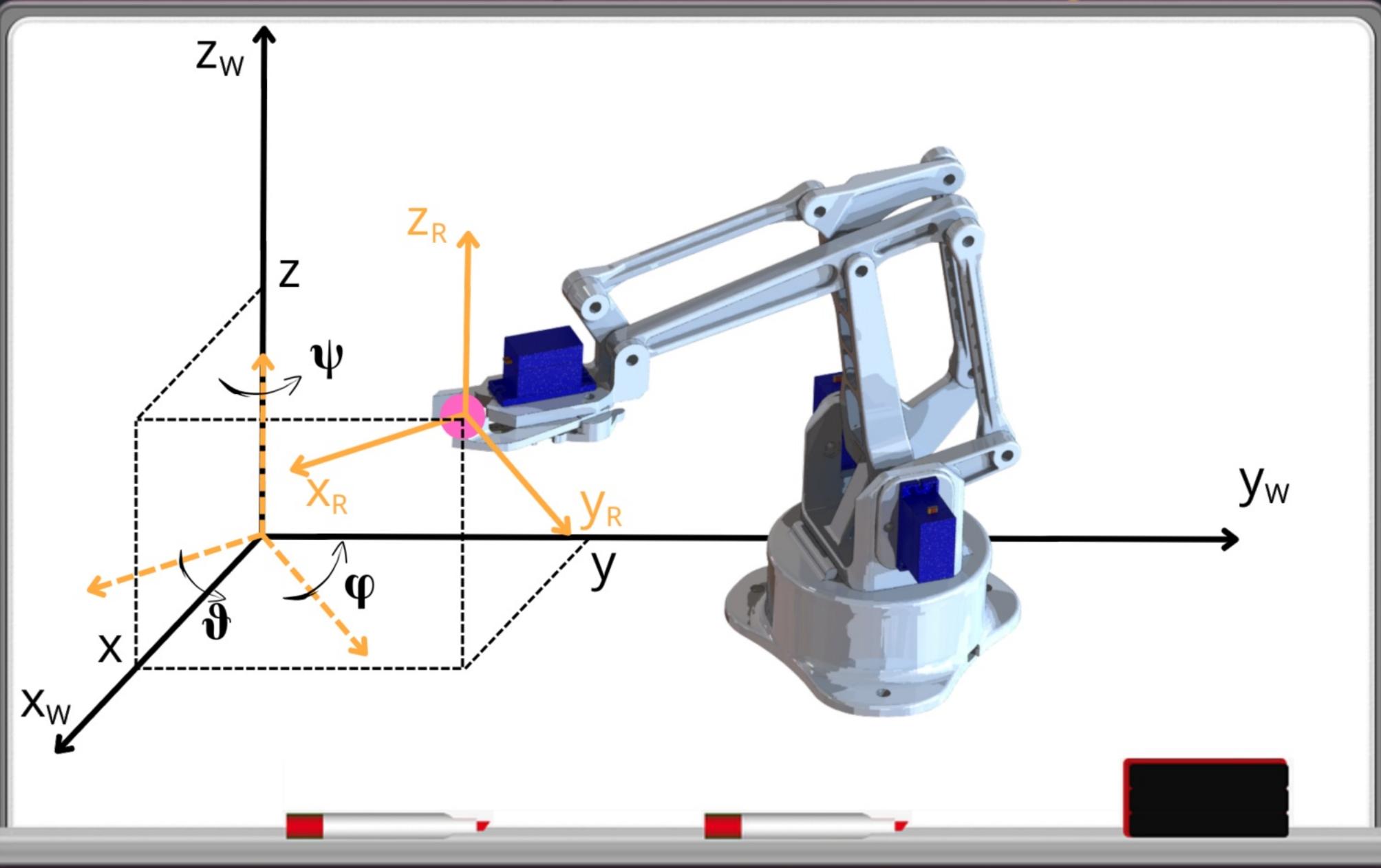


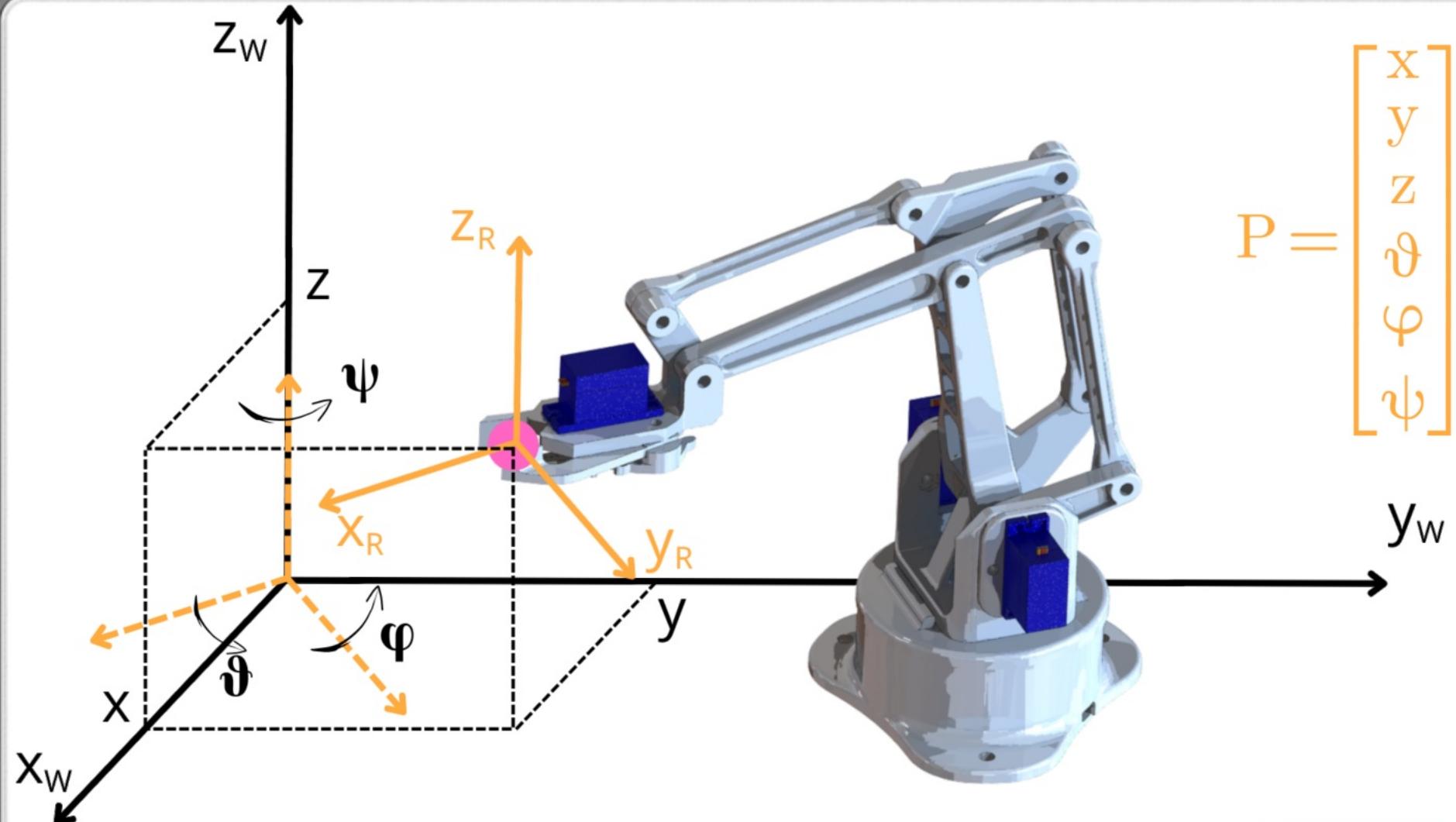


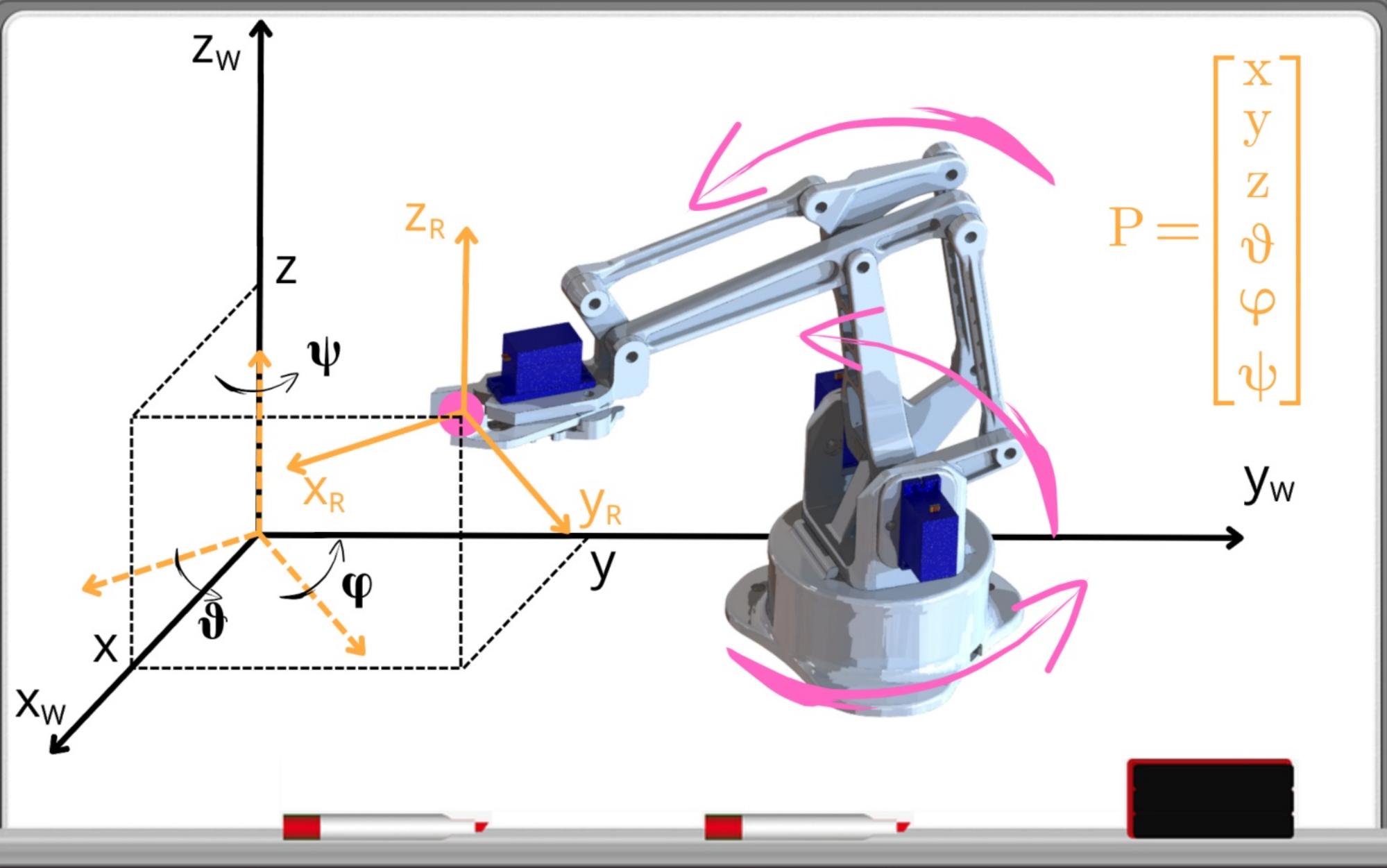








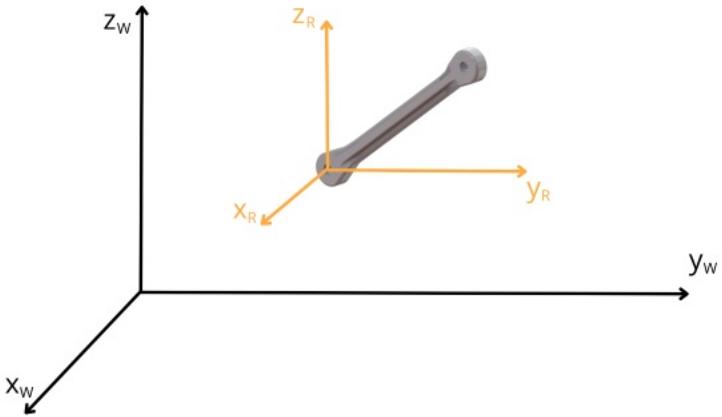




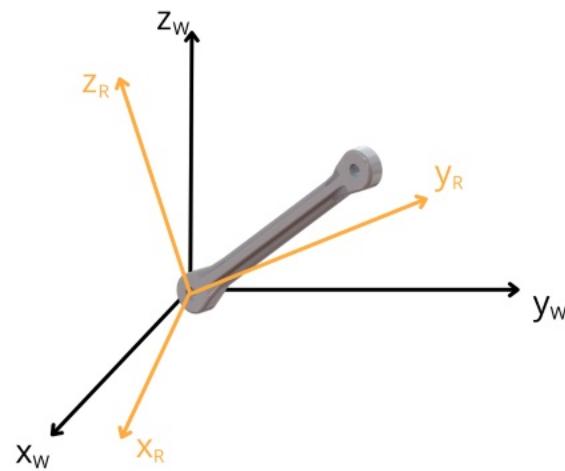




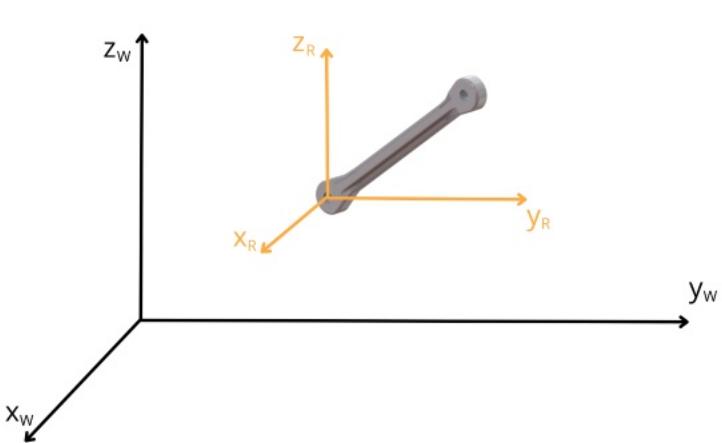




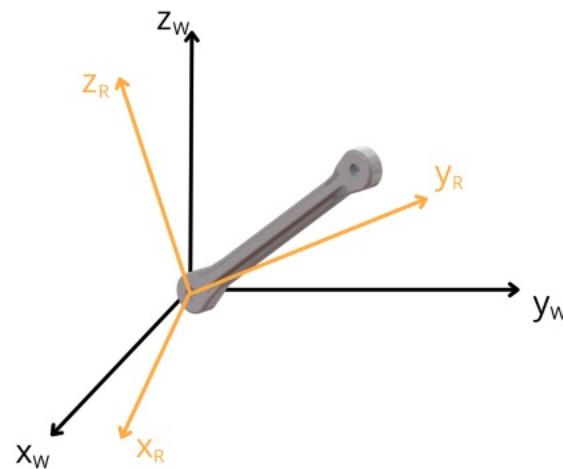
Translation



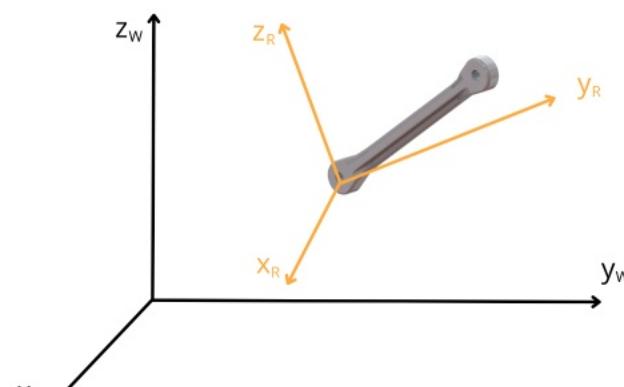
Rotation



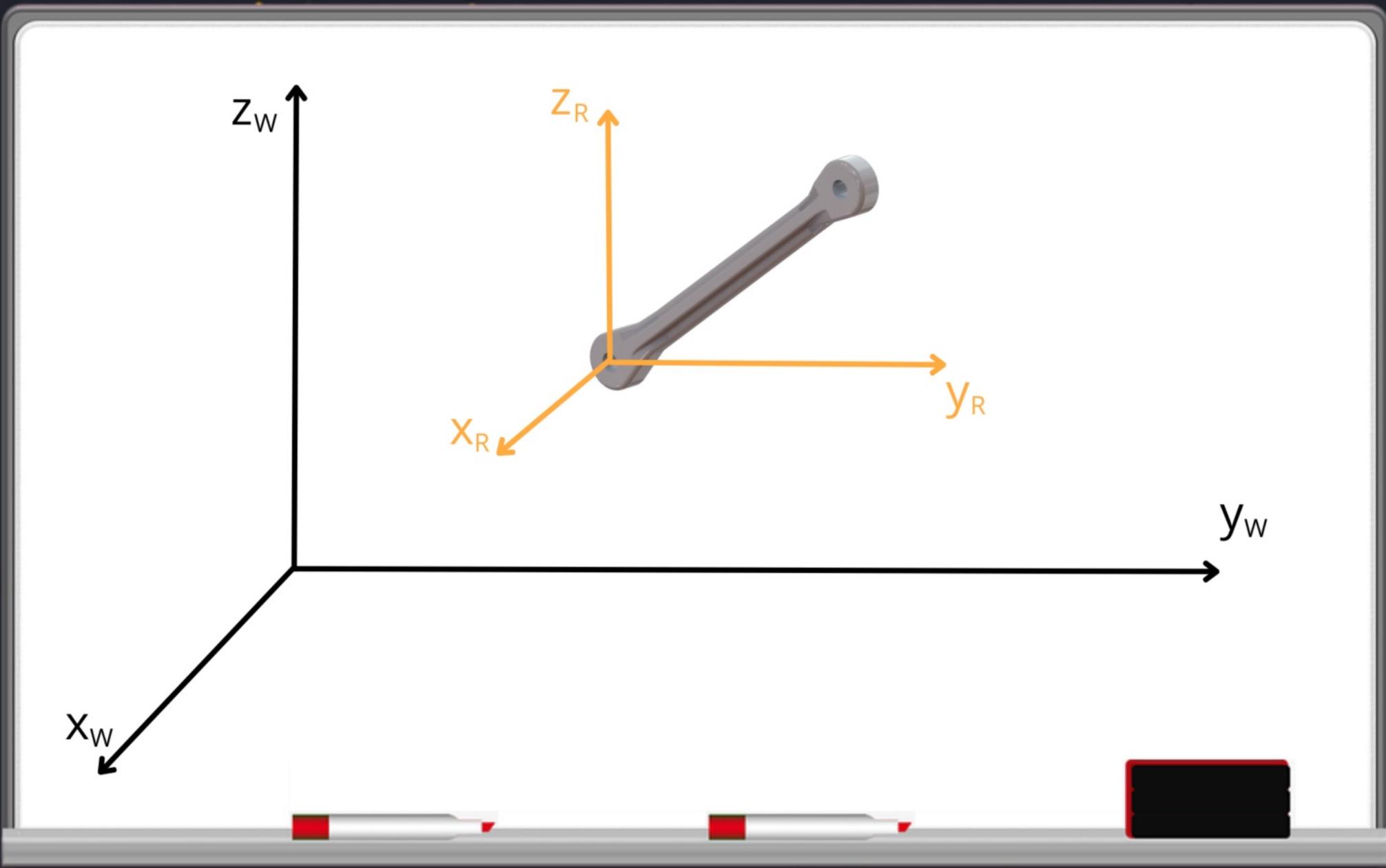
Translation

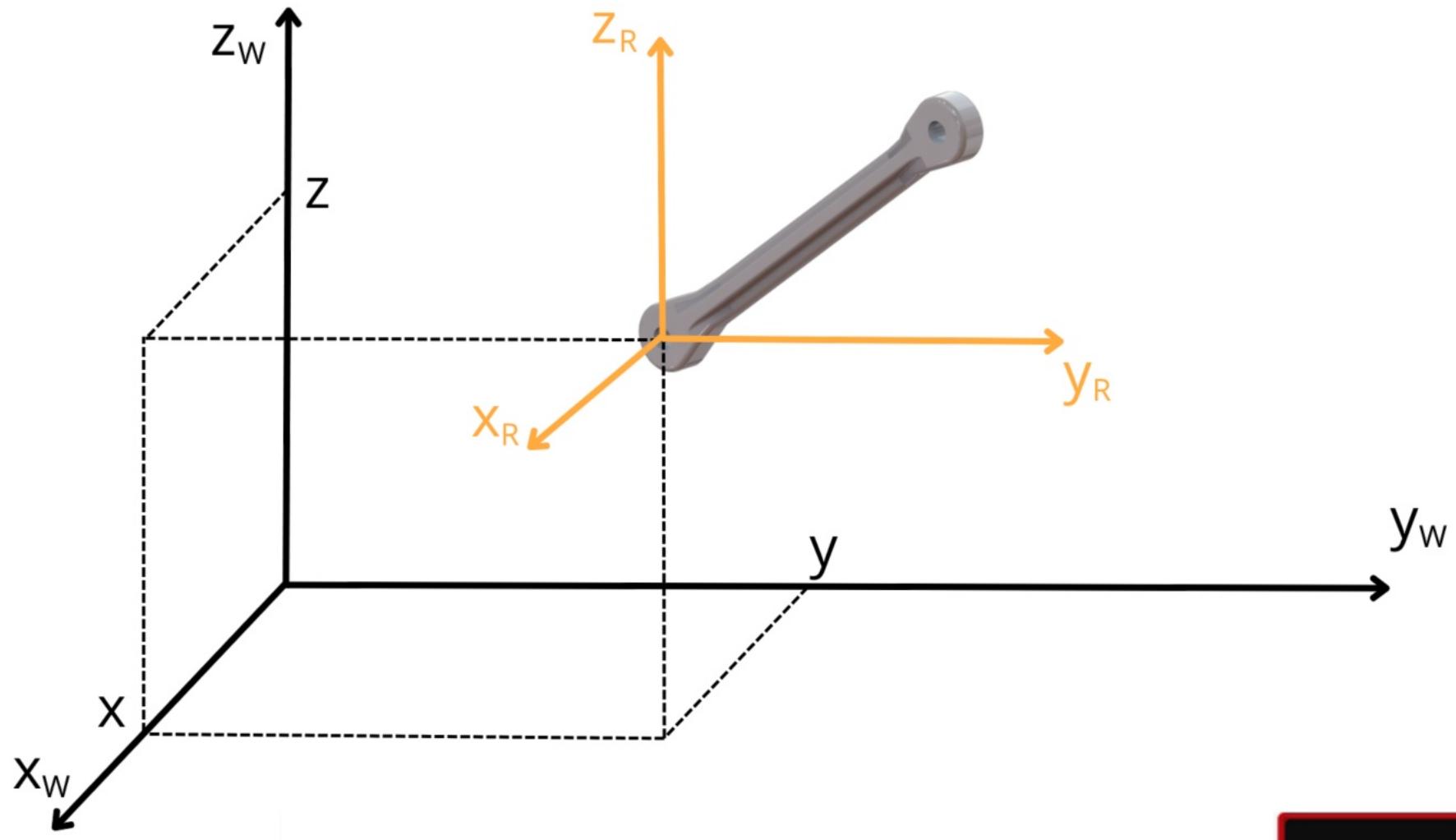


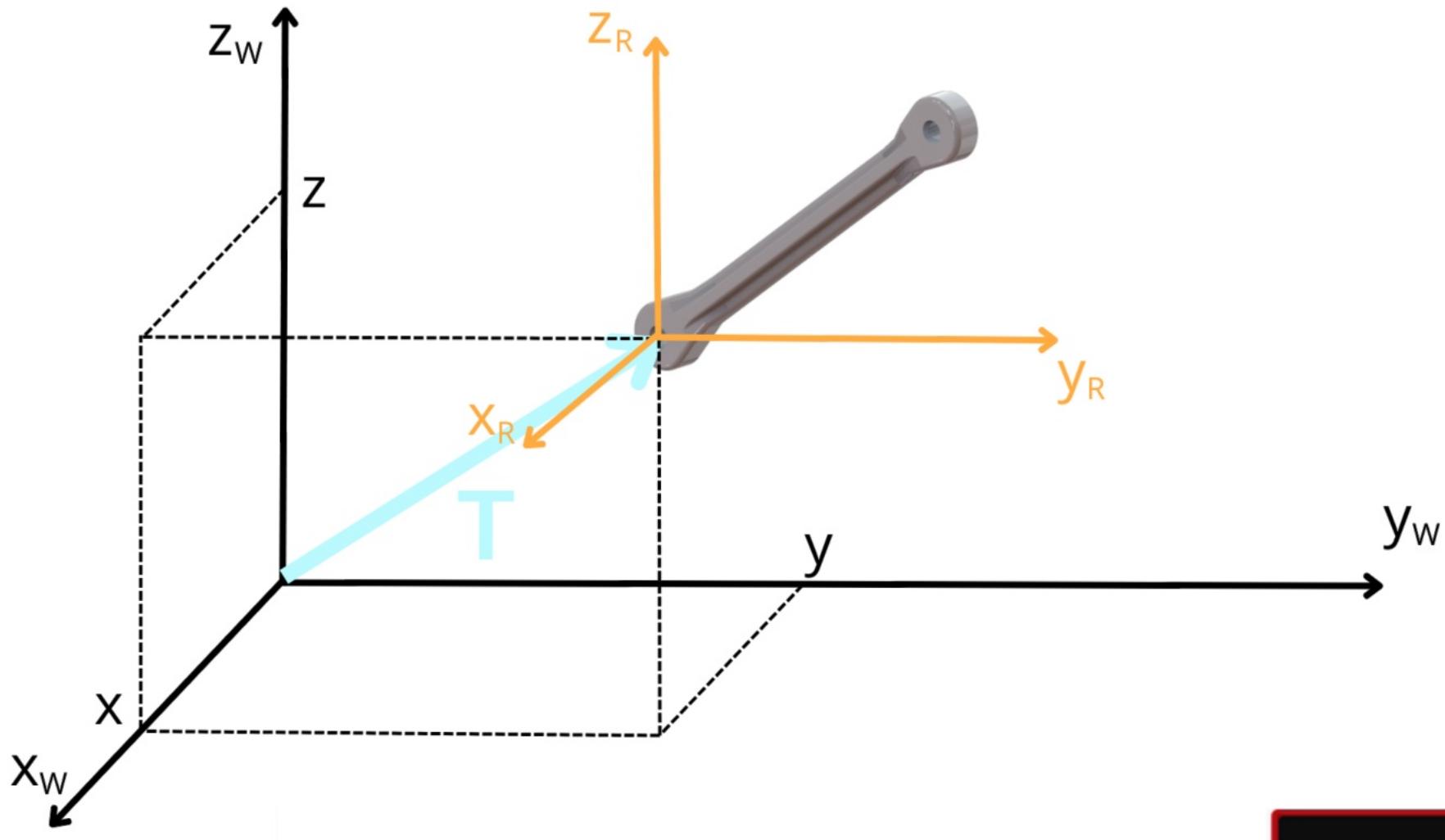
Rotation

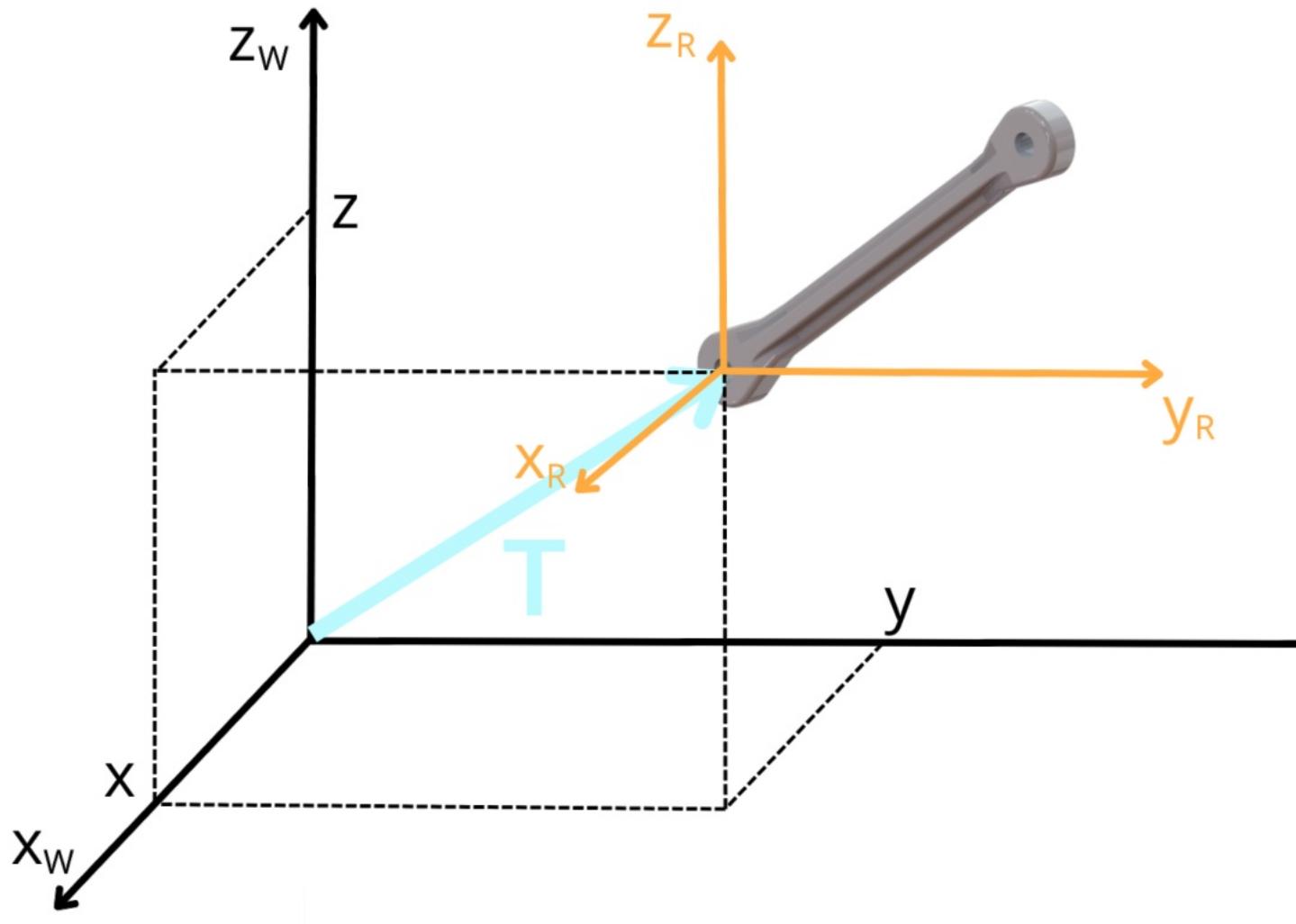


Roto-Translation

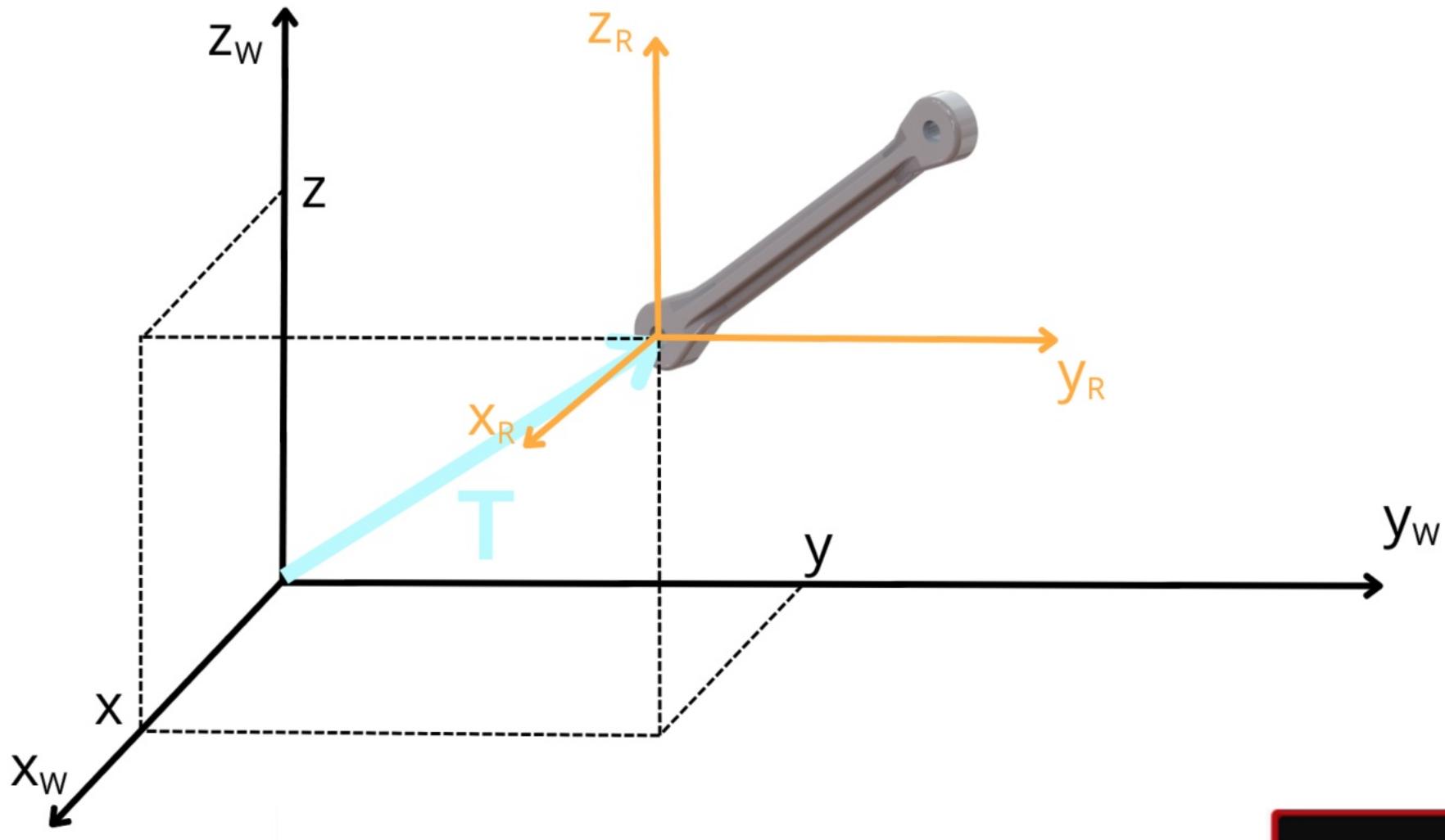


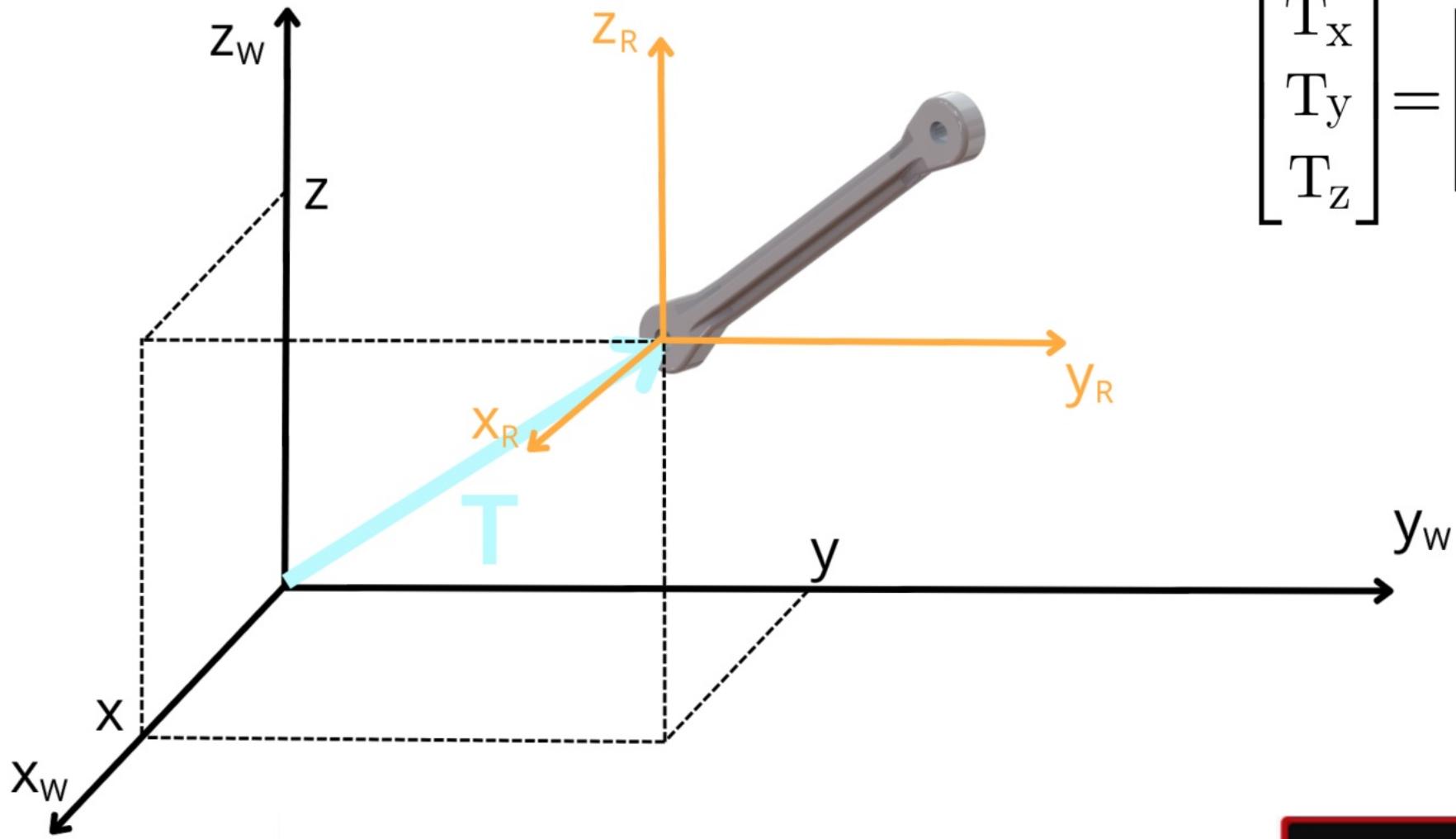




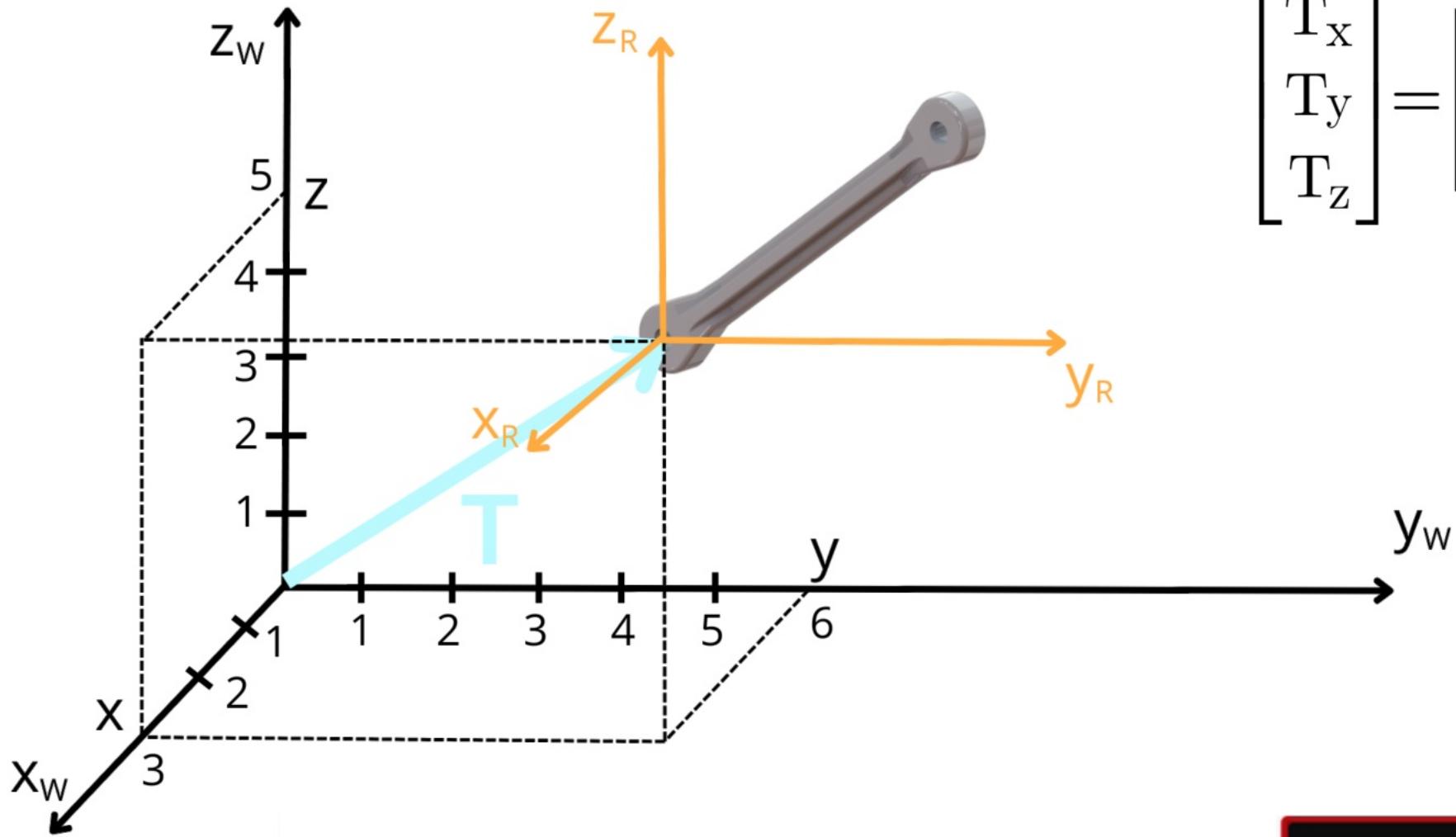


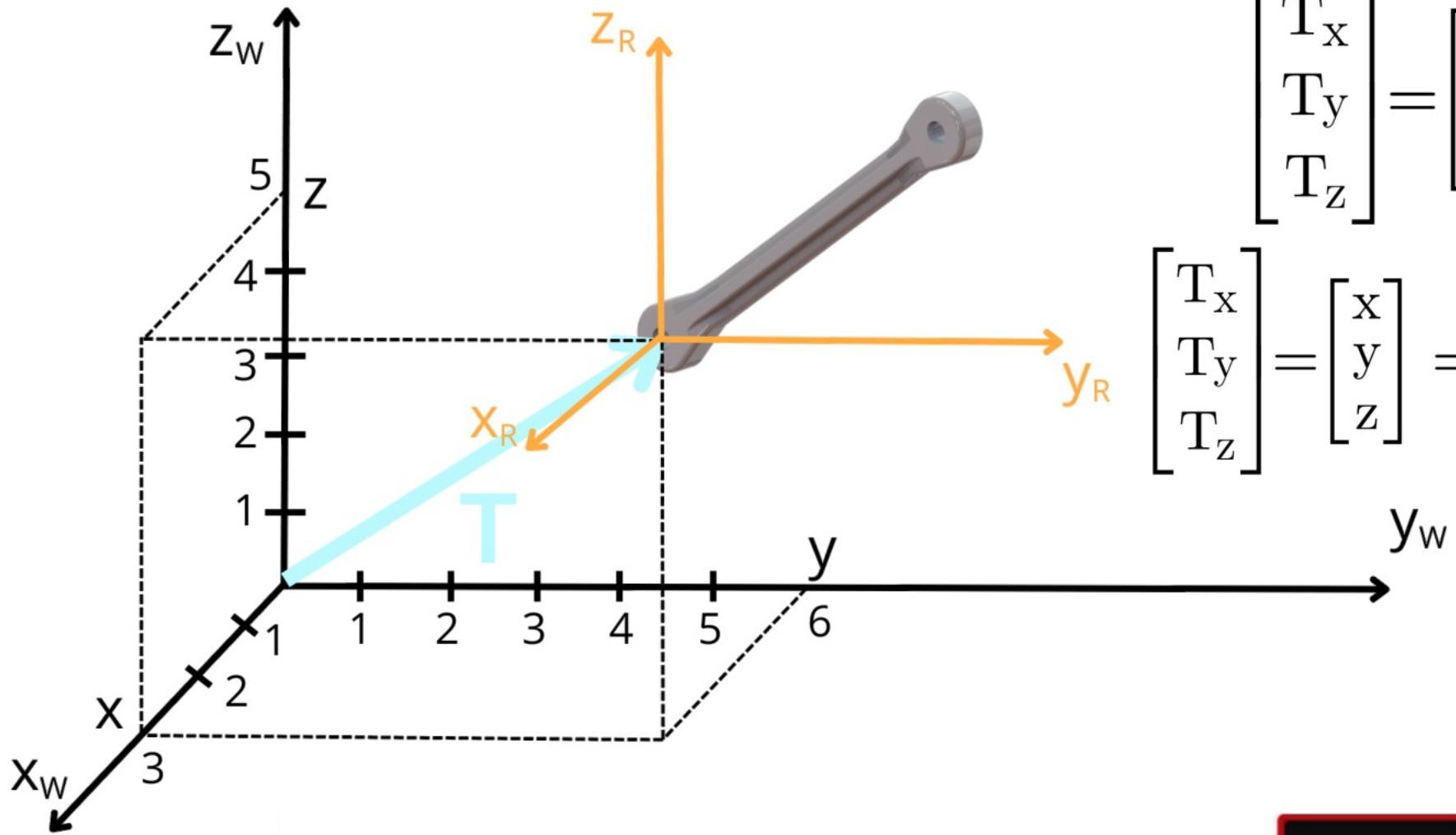
$$\begin{cases} T_x = x \\ T_y = y \\ T_z = z \end{cases}$$

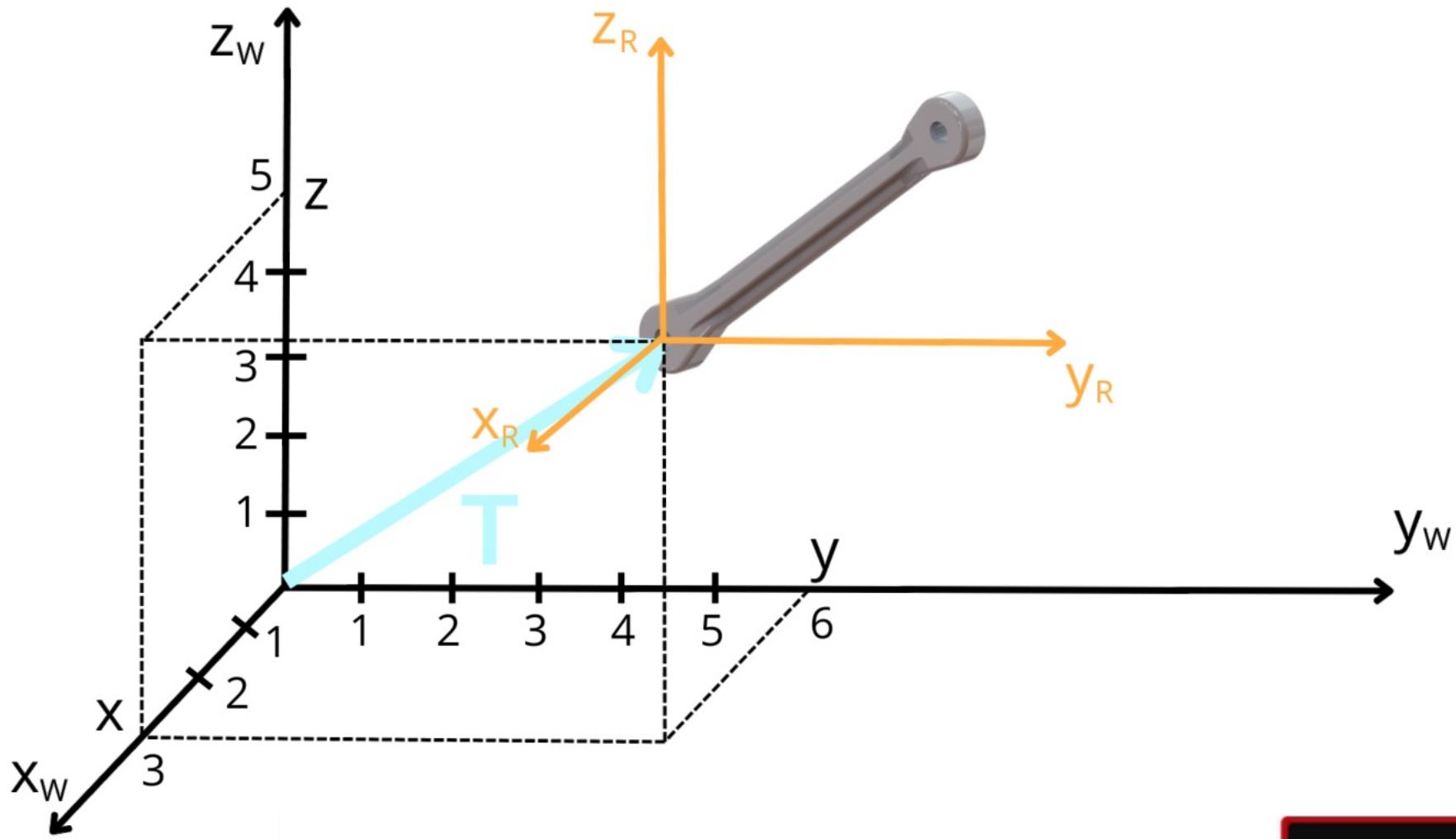


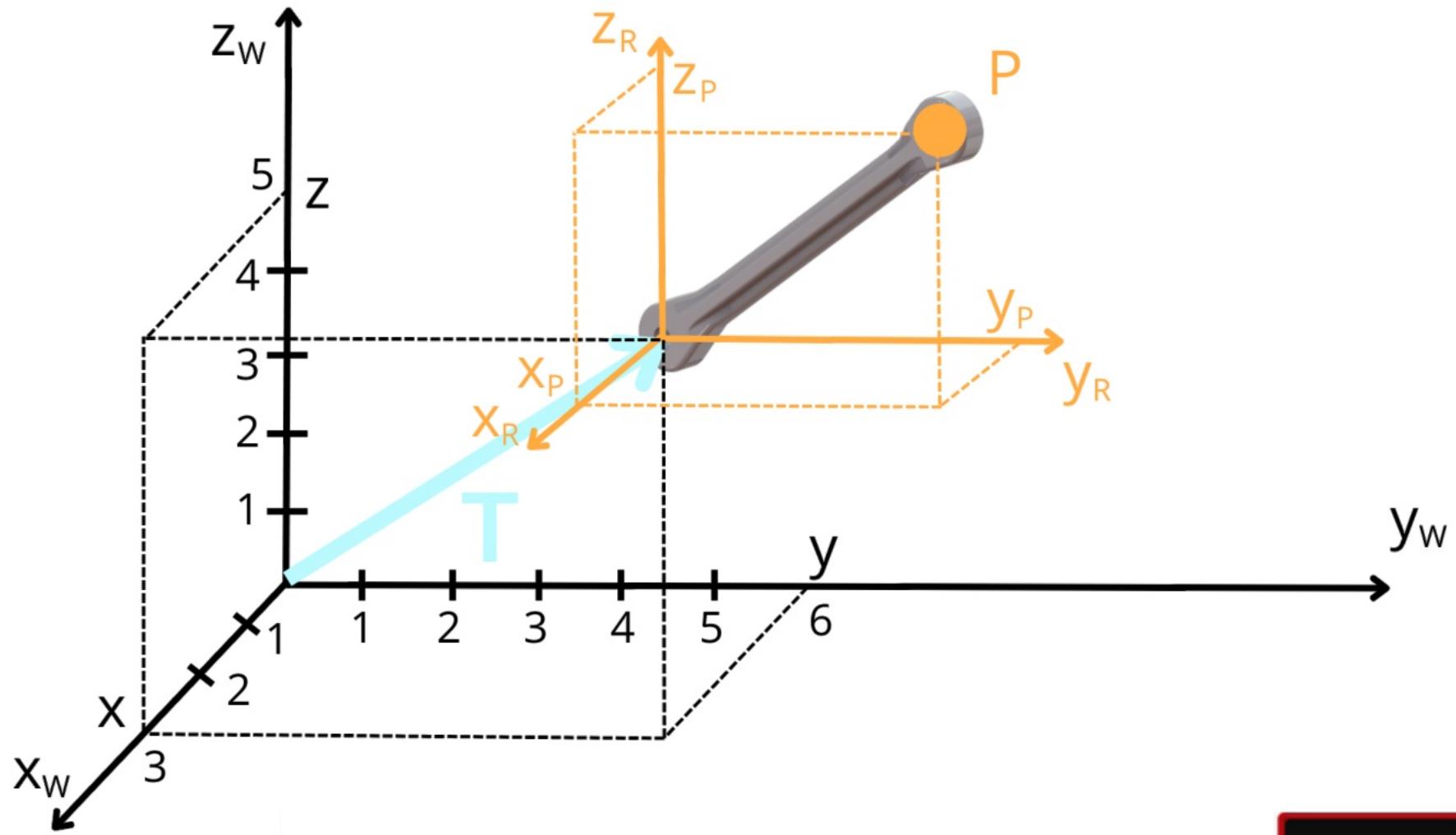


$$\begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

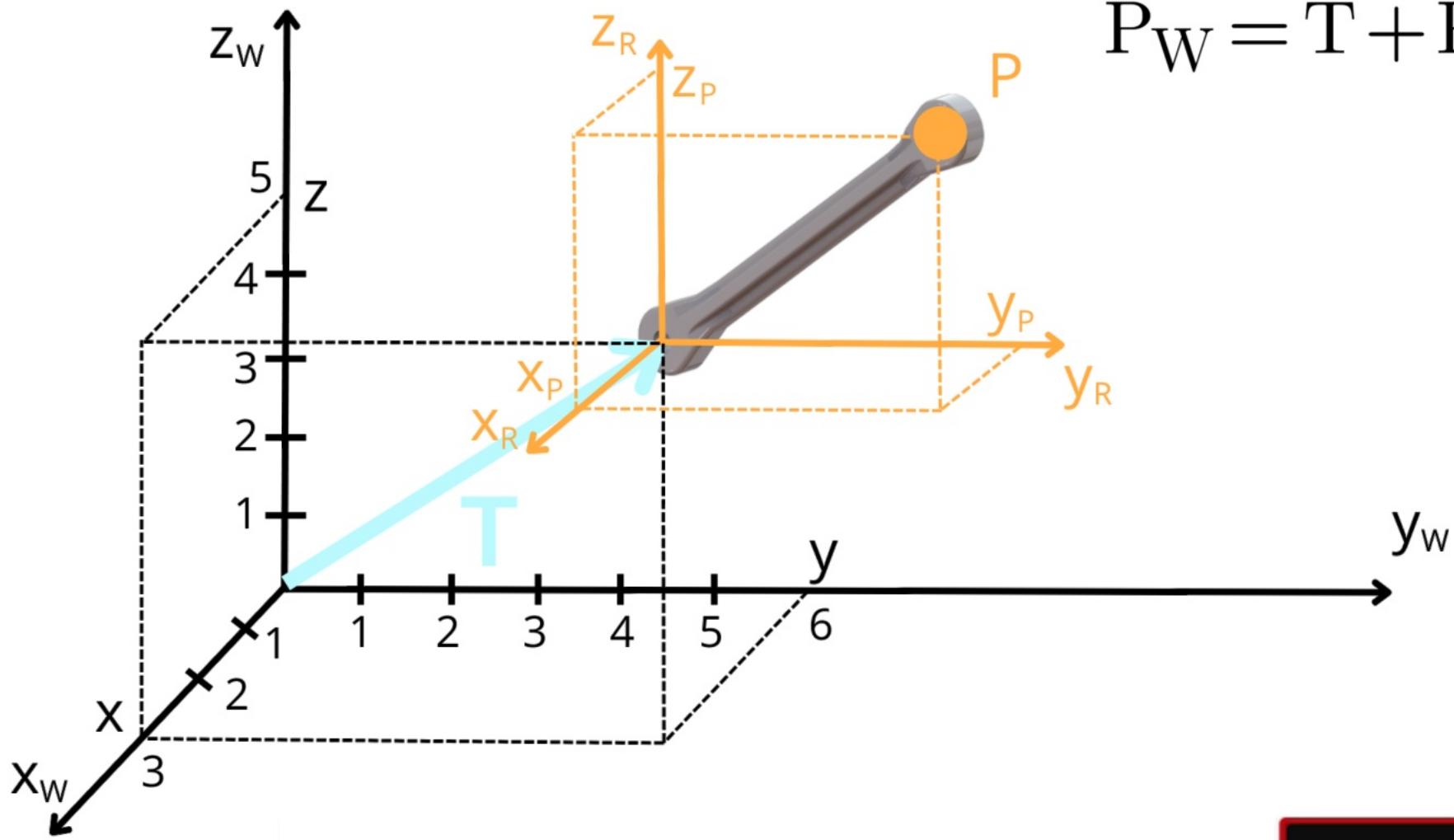






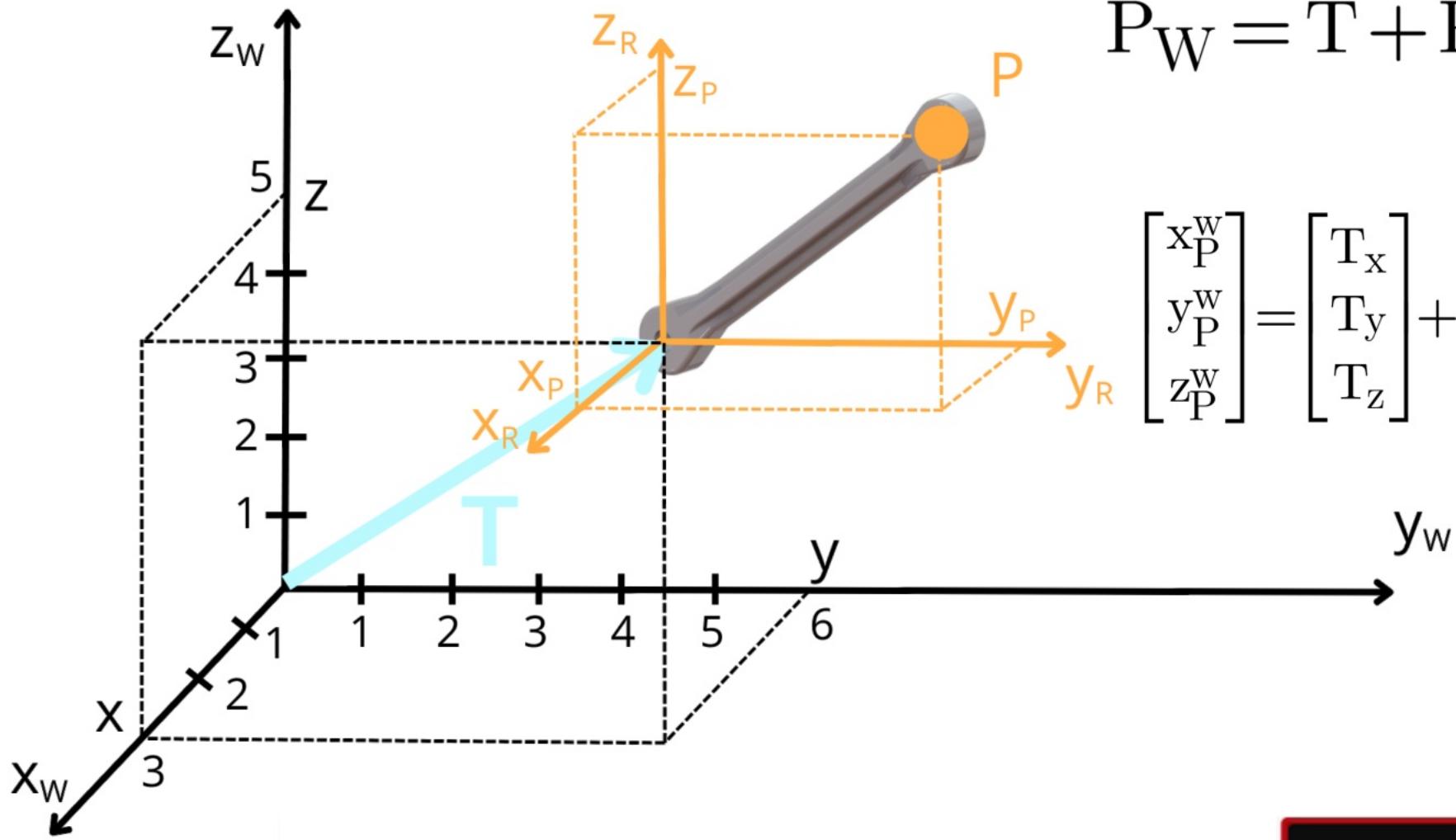


$$P_W = T + P_R$$



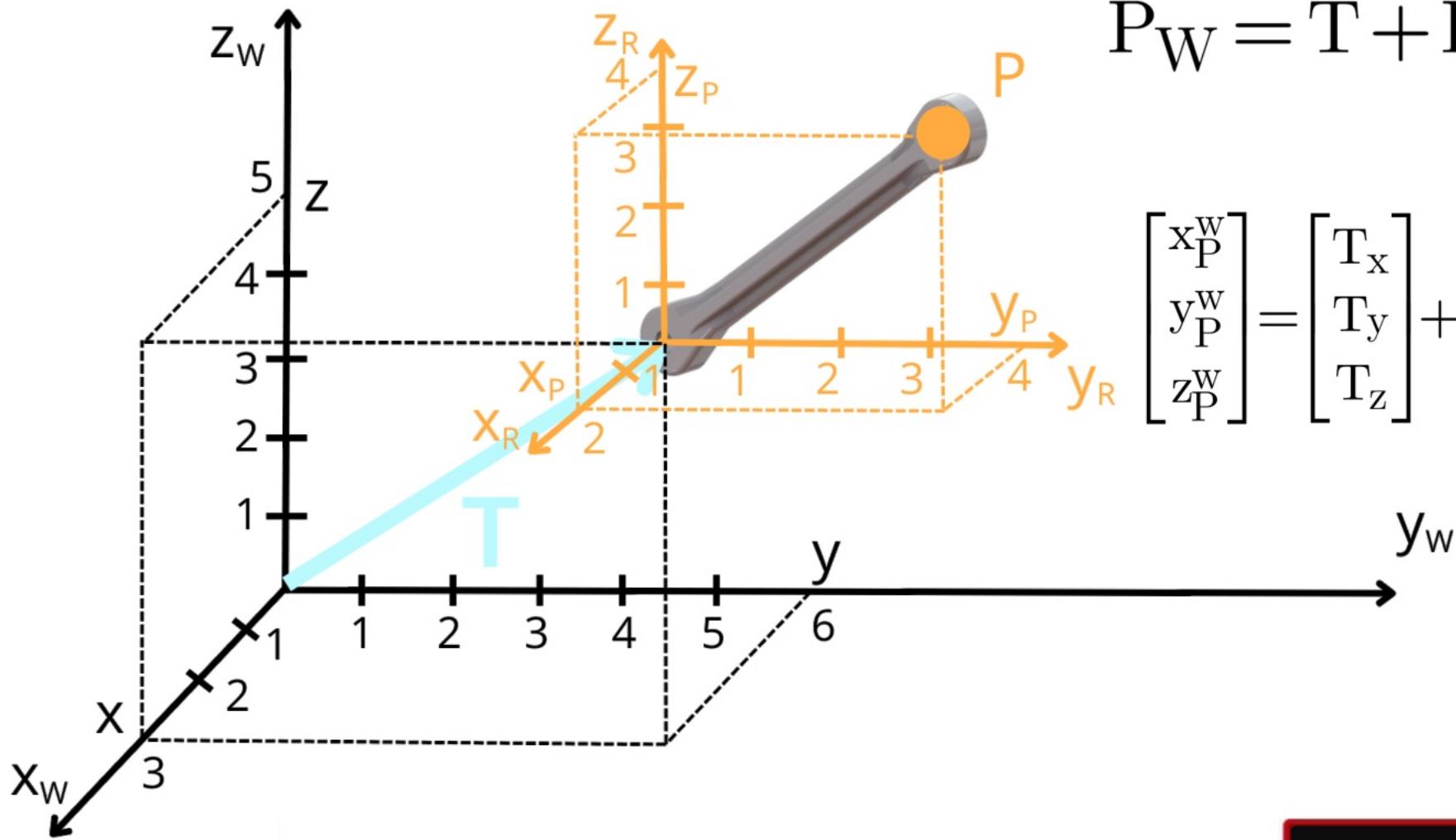
$$P_W = T + P_R$$

$$\begin{bmatrix} x_P^W \\ y_P^W \\ z_P^W \end{bmatrix} = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} + \begin{bmatrix} x_P \\ y_P \\ z_P \end{bmatrix}$$

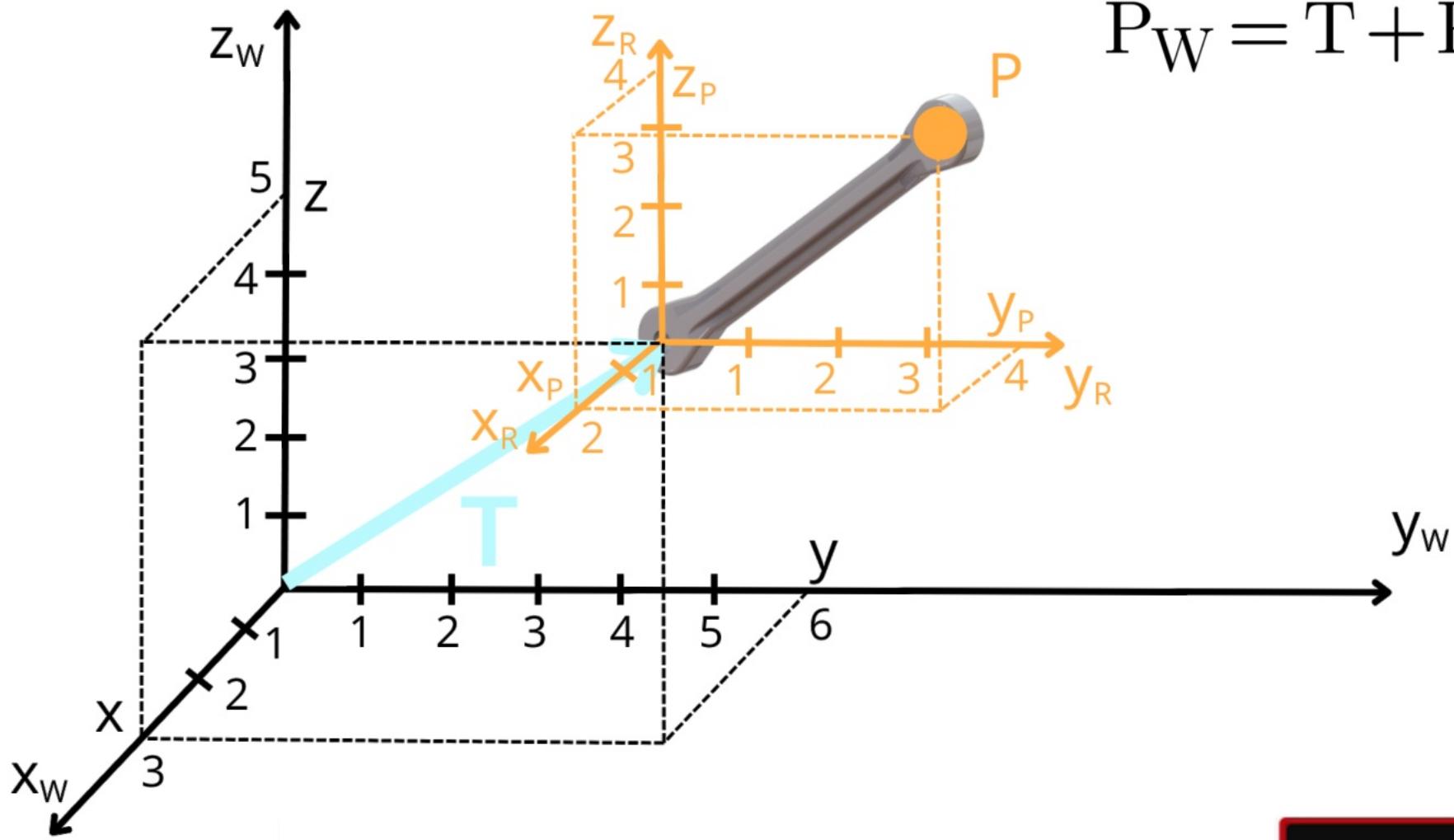


$$P_W = T + P_R$$

$$\begin{bmatrix} x_P^W \\ y_P^W \\ z_P^W \end{bmatrix} = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} + \begin{bmatrix} x_P \\ y_P \\ z_P \end{bmatrix}$$

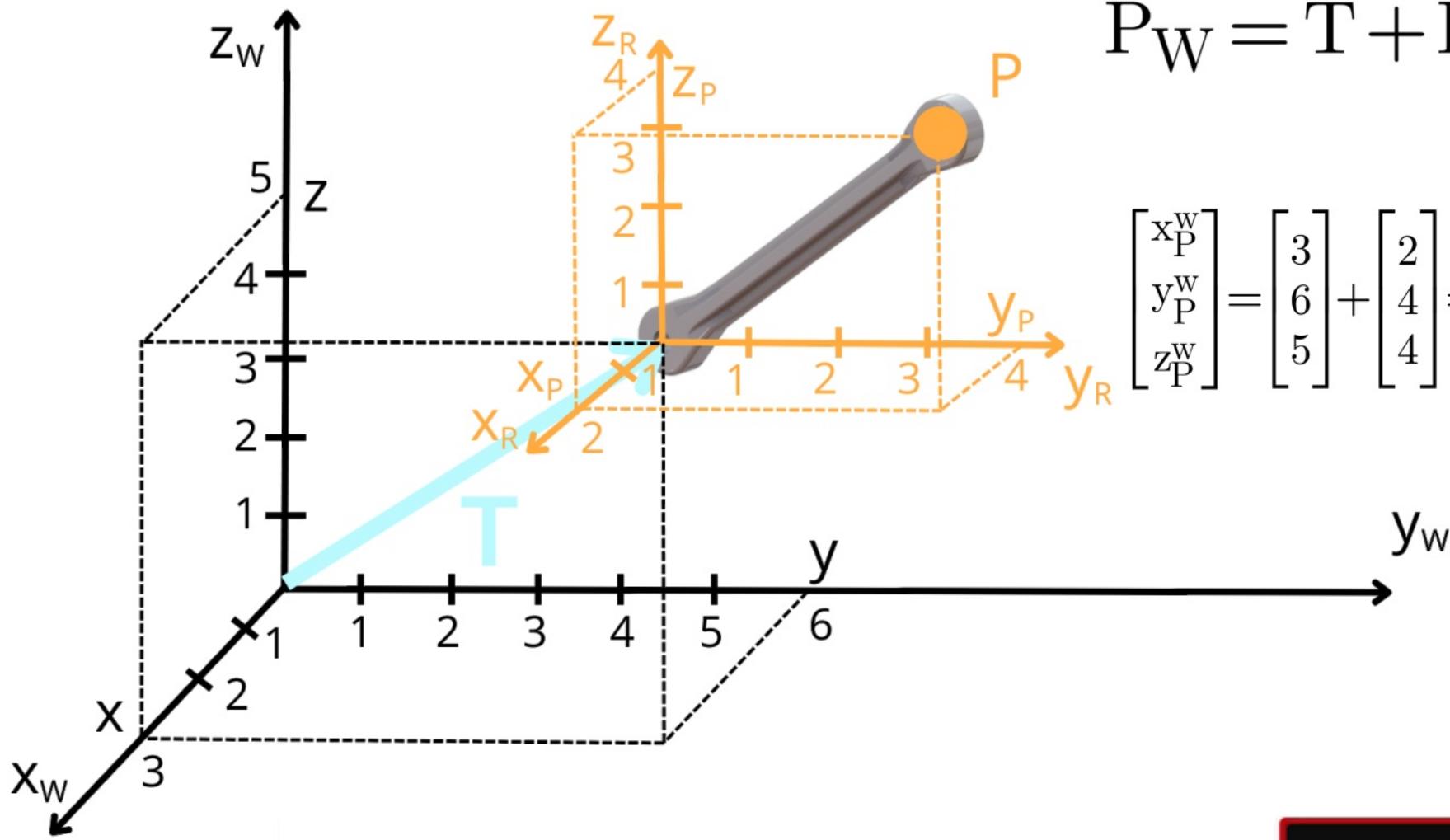


$$P_W = T + P_R$$



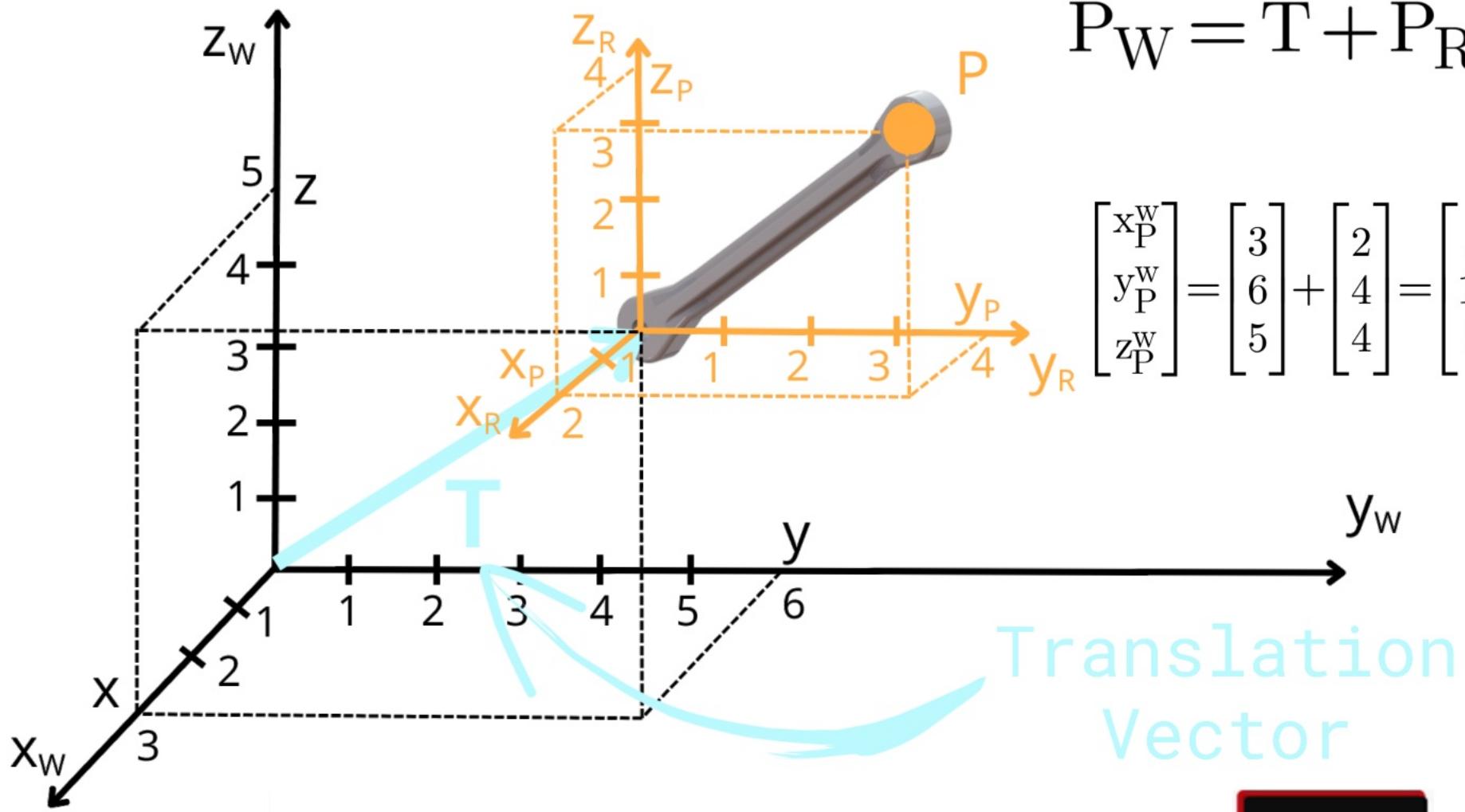
$$P_W = T + P_R$$

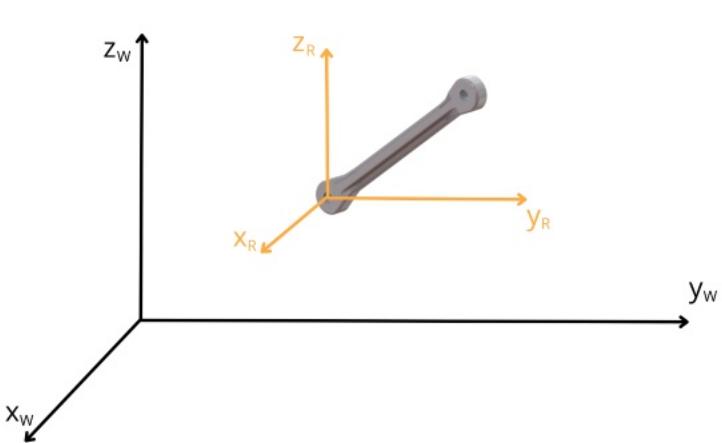
$$\begin{bmatrix} x_P^W \\ y_P^W \\ z_P^W \end{bmatrix} = \begin{bmatrix} 3 \\ 6 \\ 5 \end{bmatrix} + \begin{bmatrix} 2 \\ 4 \\ 4 \end{bmatrix} = \begin{bmatrix} 5 \\ 10 \\ 9 \end{bmatrix}$$



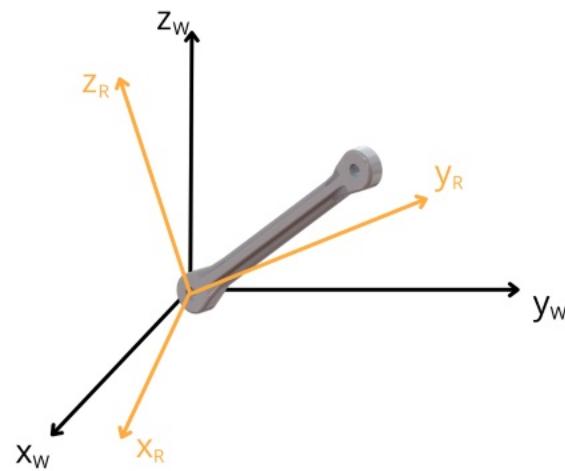
$$P_W = T + P_R$$

$$\begin{bmatrix} x_P^W \\ y_P^W \\ z_P^W \end{bmatrix} = \begin{bmatrix} 3 \\ 6 \\ 5 \end{bmatrix} + \begin{bmatrix} 2 \\ 4 \\ 4 \end{bmatrix} = \begin{bmatrix} 5 \\ 10 \\ 9 \end{bmatrix}$$

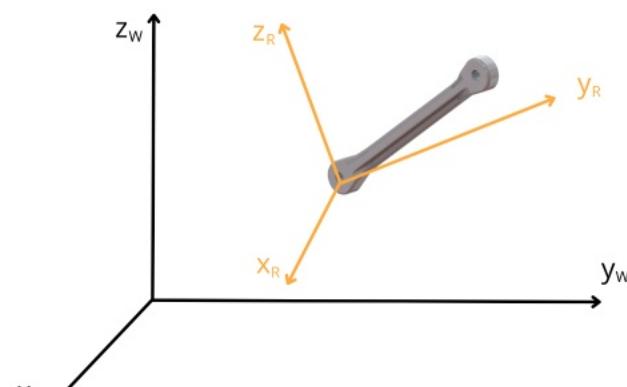




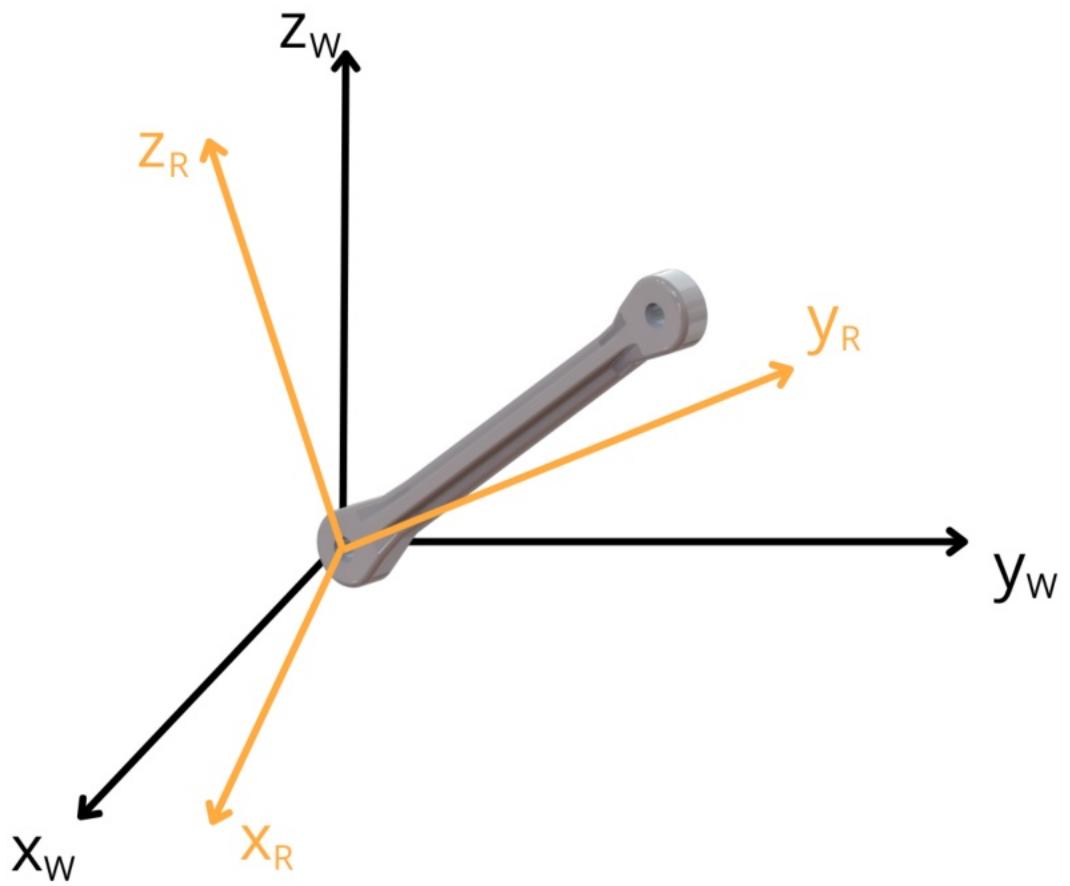
Translation

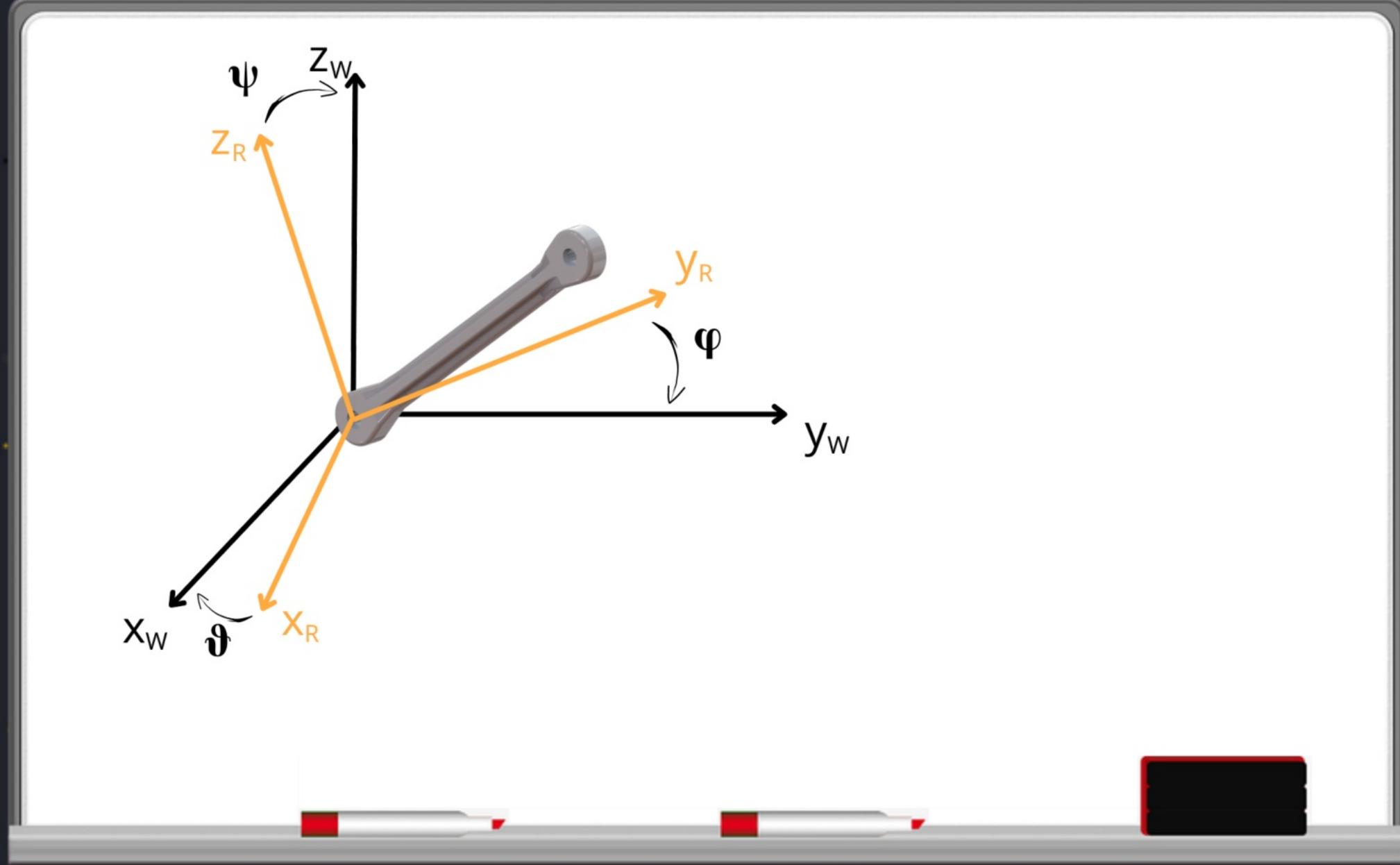
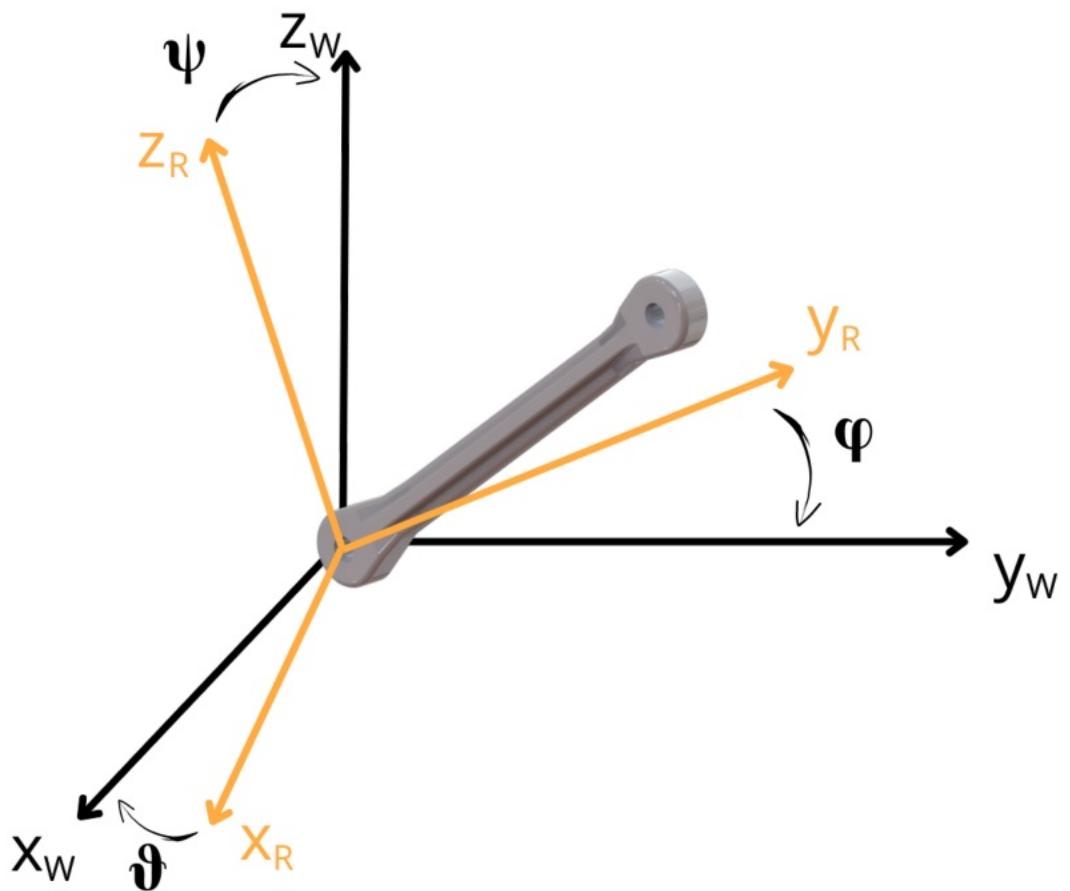


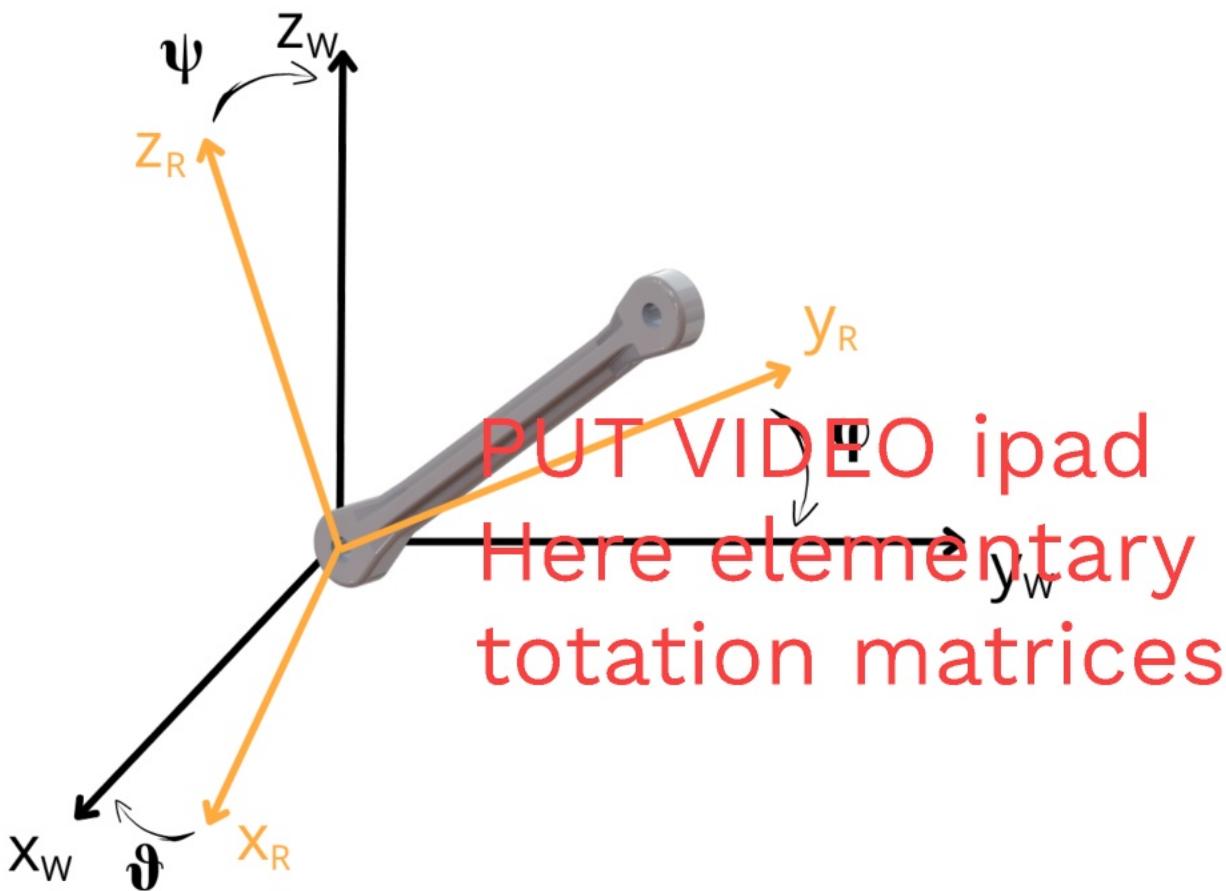
Rotation

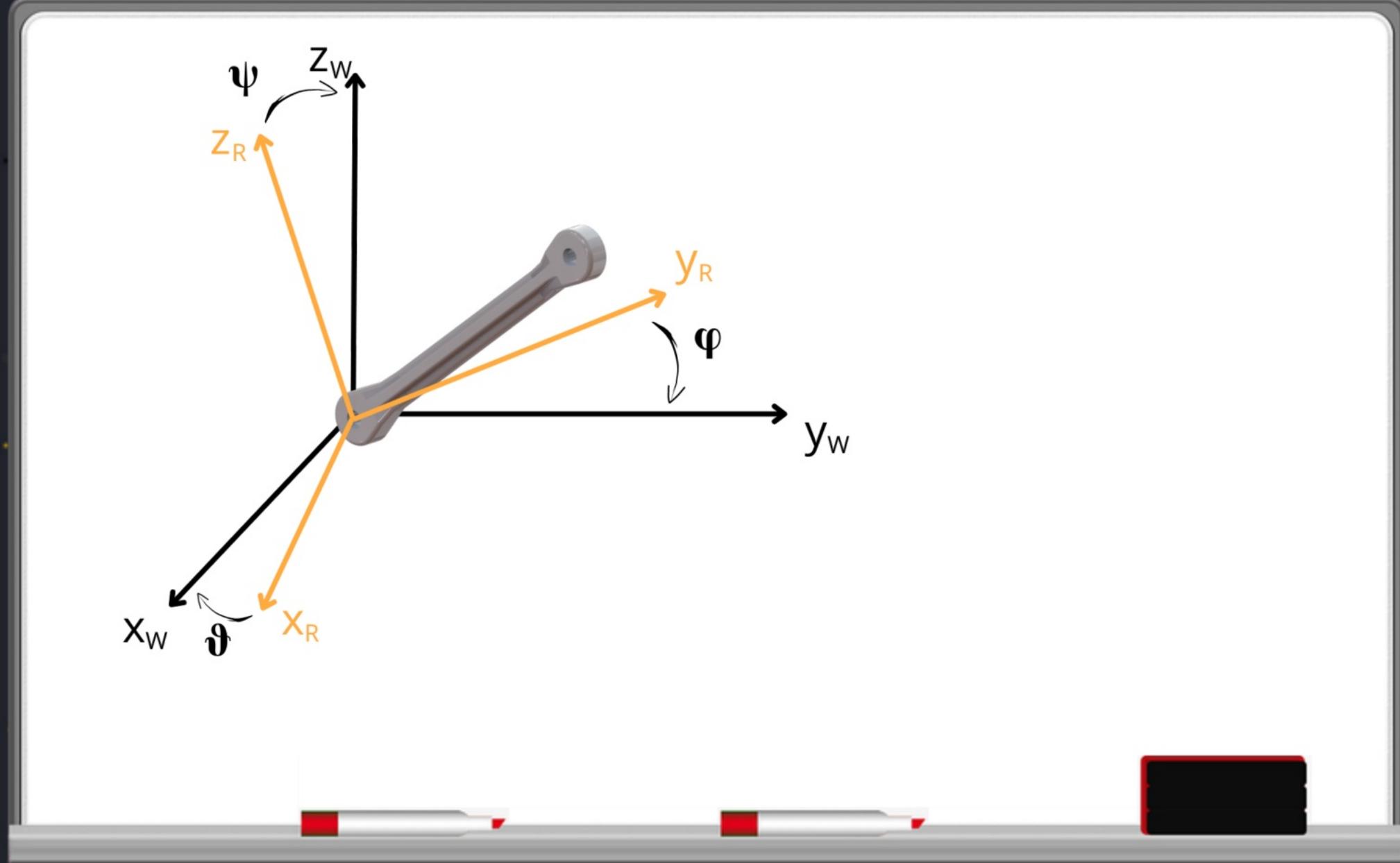
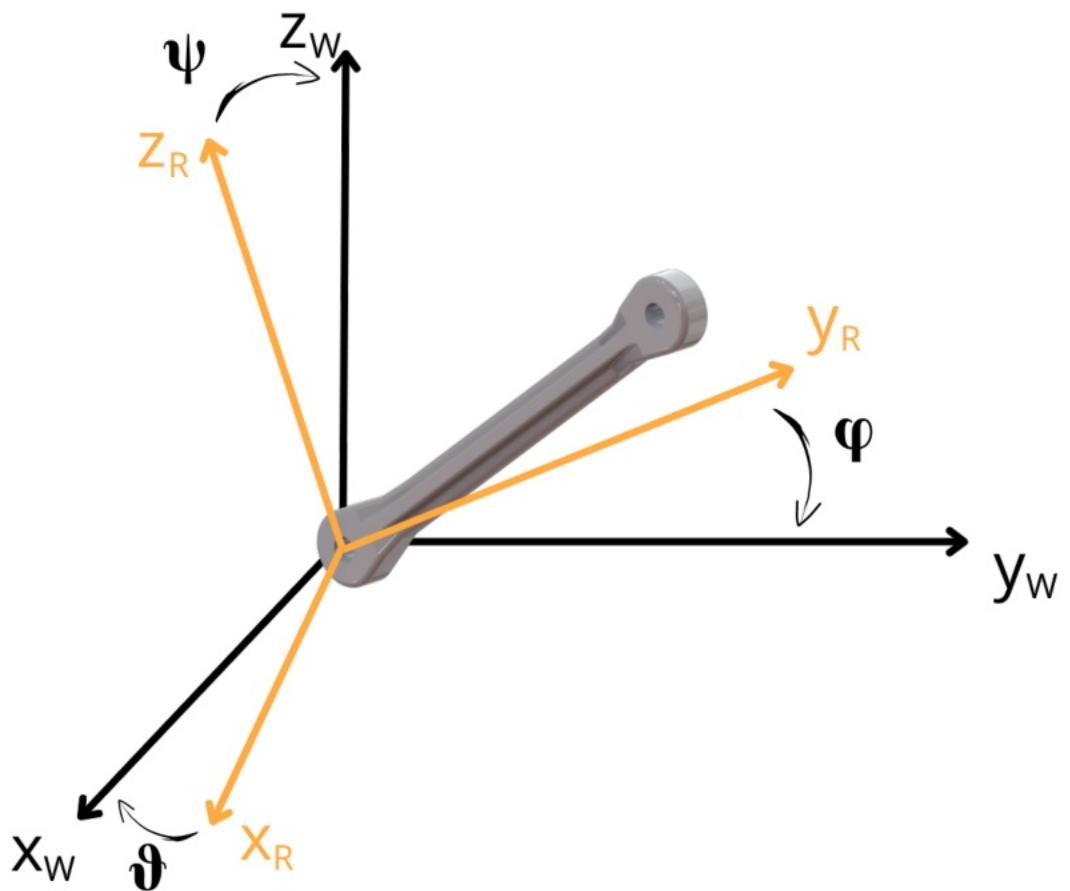


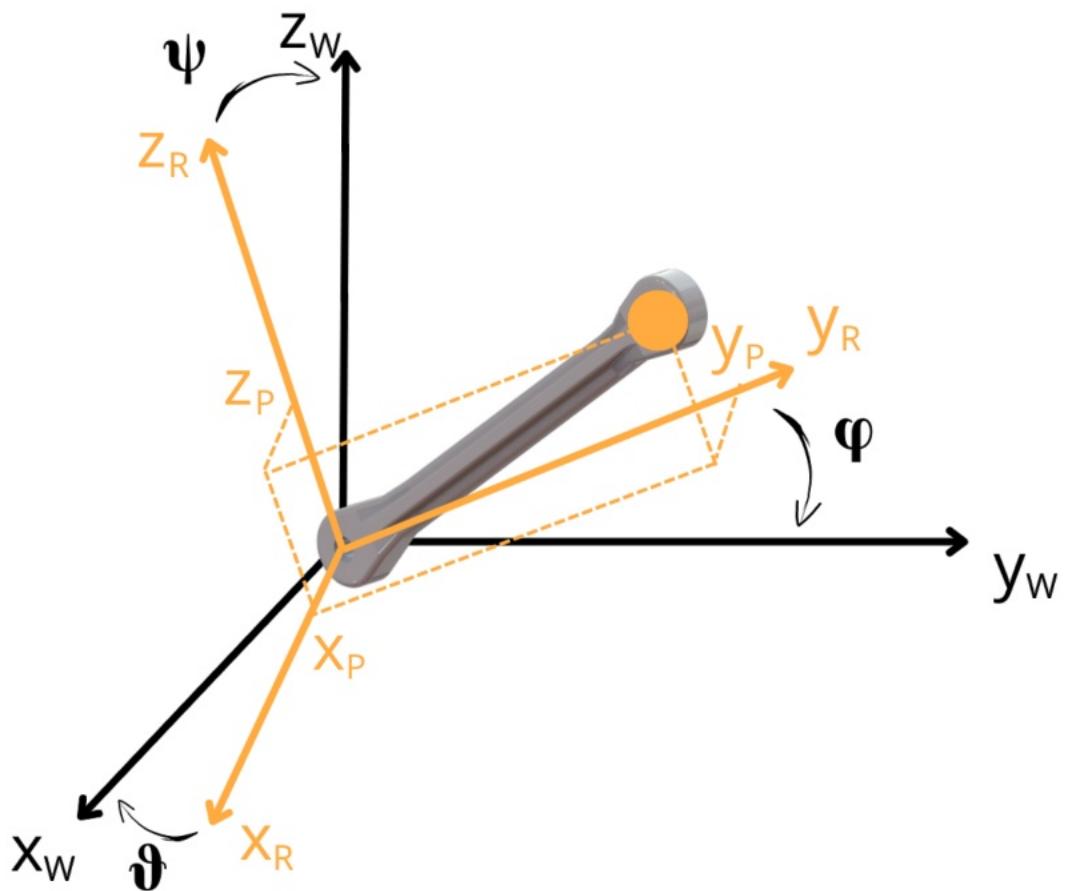
Roto-Translation

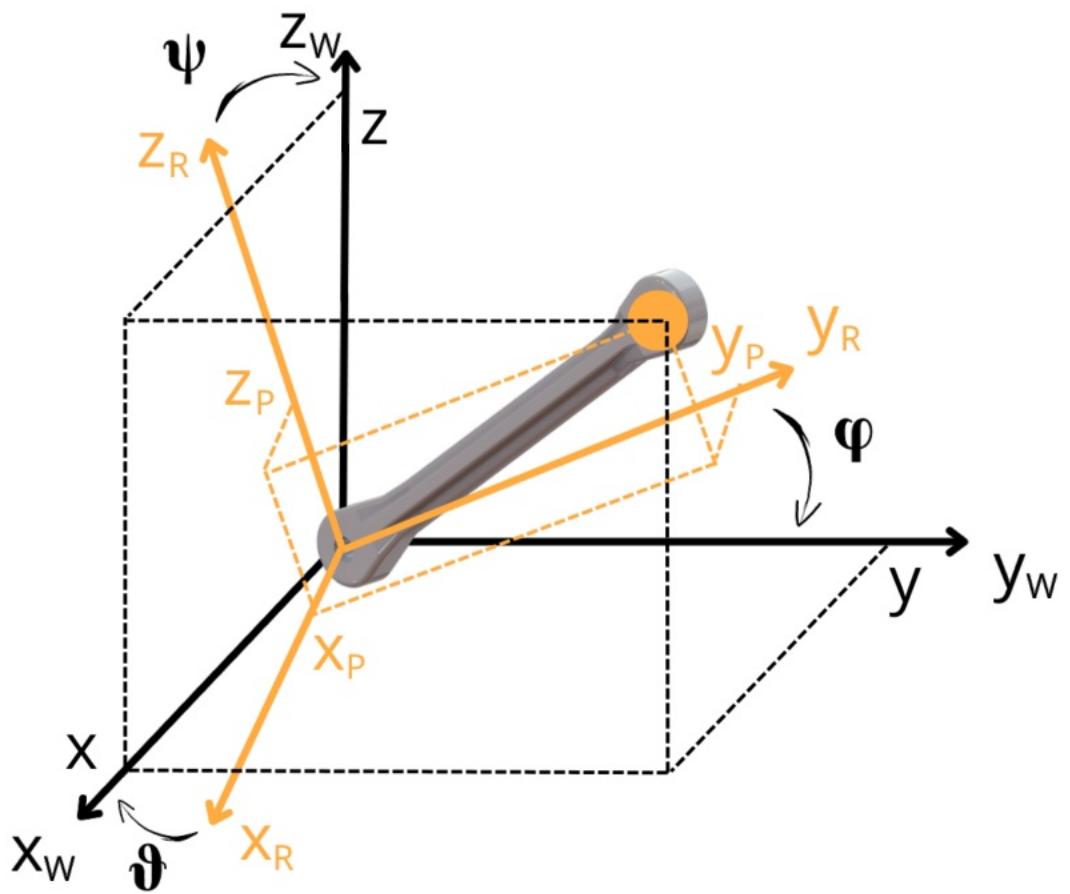


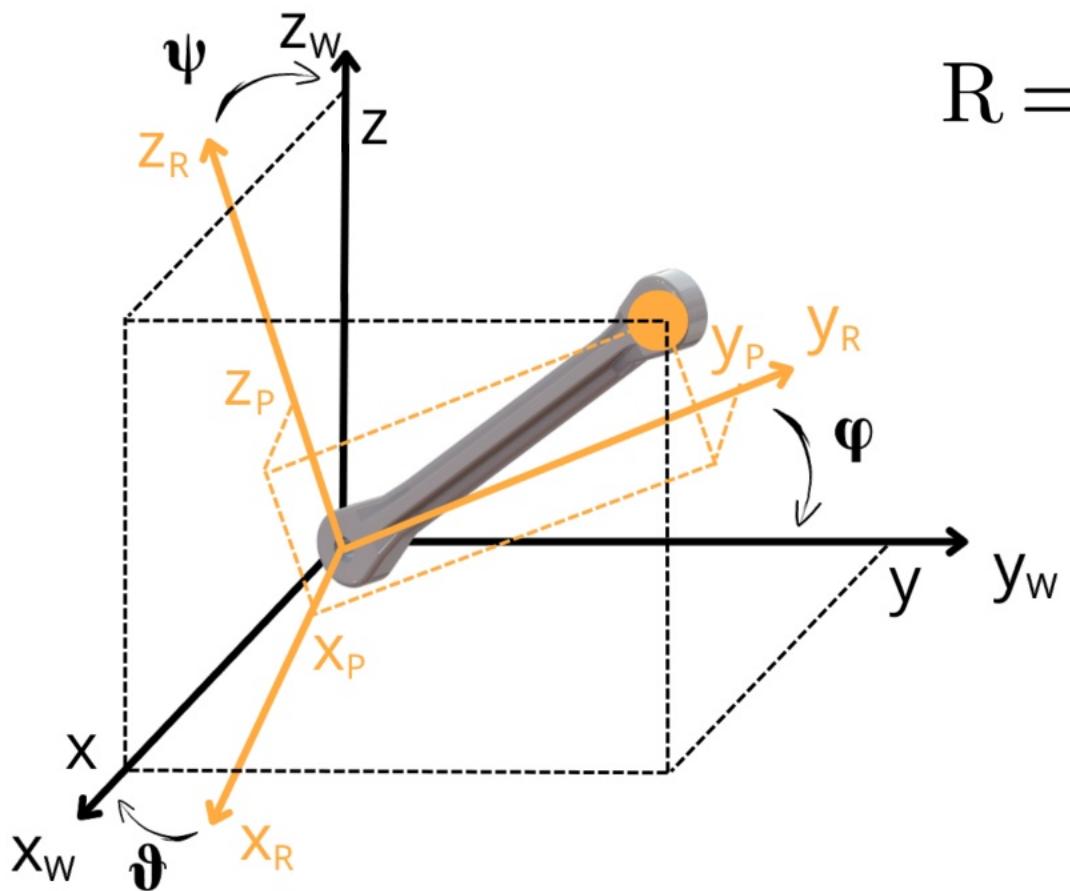




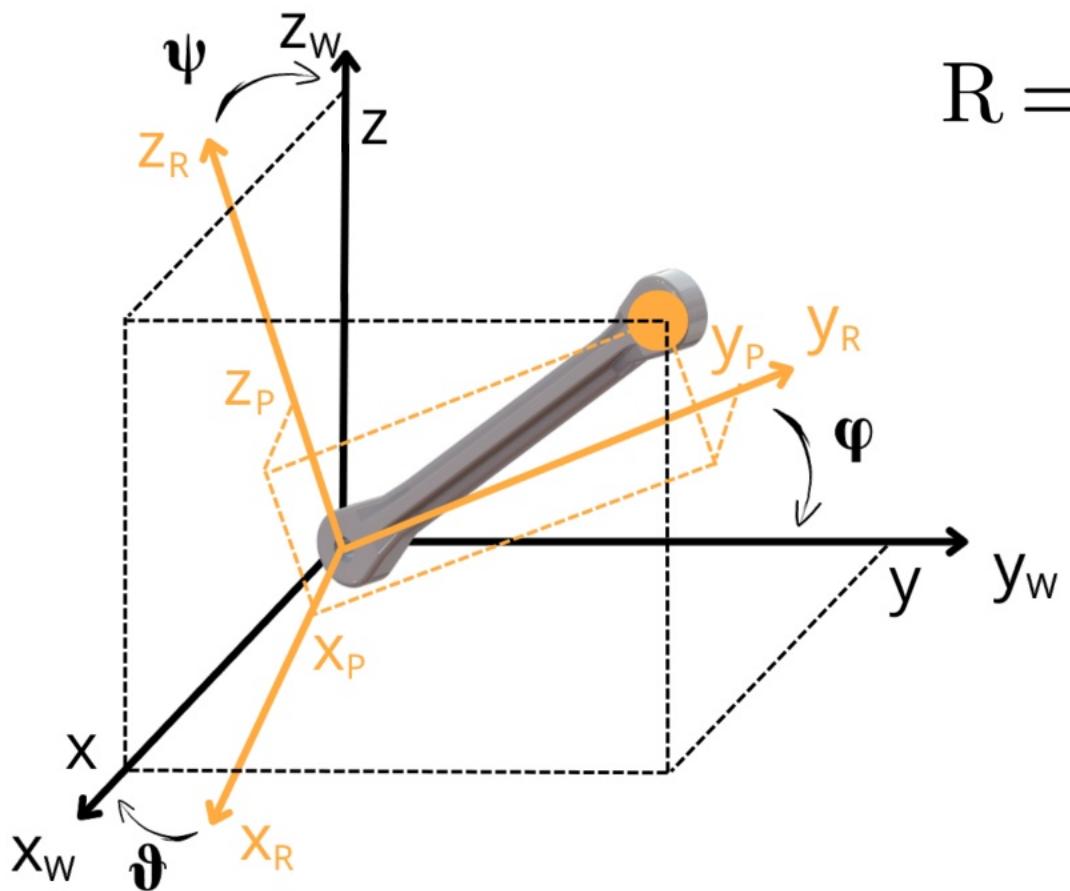






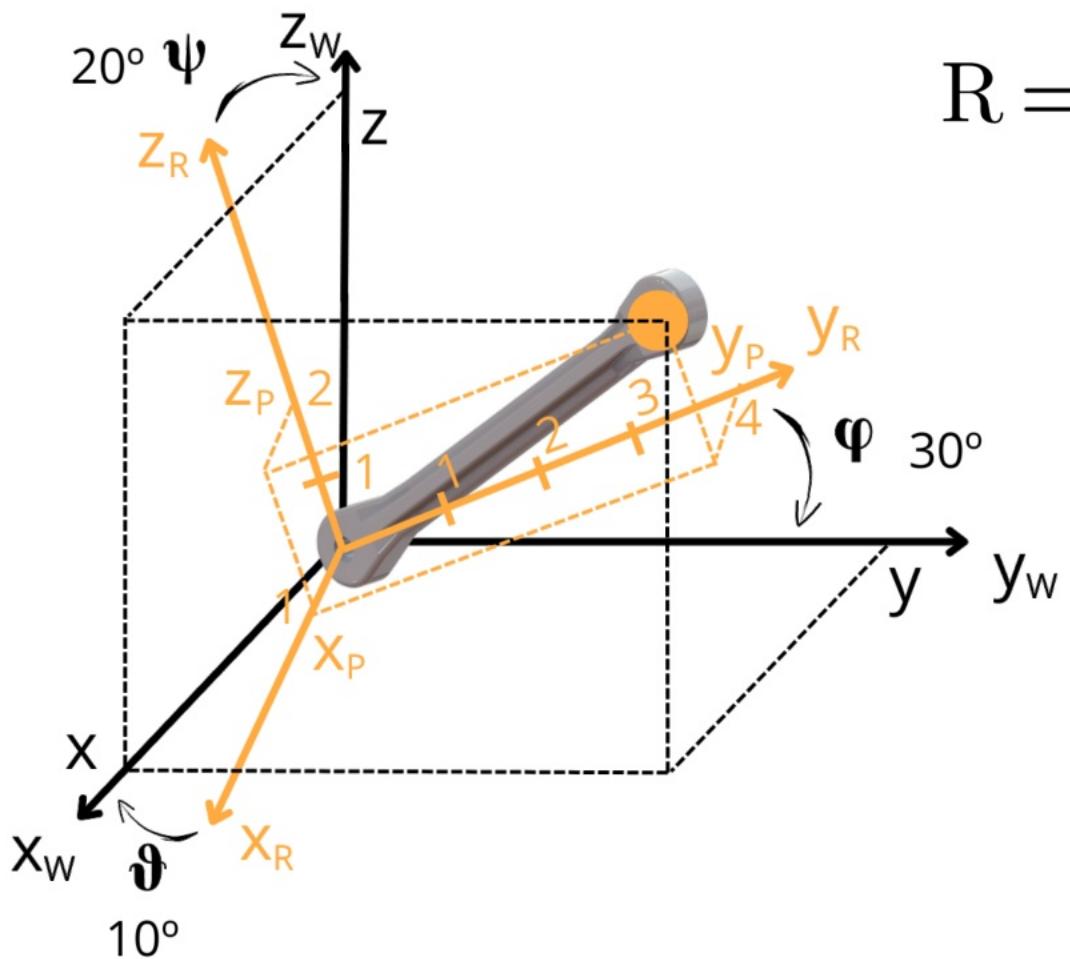


$$R = R_z(\psi) \cdot R_y(\varphi) \cdot R_x(\vartheta)$$



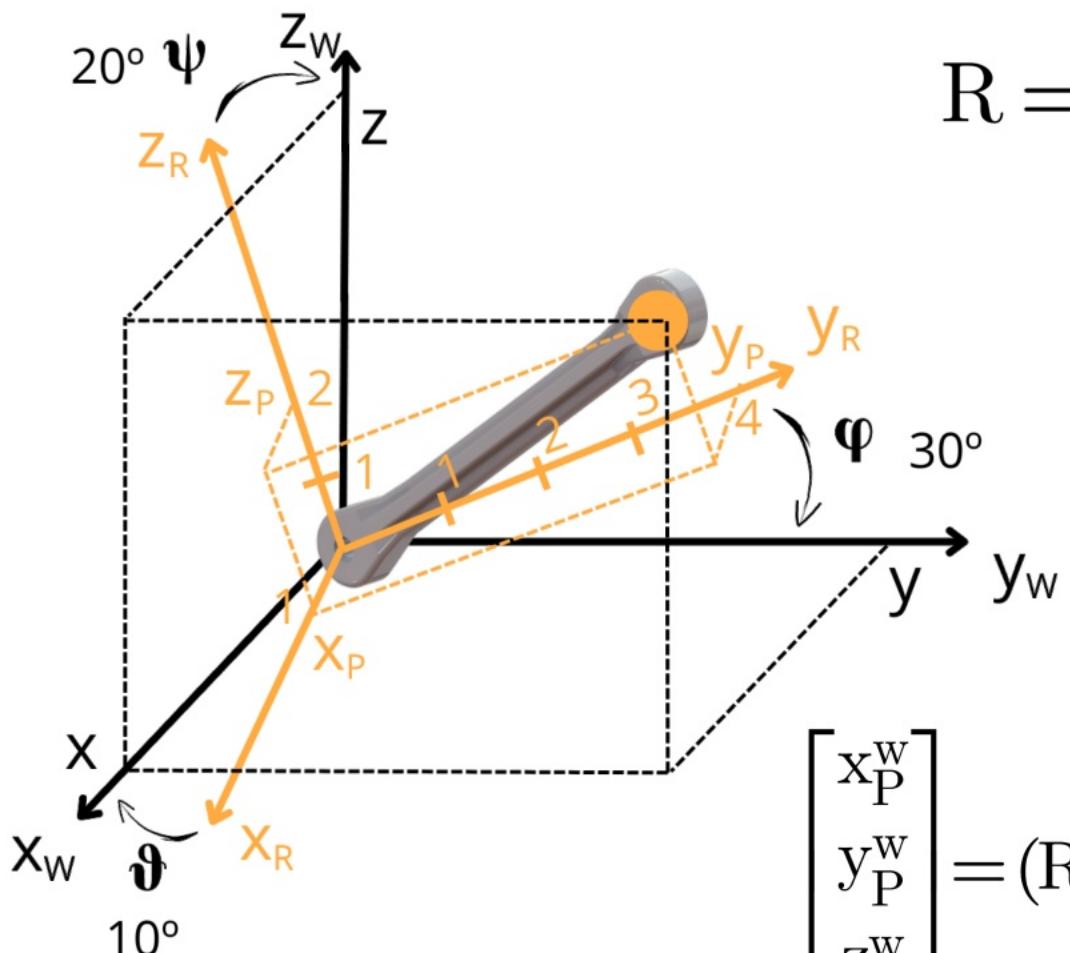
$$R = R_z(\psi) \cdot R_y(\varphi) \cdot R_x(\vartheta)$$

$$\begin{bmatrix} x_P^w \\ y_P^w \\ z_P^w \end{bmatrix} = R \cdot \begin{bmatrix} x_P \\ y_P \\ z_P \end{bmatrix}$$



$$R = R_z(\psi) \cdot R_y(\varphi) \cdot R_x(\vartheta)$$

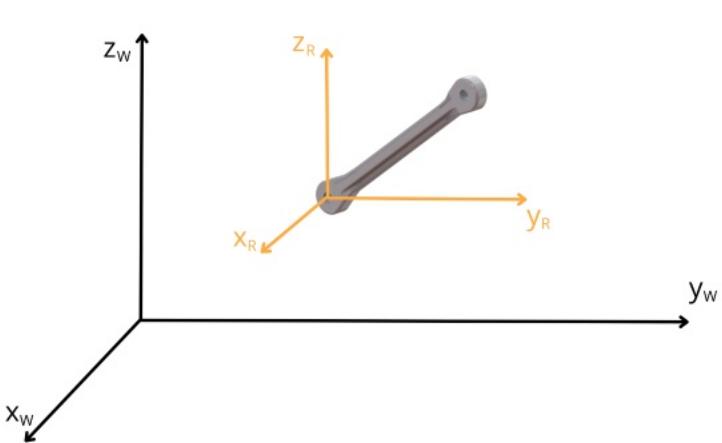
$$\begin{bmatrix} x_P^w \\ y_P^w \\ z_P^w \end{bmatrix} = R \cdot \begin{bmatrix} x_P \\ y_P \\ z_P \end{bmatrix}$$



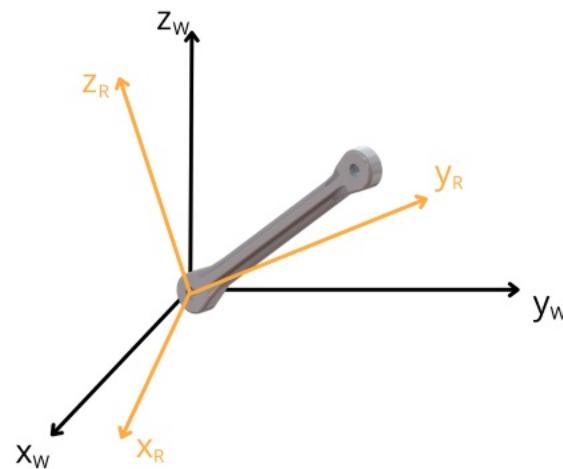
$$R = R_z(\psi) \cdot R_y(\varphi) \cdot R_x(\theta)$$

$$\begin{bmatrix} x_P^w \\ y_P^w \\ z_P^w \end{bmatrix} = R \cdot \begin{bmatrix} x_P \\ y_P \\ z_P \end{bmatrix}$$

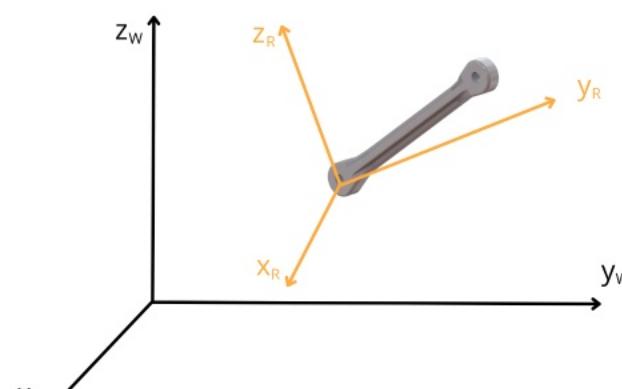
$$\begin{bmatrix} x_P^w \\ y_P^w \\ z_P^w \end{bmatrix} = (R_z(20^\circ) \cdot R_y(30^\circ) \cdot R_x(10^\circ)) \cdot \begin{bmatrix} 1 \\ 4 \\ 2 \end{bmatrix}$$



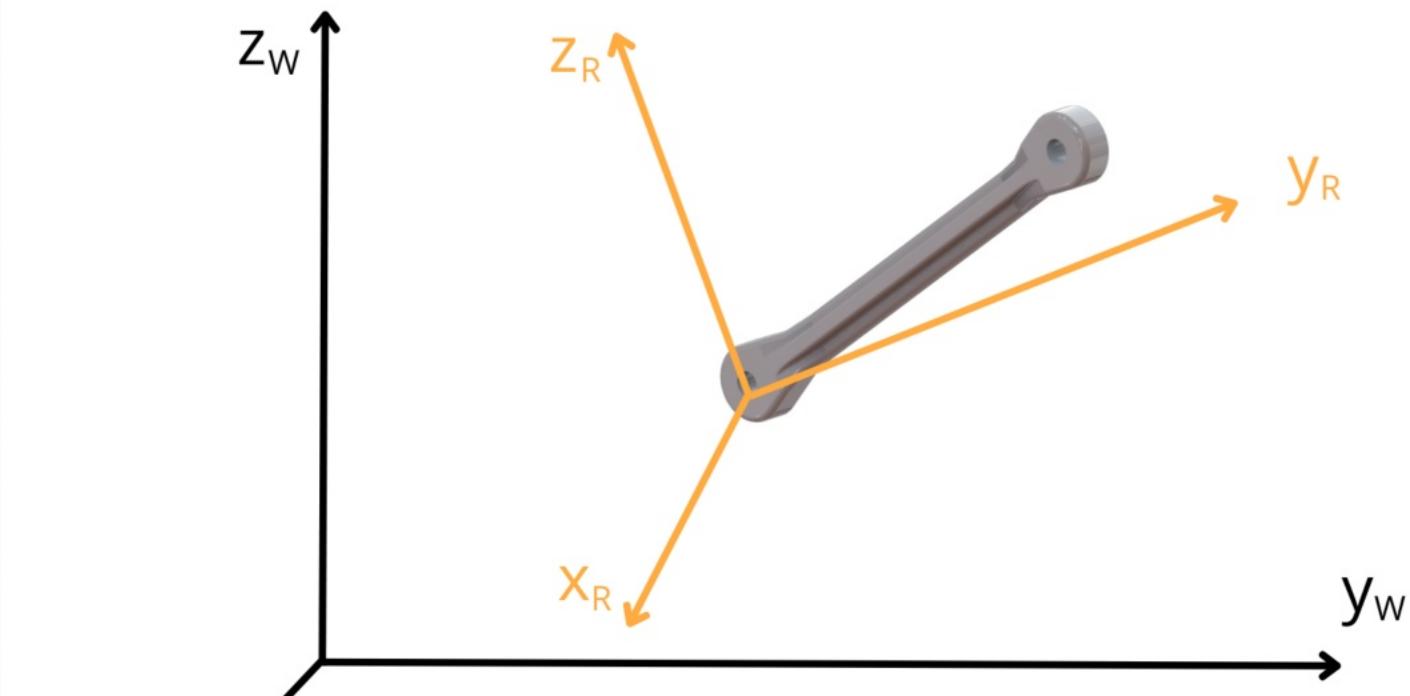
Translation

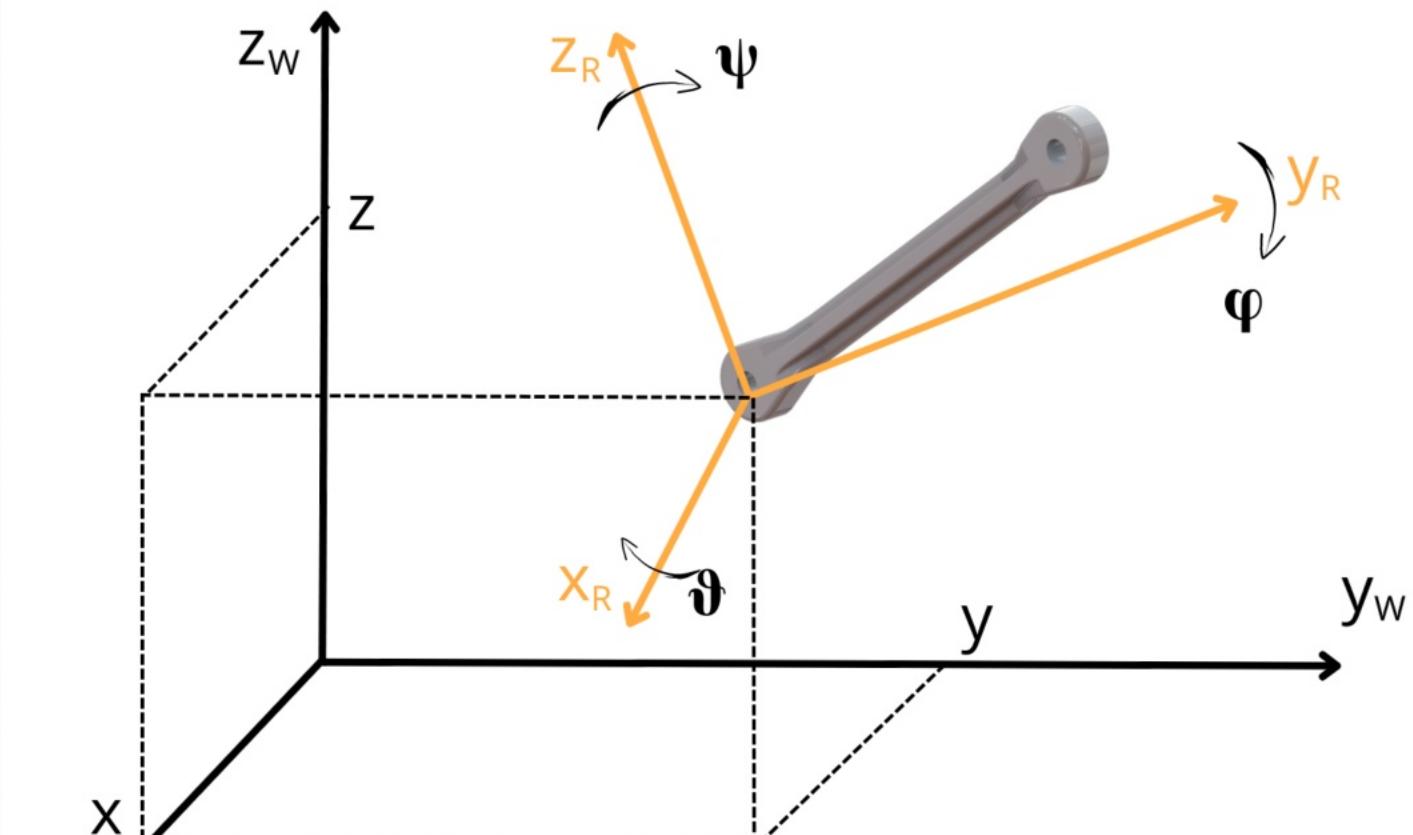


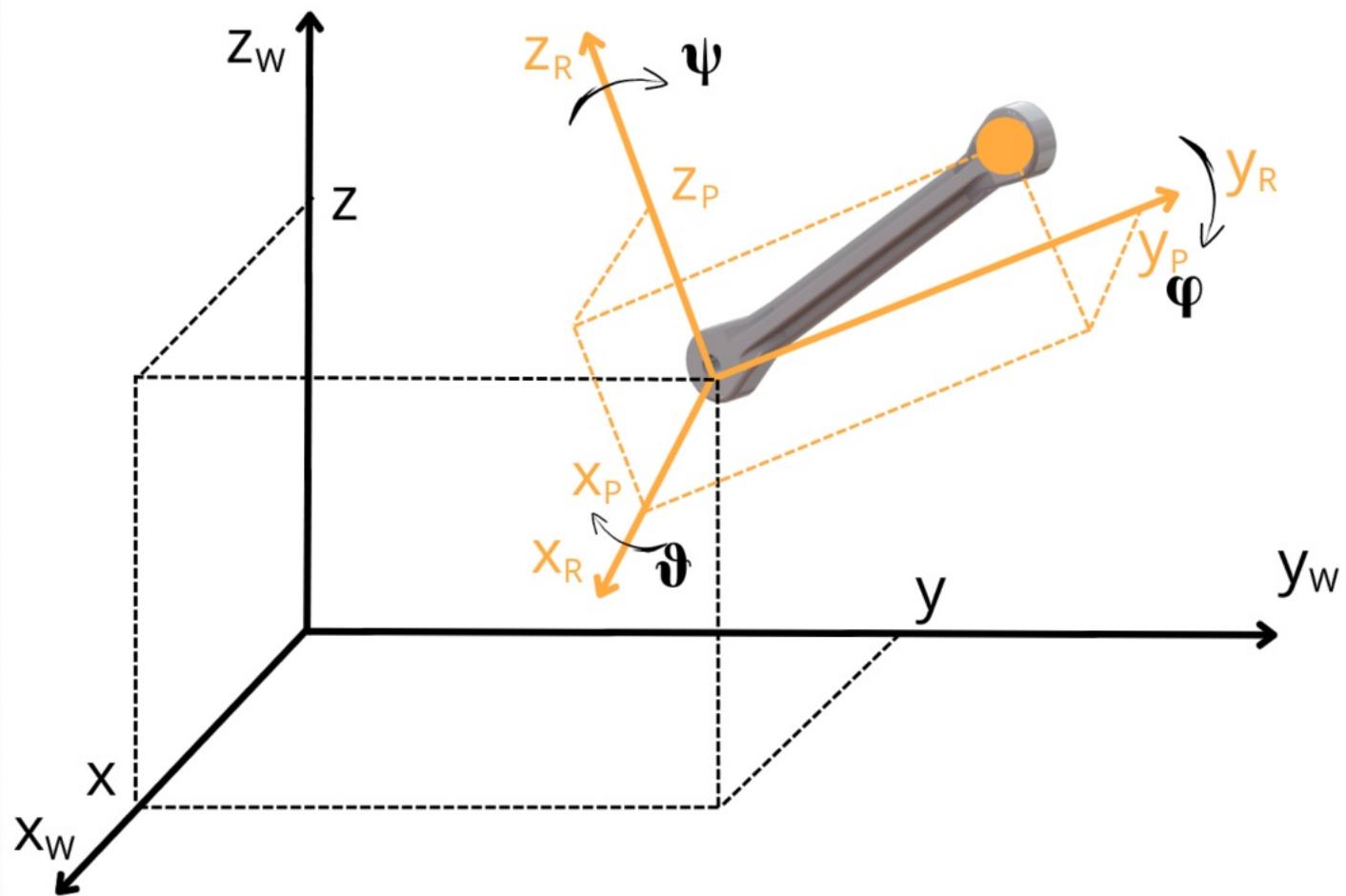
Rotation

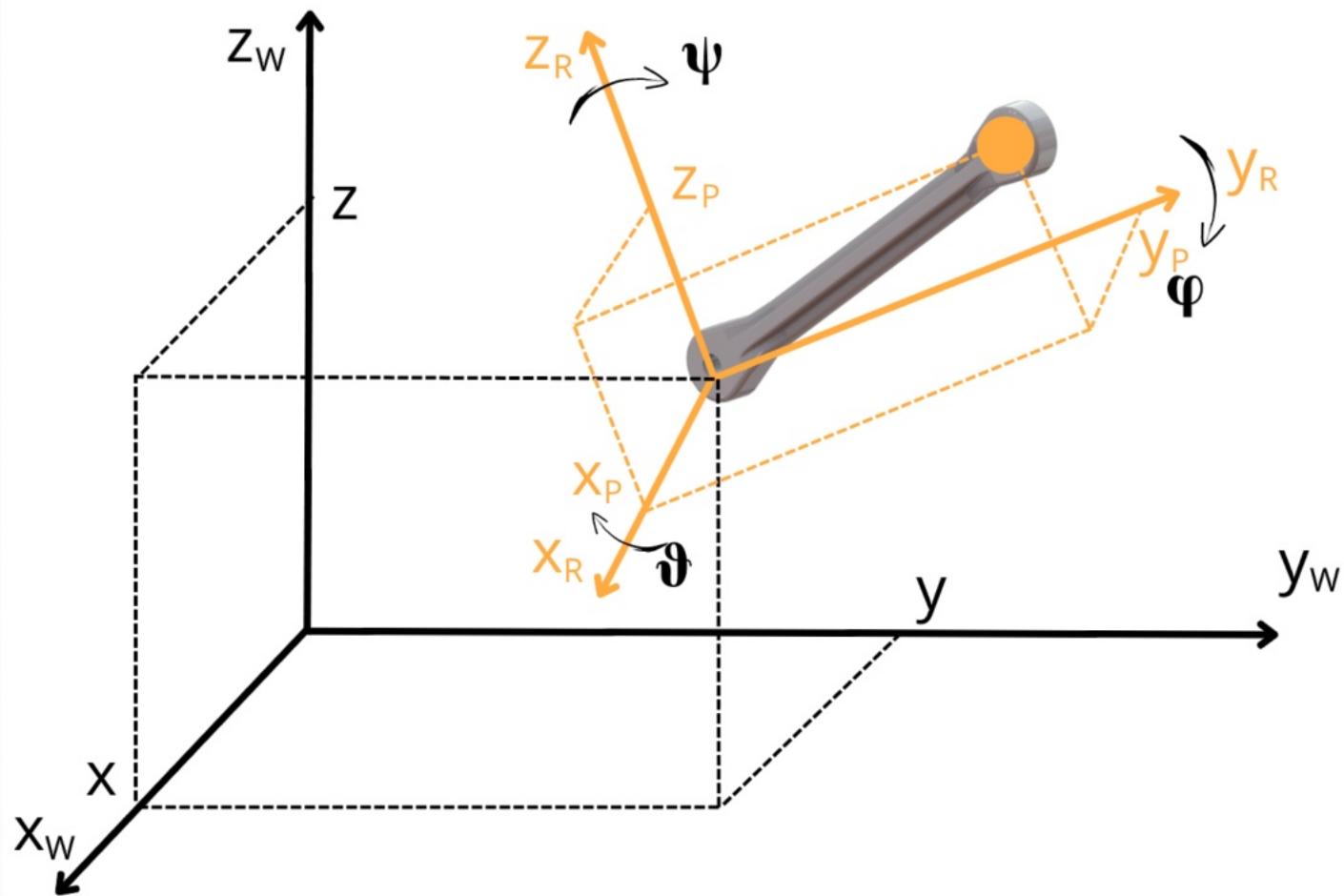


Roto-Translation



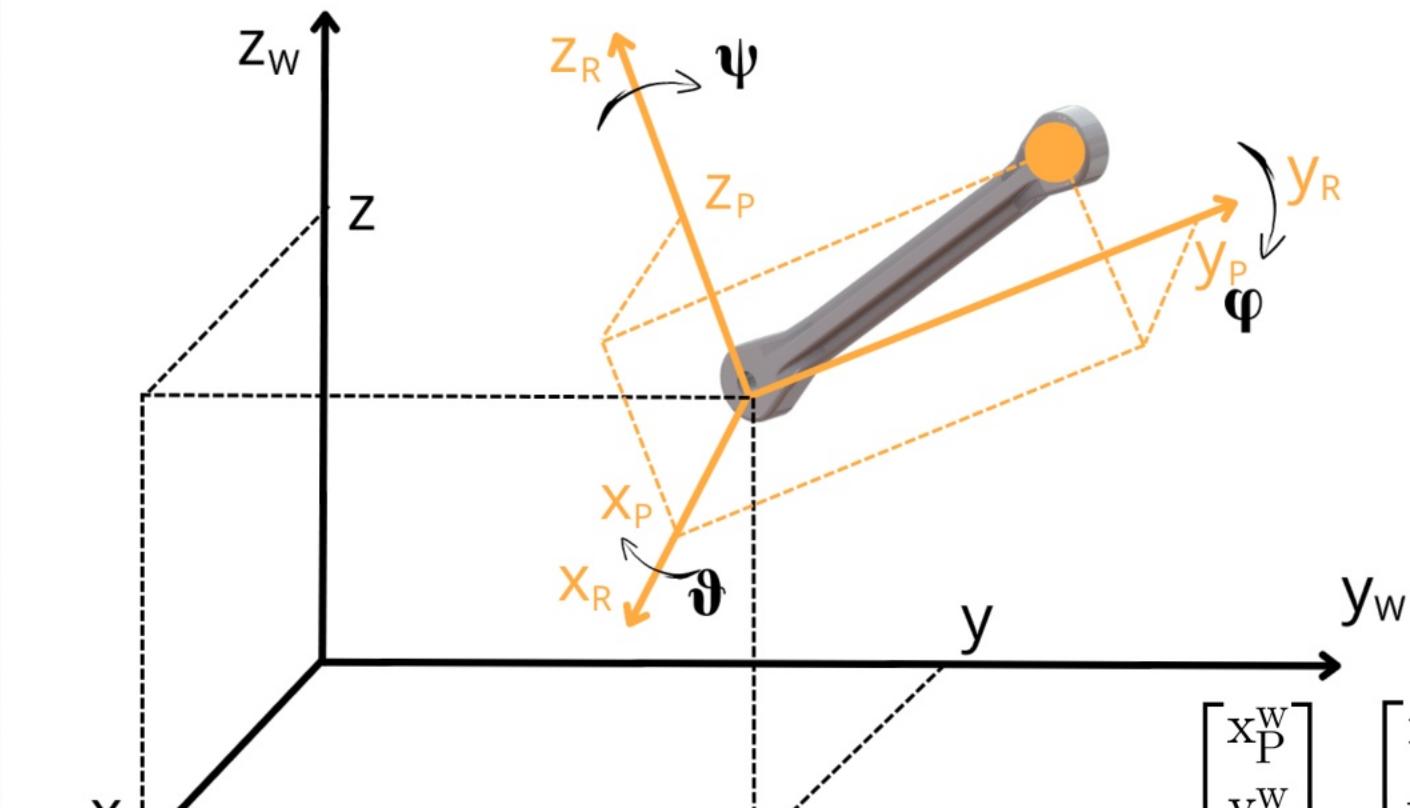






$$\begin{bmatrix} x_P^w \\ y_P^w \\ z_P^w \end{bmatrix} = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} + \begin{bmatrix} x_P \\ y_P \\ z_P \end{bmatrix}$$

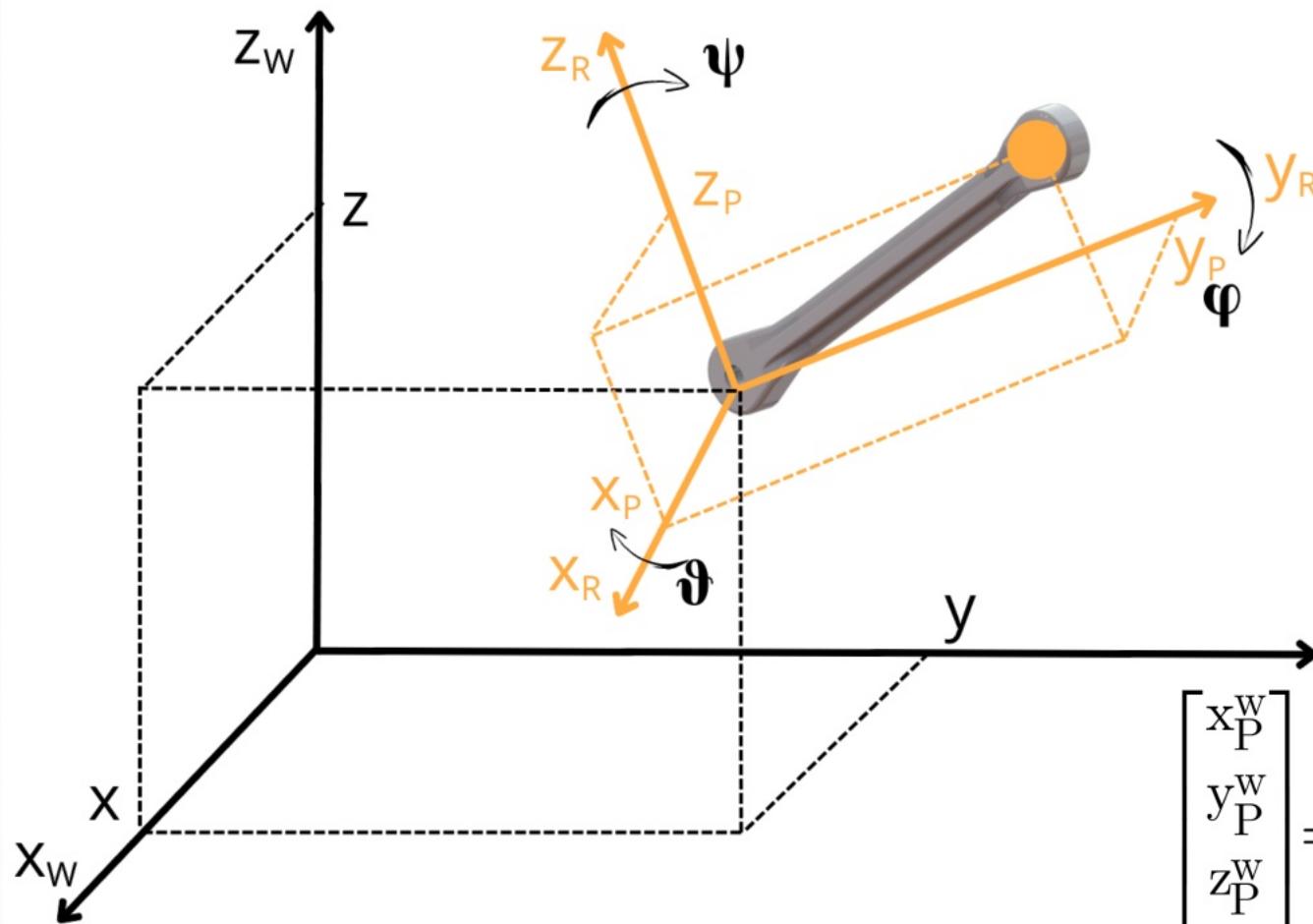
$$\begin{bmatrix} x_P^w \\ y_P^w \\ z_P^w \end{bmatrix} = R \cdot \begin{bmatrix} x_P \\ y_P \\ z_P \end{bmatrix}$$



$$\begin{bmatrix} x_P^w \\ y_P^w \\ z_P^w \end{bmatrix} = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} + \begin{bmatrix} x_P \\ y_P \\ z_P \end{bmatrix}$$

$$\begin{bmatrix} x_P^w \\ y_P^w \\ z_P^w \end{bmatrix} = R \cdot \begin{bmatrix} x_P \\ y_P \\ z_P \end{bmatrix}$$

$$\begin{bmatrix} x_P^w \\ y_P^w \\ z_P^w \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & T_x \\ r_{21} & r_{22} & r_{23} & T_y \\ r_{31} & r_{32} & r_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_P \\ y_P \\ z_P \\ 1 \end{bmatrix}$$

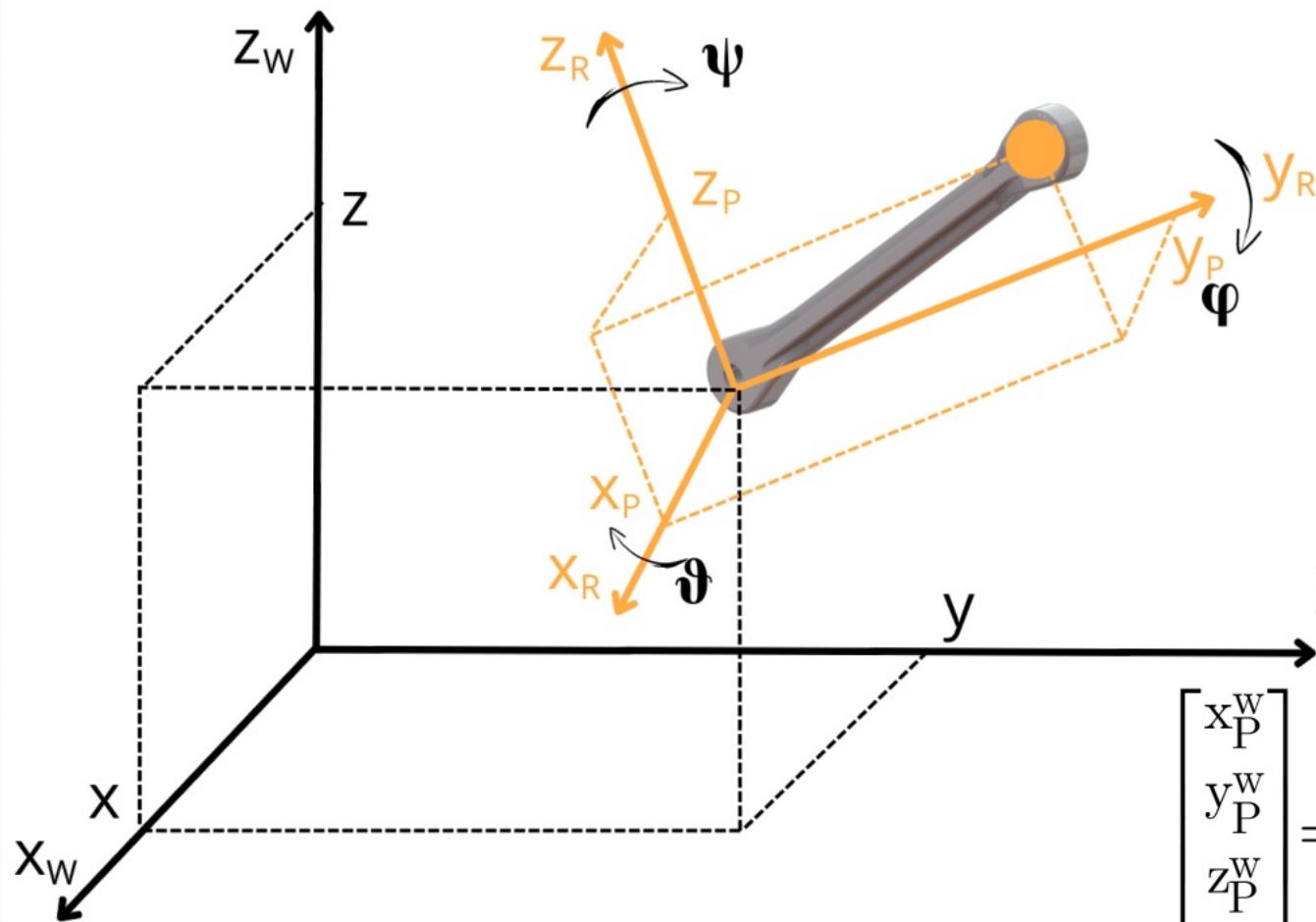


$$\begin{bmatrix} x_P^w \\ y_P^w \\ z_P^w \end{bmatrix} = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} + \begin{bmatrix} x_P \\ y_P \\ z_P \end{bmatrix}$$

$$\begin{bmatrix} x_P^w \\ y_P^w \\ z_P^w \end{bmatrix} = R \cdot \begin{bmatrix} x_P \\ y_P \\ z_P \end{bmatrix}$$

Rotation Matrix

$$\begin{bmatrix} x_P^w \\ y_P^w \\ z_P^w \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & T_x \\ r_{21} & r_{22} & r_{23} & T_y \\ r_{31} & r_{32} & r_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_P \\ y_P \\ z_P \\ 1 \end{bmatrix}$$



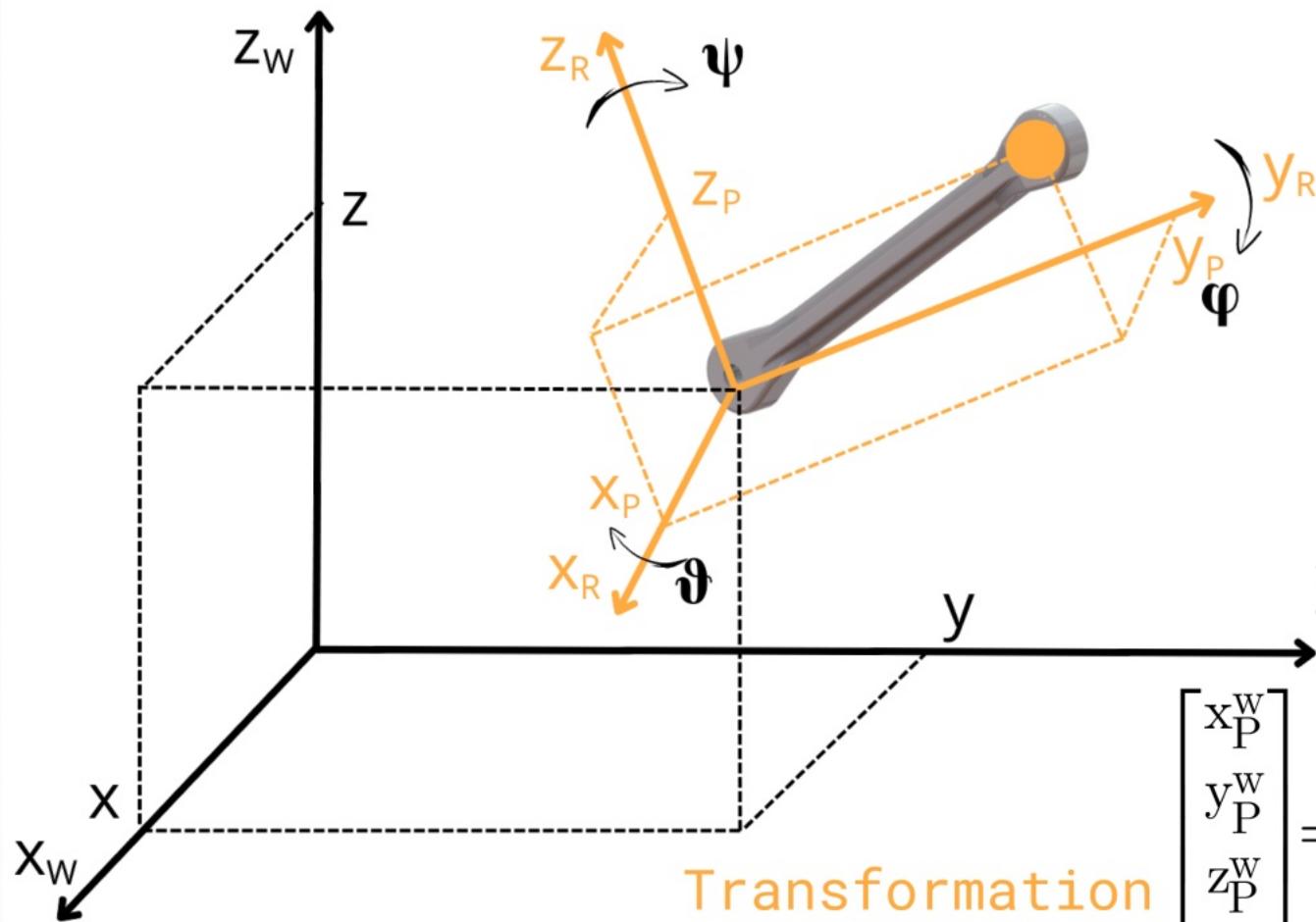
$$\begin{bmatrix} x_P^w \\ y_P^w \\ z_P^w \end{bmatrix} = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} + \begin{bmatrix} x_P \\ y_P \\ z_P \end{bmatrix}$$

$$\begin{bmatrix} x_P^w \\ y_P^w \\ z_P^w \end{bmatrix} = R \cdot \begin{bmatrix} x_P \\ y_P \\ z_P \end{bmatrix}$$

Rotation Matrix

$$\begin{bmatrix} x_P^w \\ y_P^w \\ z_P^w \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & T_x \\ r_{21} & r_{22} & r_{23} & T_y \\ r_{31} & r_{32} & r_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_P \\ y_P \\ z_P \\ 1 \end{bmatrix}$$

Translation Vector



$$\begin{bmatrix} x_P^w \\ y_P^w \\ z_P^w \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & T_x \\ r_{21} & r_{22} & r_{23} & T_y \\ r_{31} & r_{32} & r_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_P \\ y_P \\ z_P \\ 1 \end{bmatrix}$$

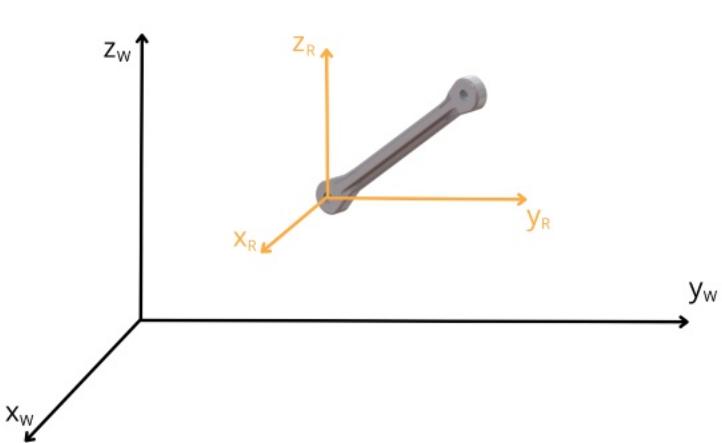
Transformation Matrix

Rotation Matrix

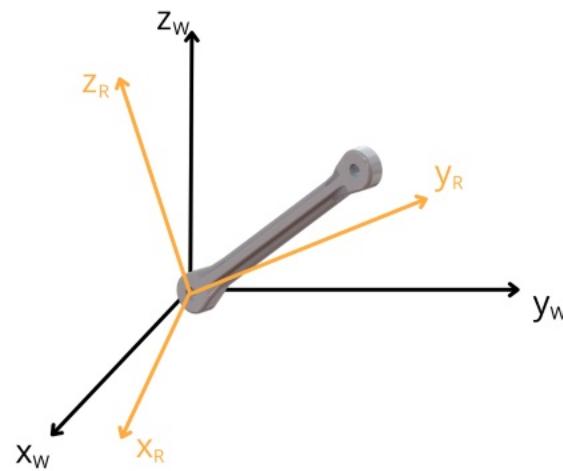
Translation Vector

$$\begin{bmatrix} x_P^w \\ y_P^w \\ z_P^w \end{bmatrix} = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} + \begin{bmatrix} x_P \\ y_P \\ z_P \end{bmatrix}$$

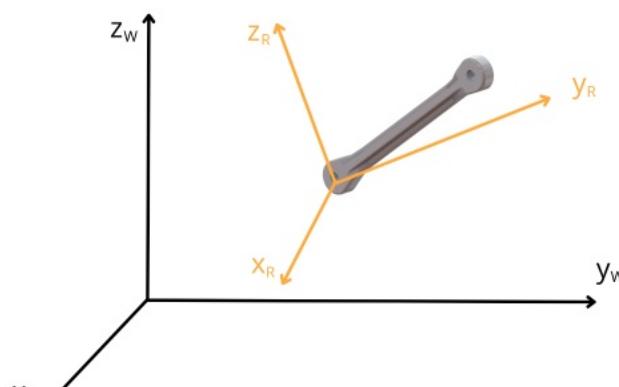
$$\begin{bmatrix} x_P^w \\ y_P^w \\ z_P^w \end{bmatrix} = R \cdot \begin{bmatrix} x_P \\ y_P \\ z_P \end{bmatrix}$$



Translation



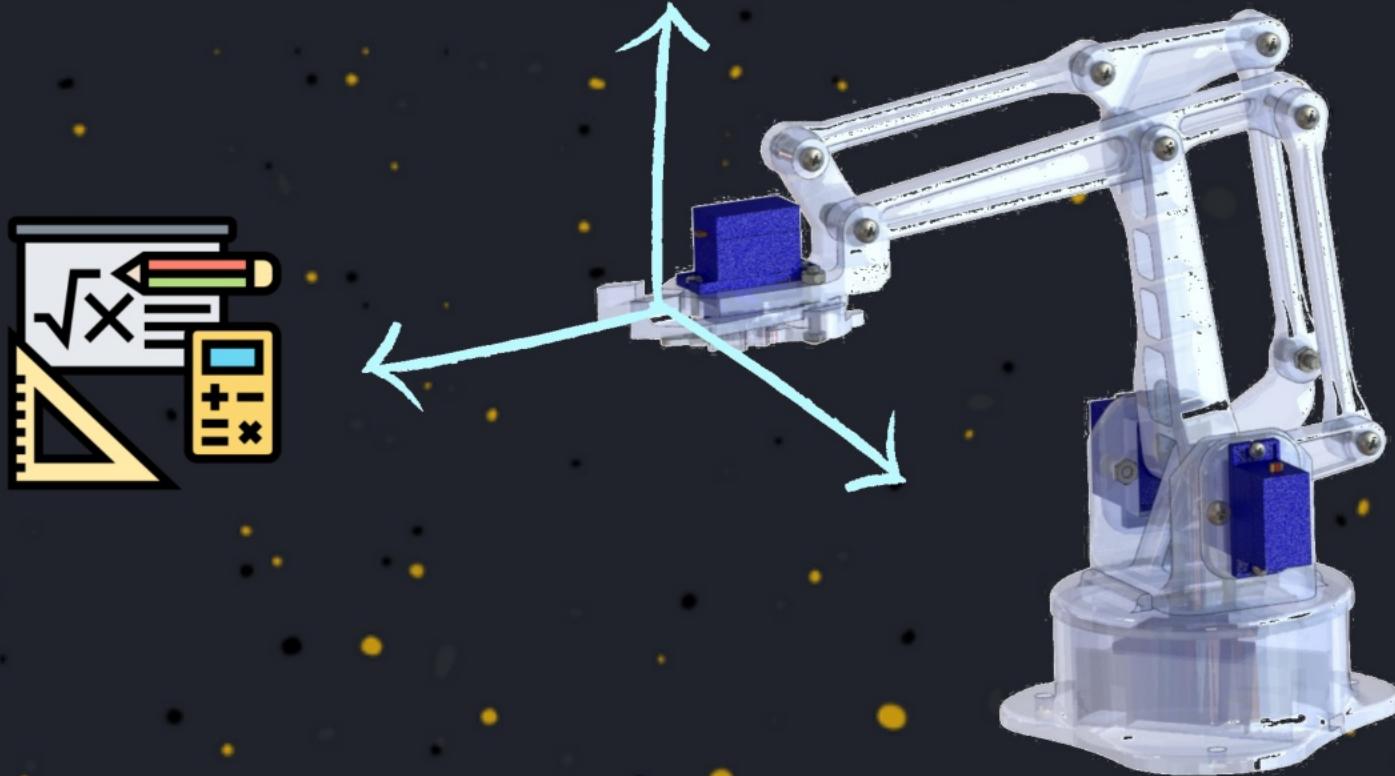
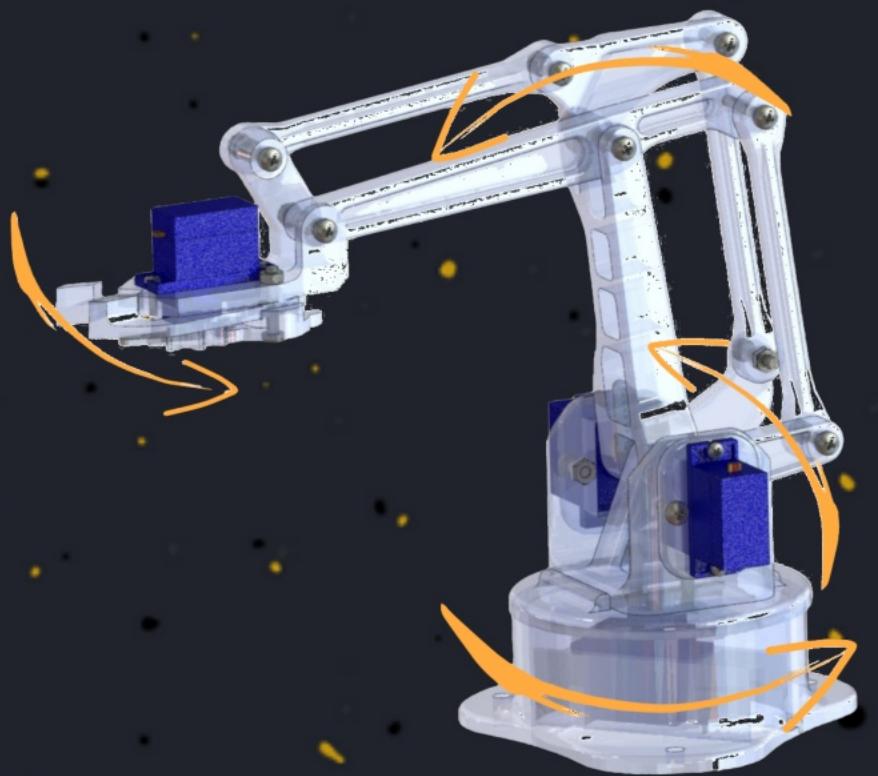
Rotation



Roto-Translation

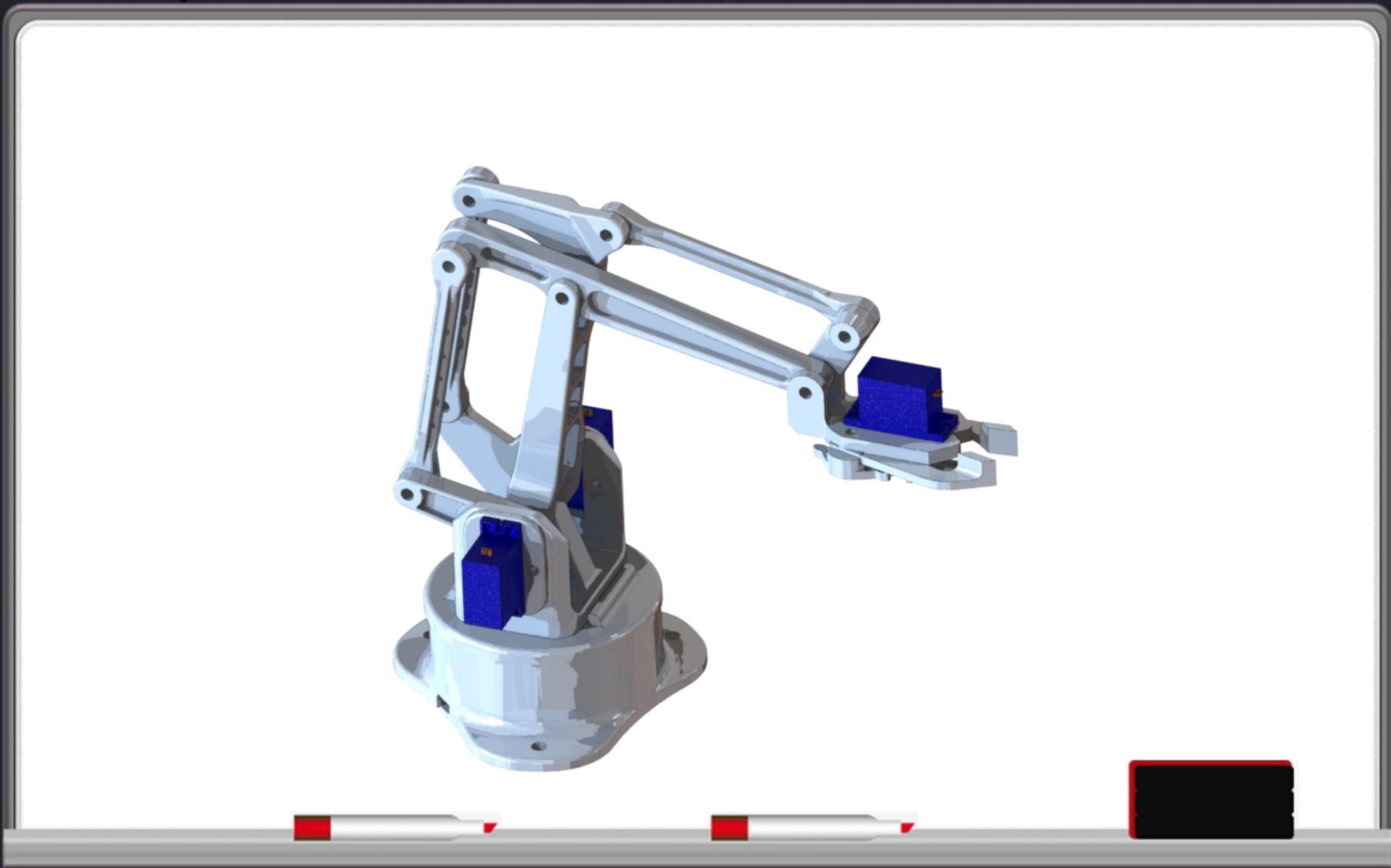


Forward

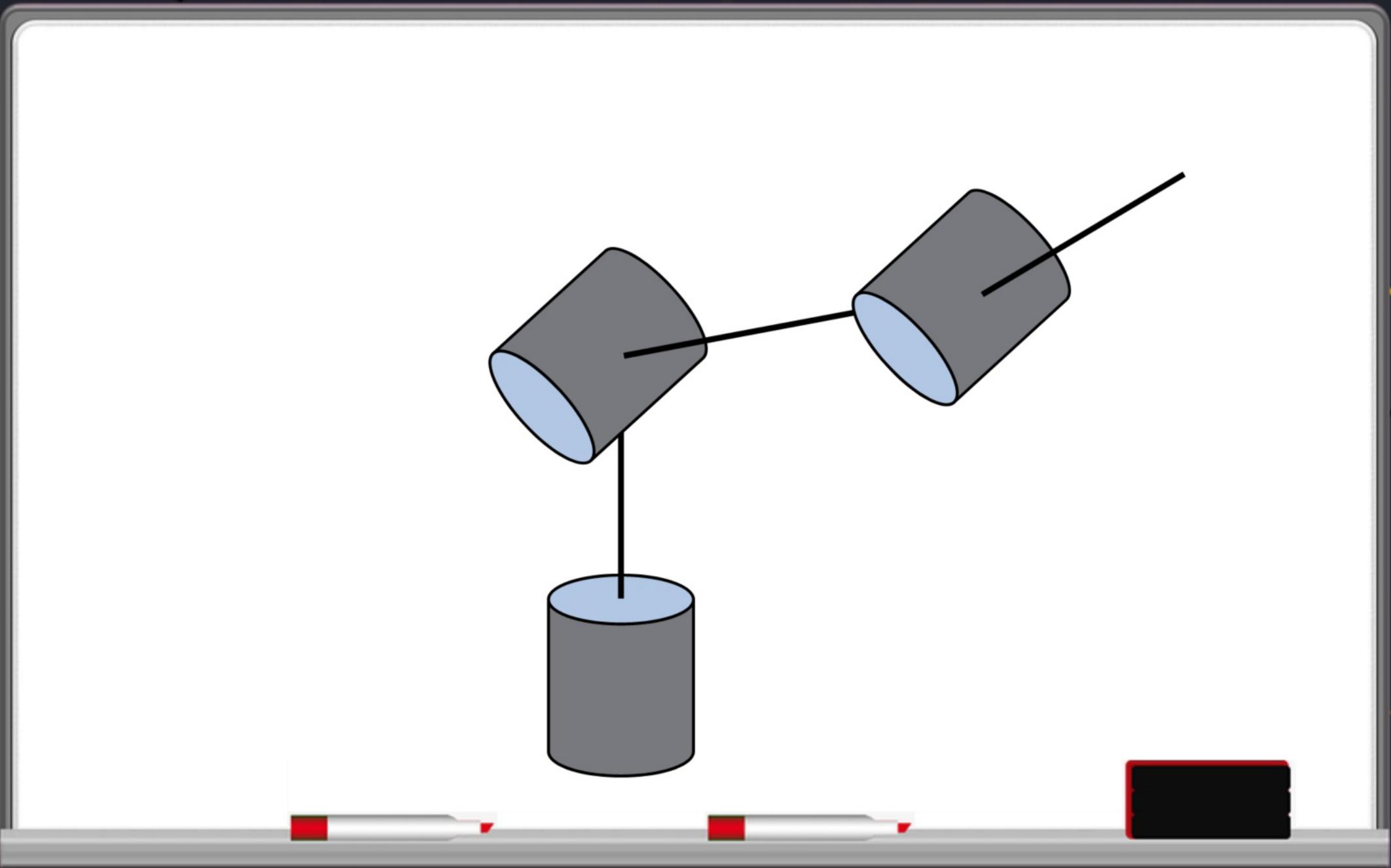


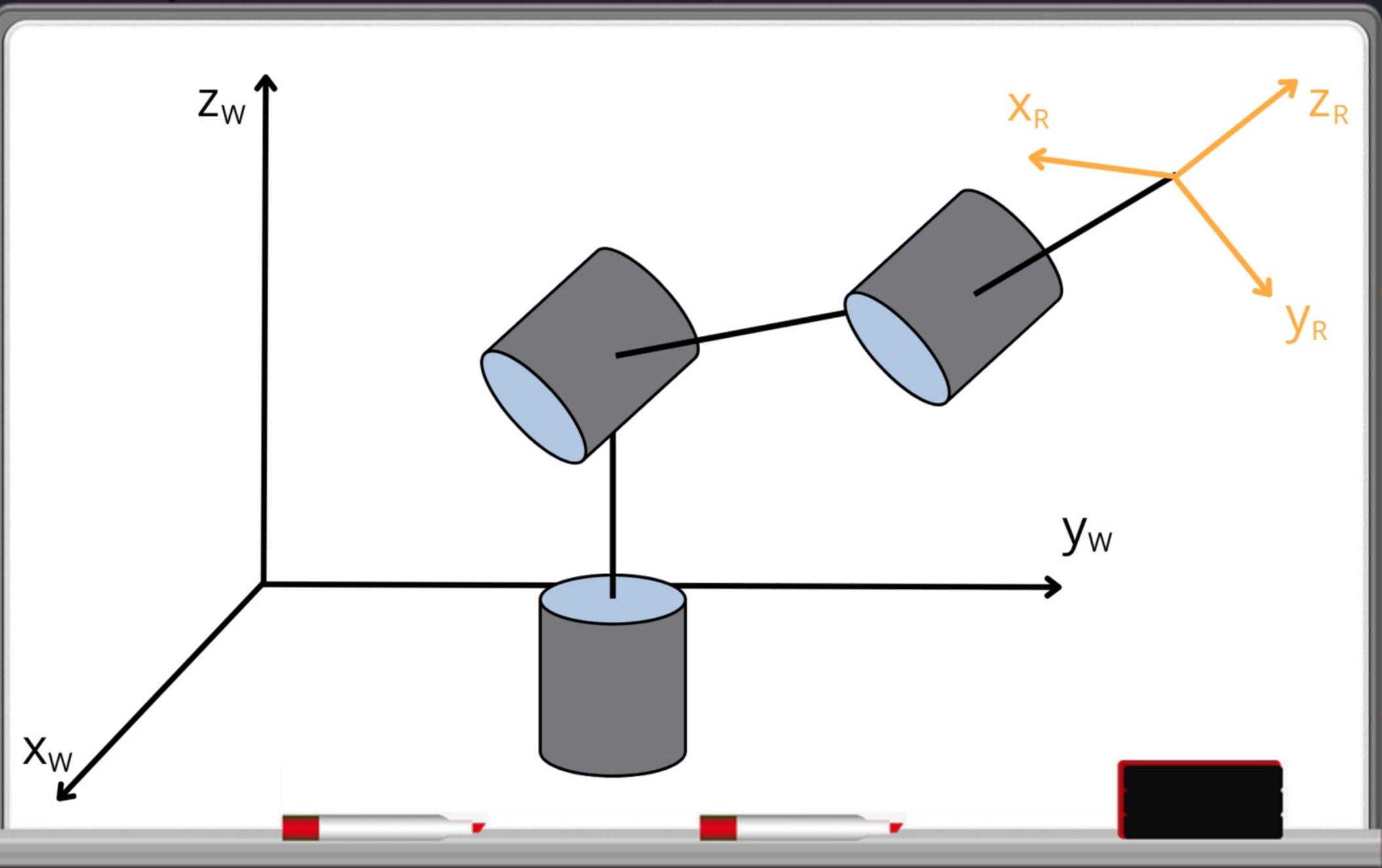
Inverse

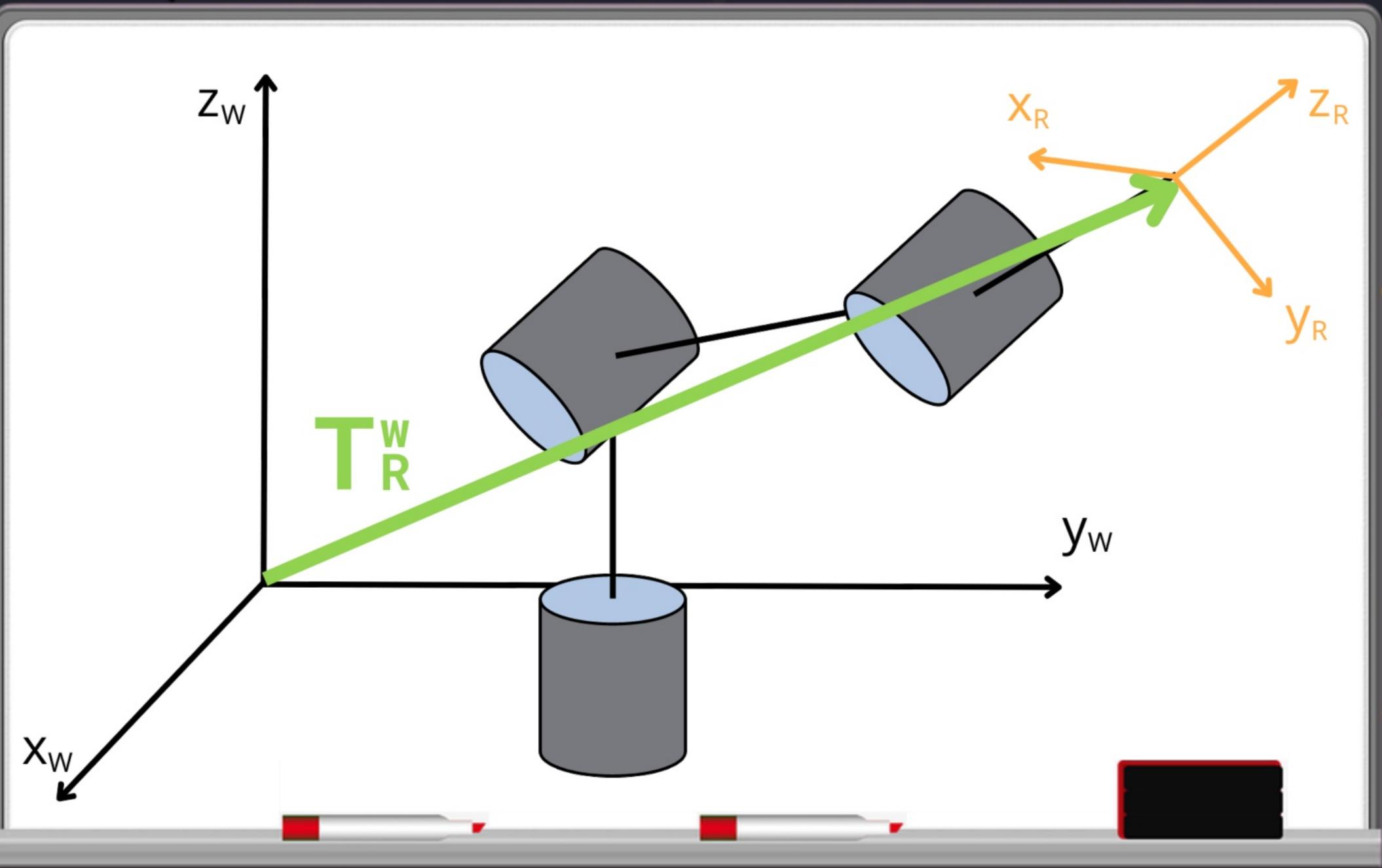


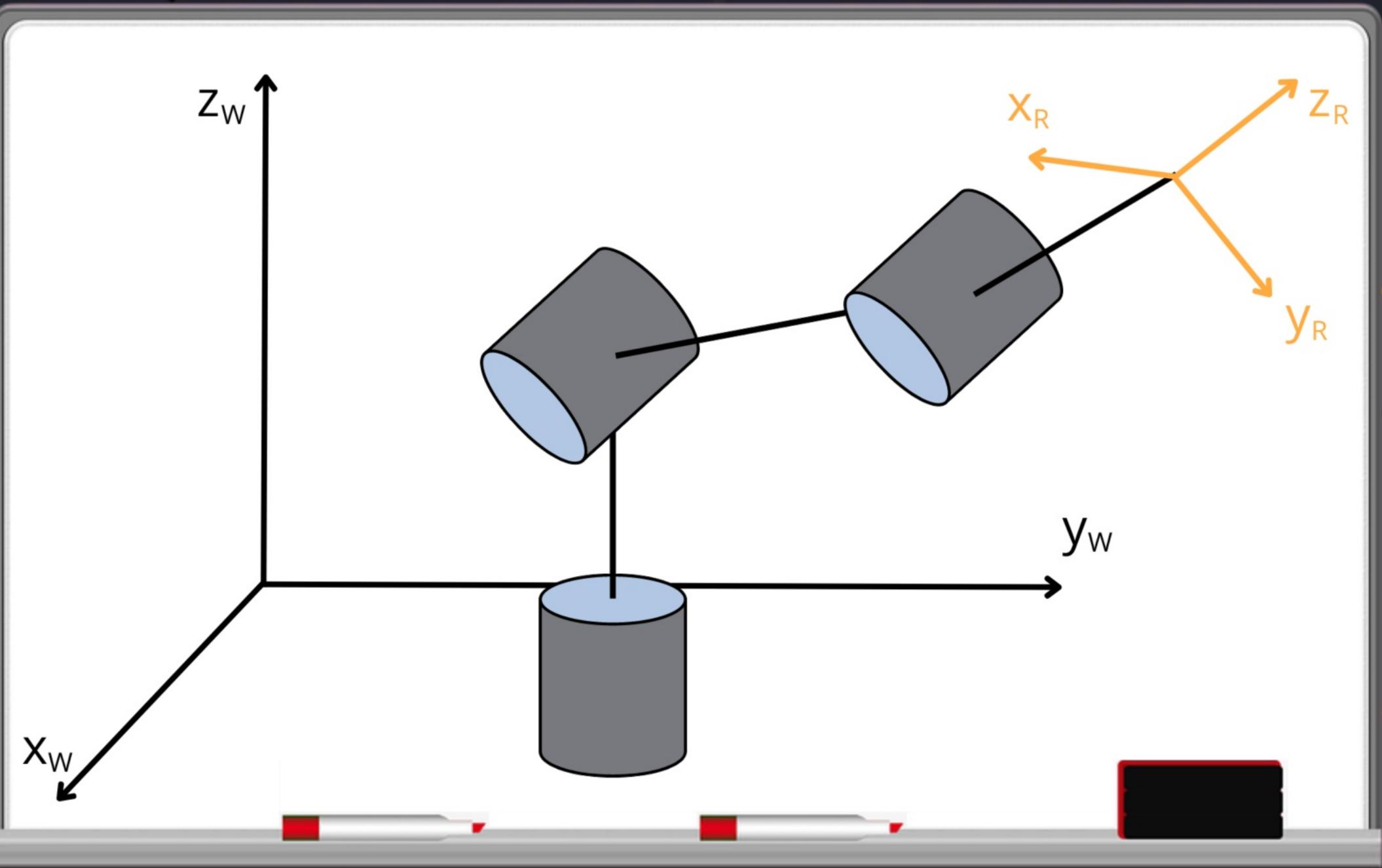


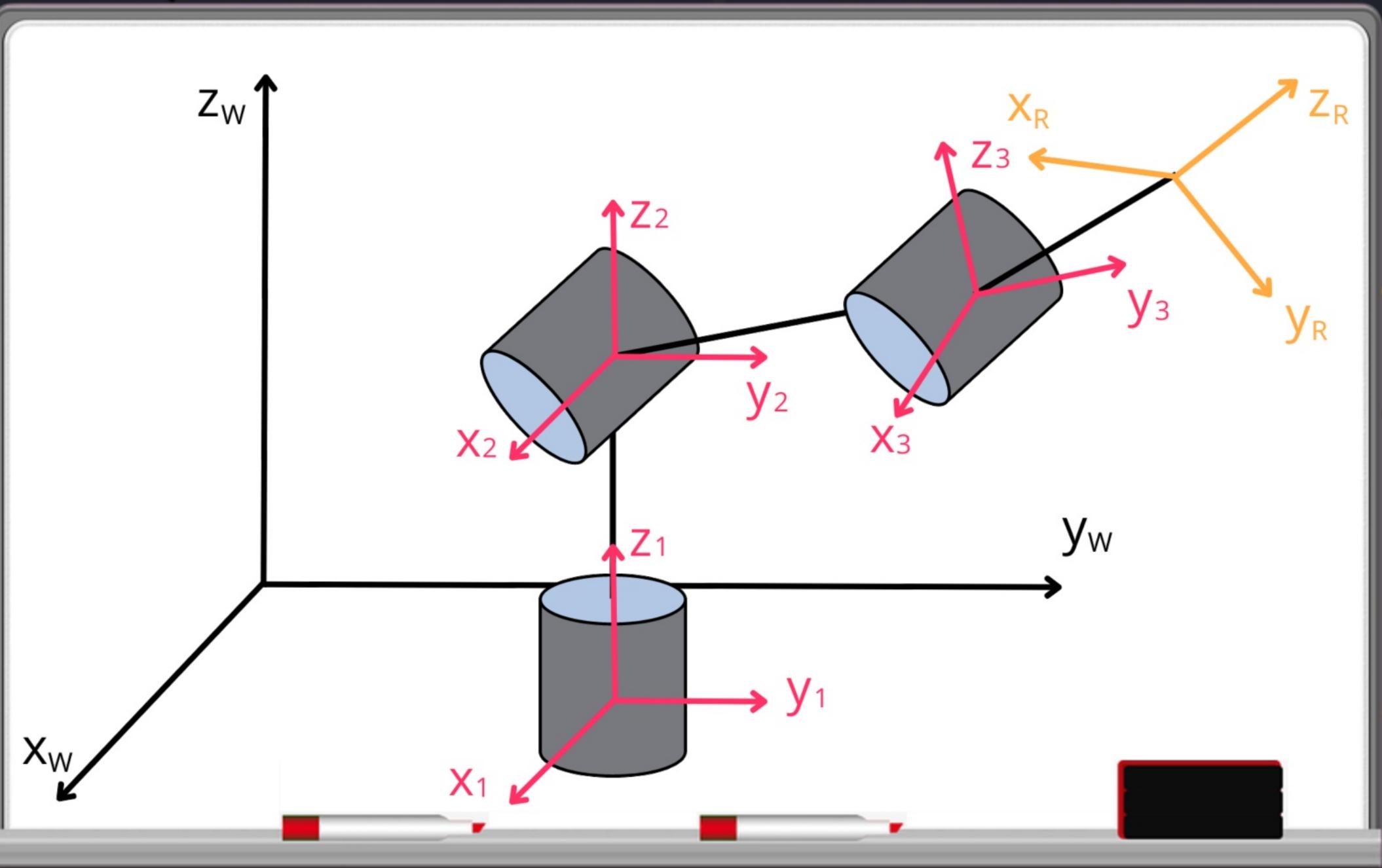


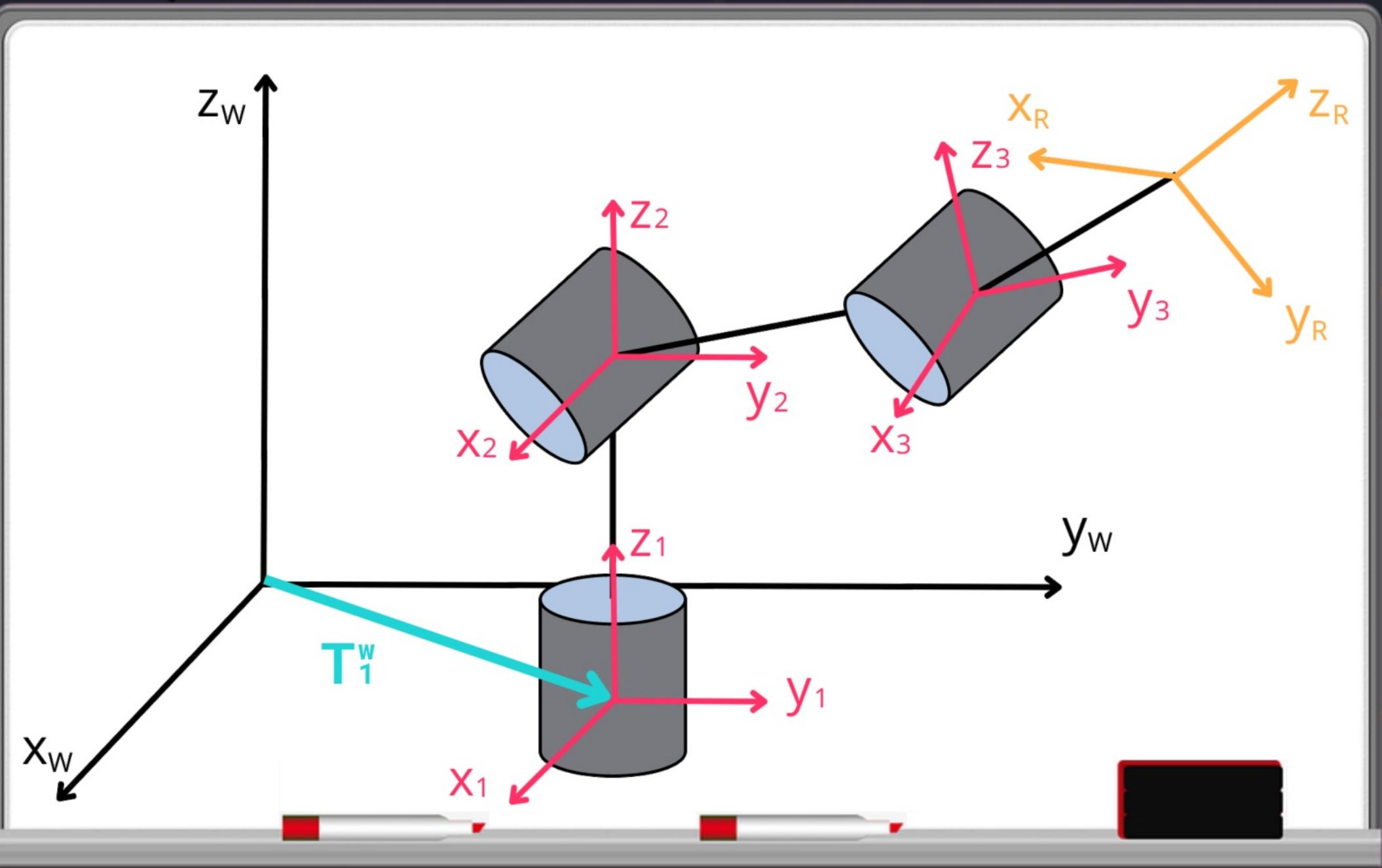


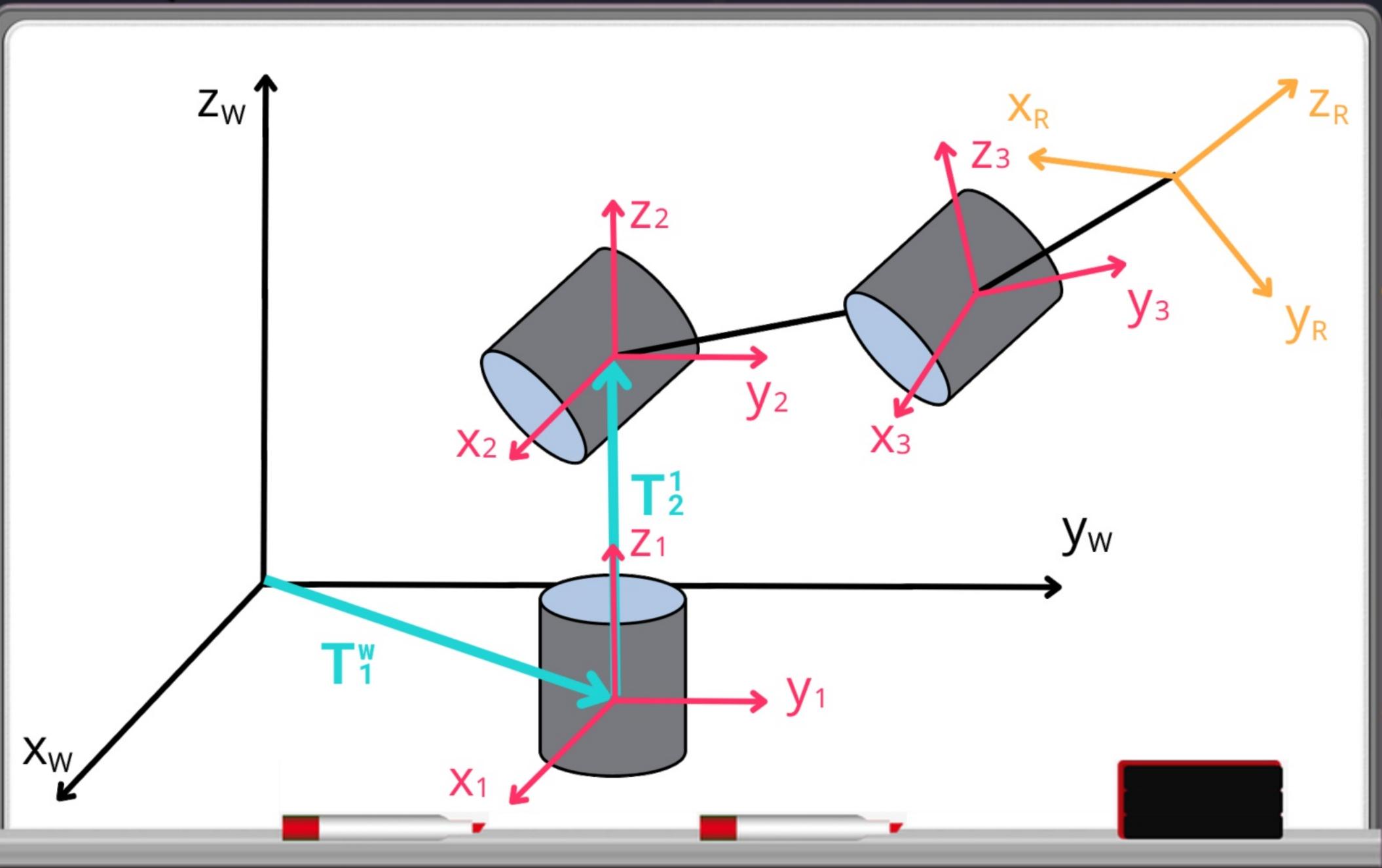


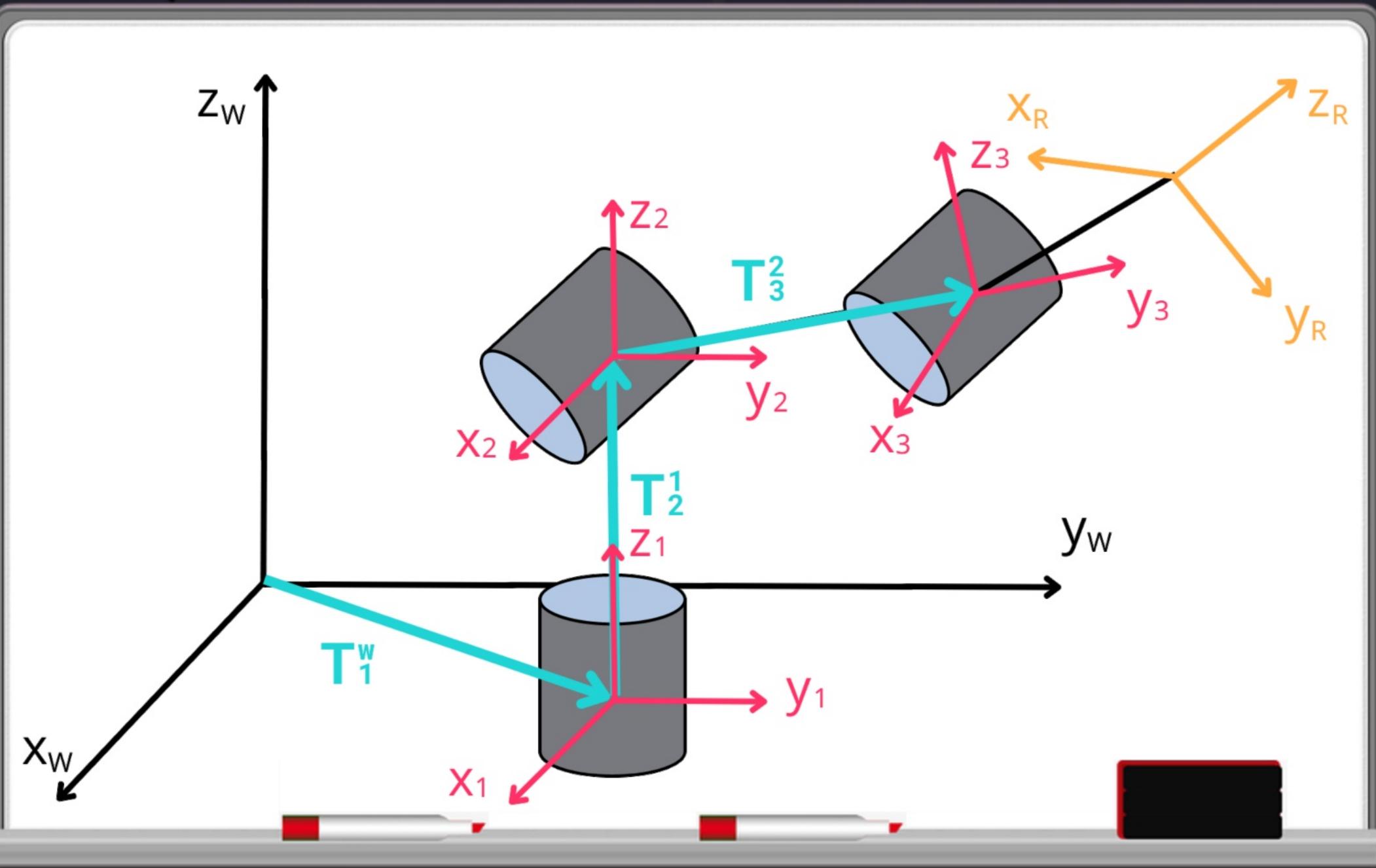


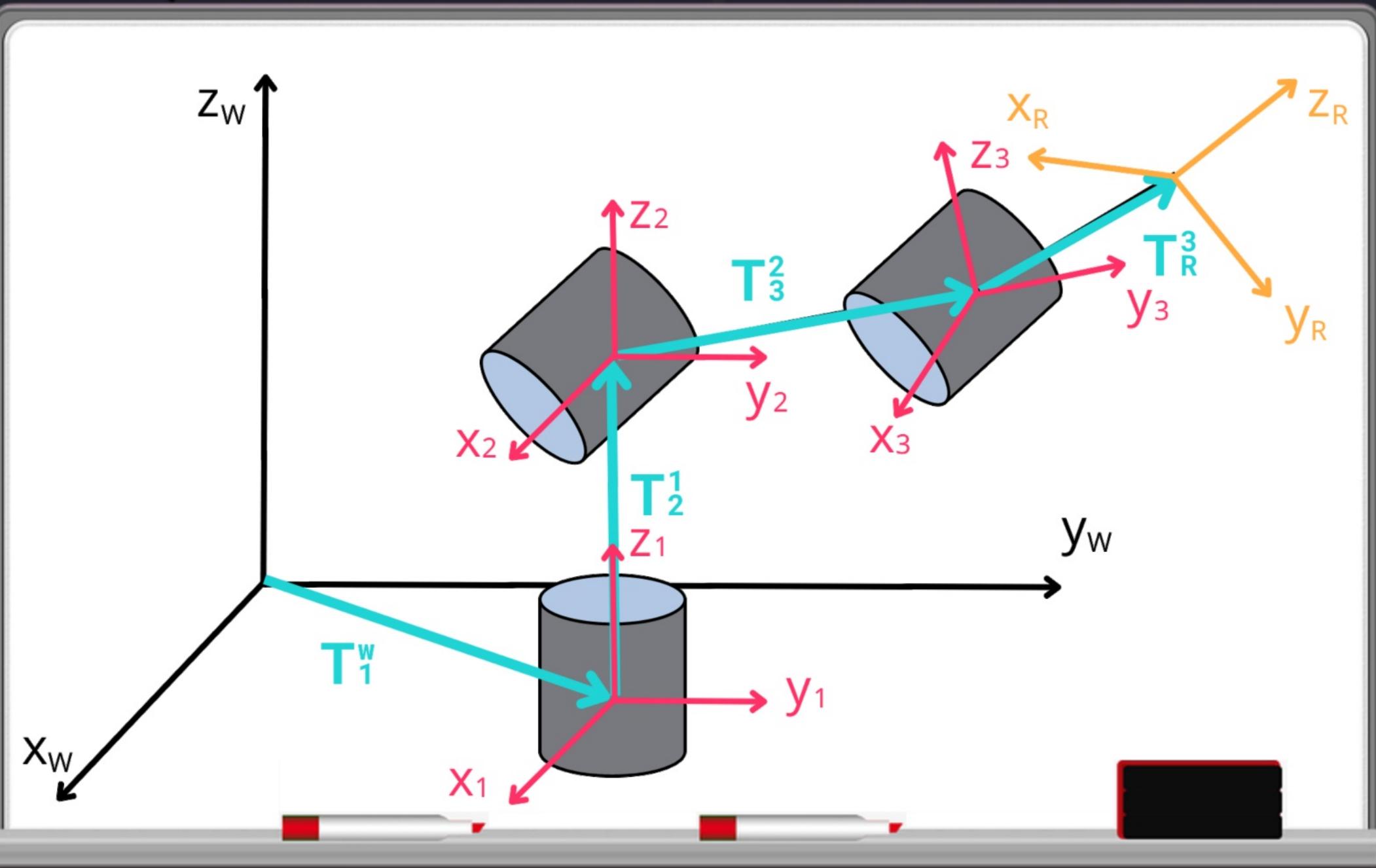


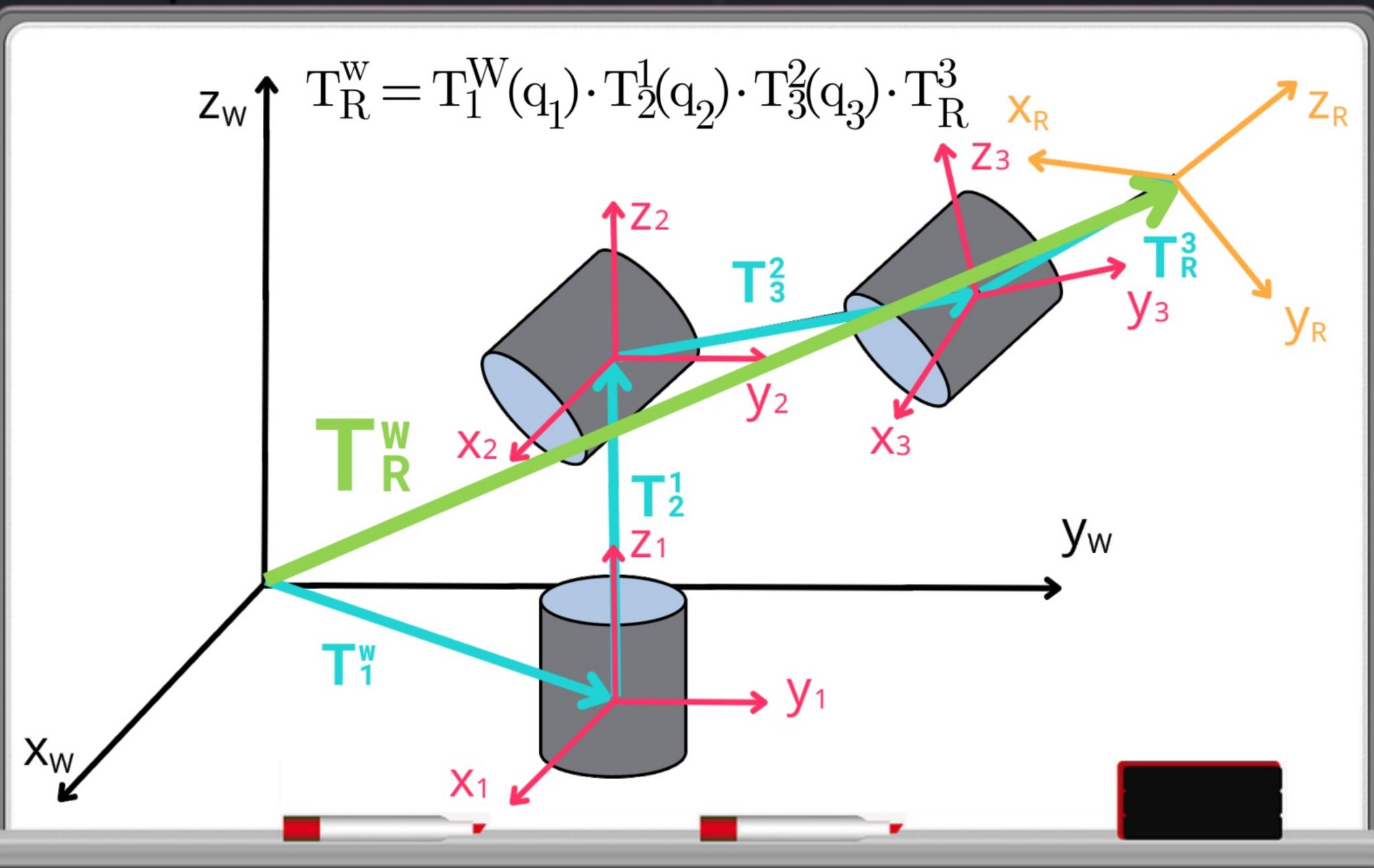


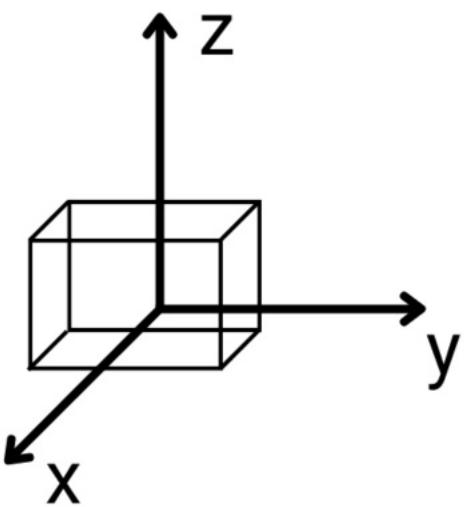


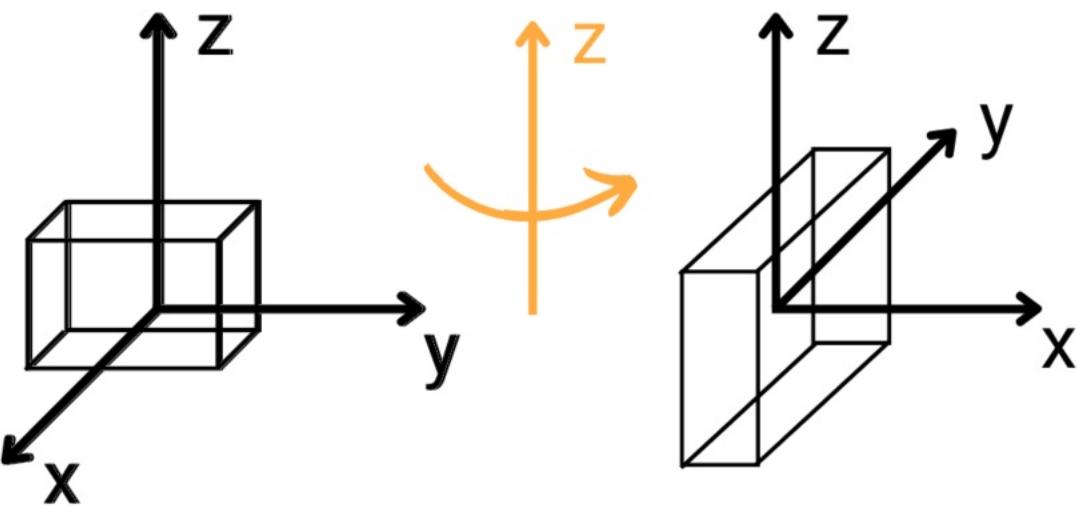


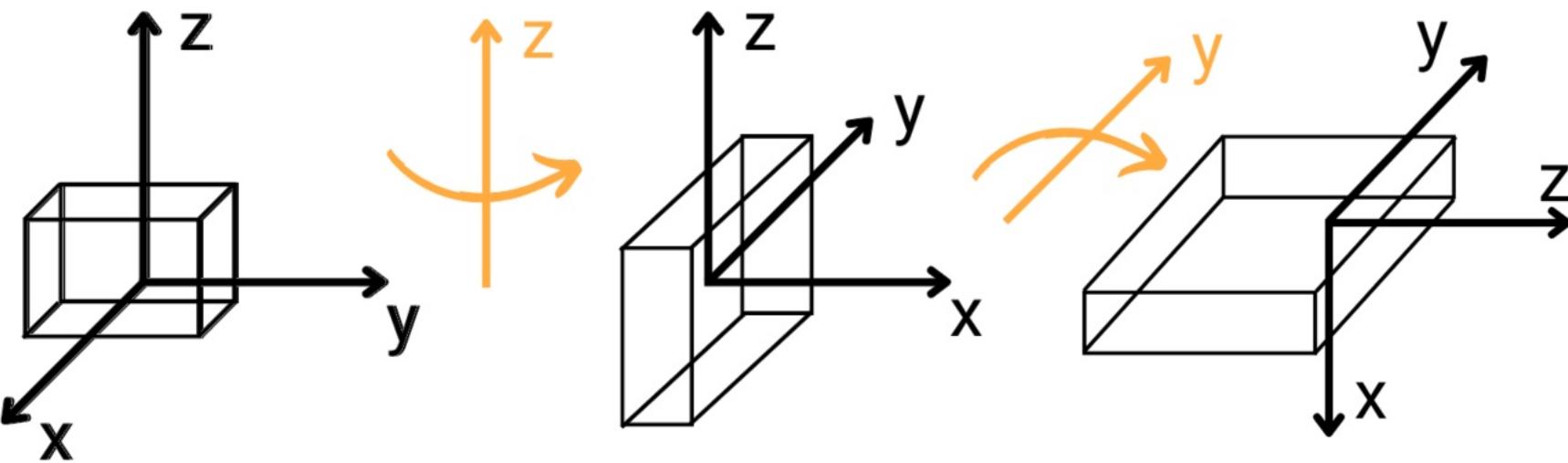


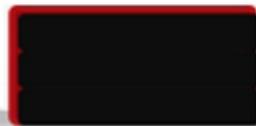
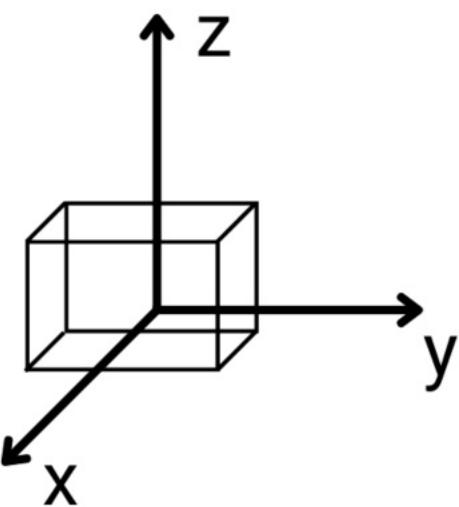
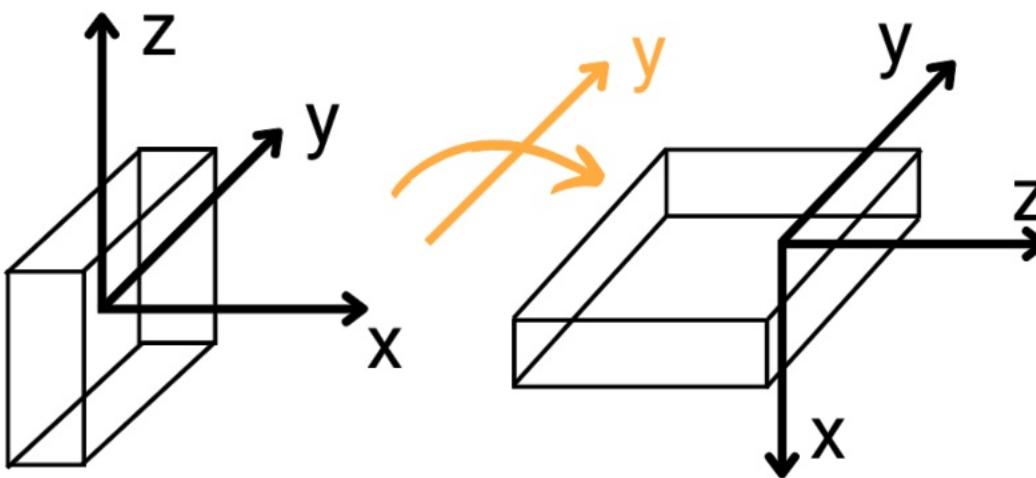
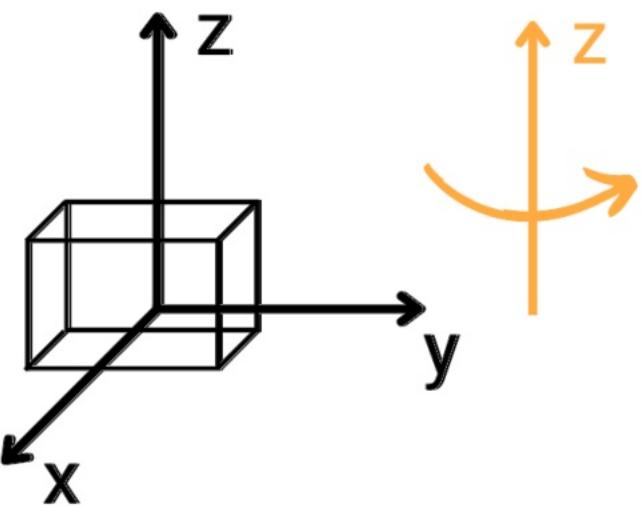


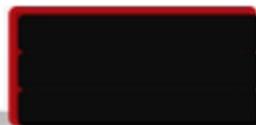
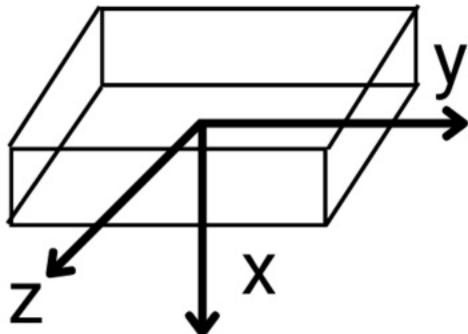
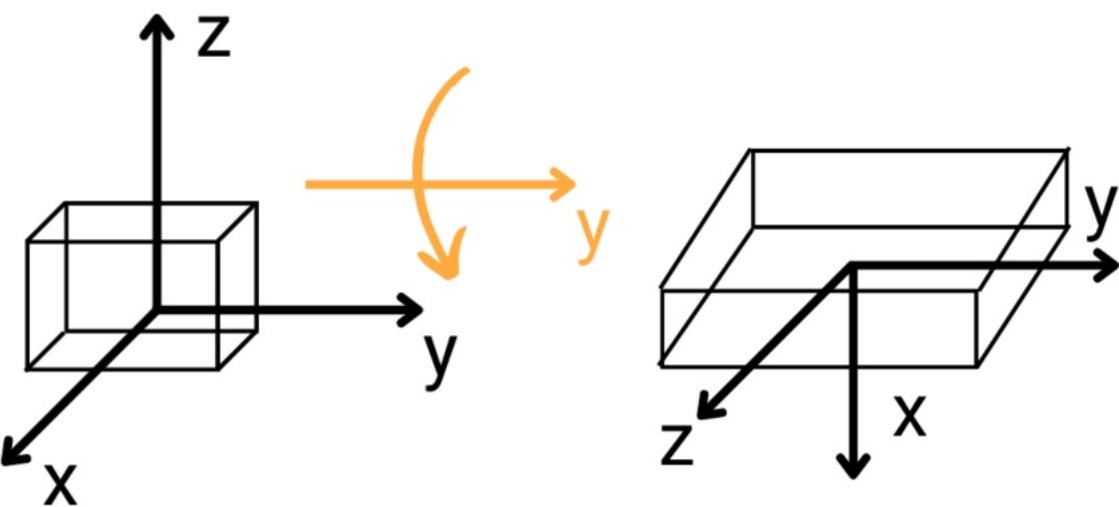
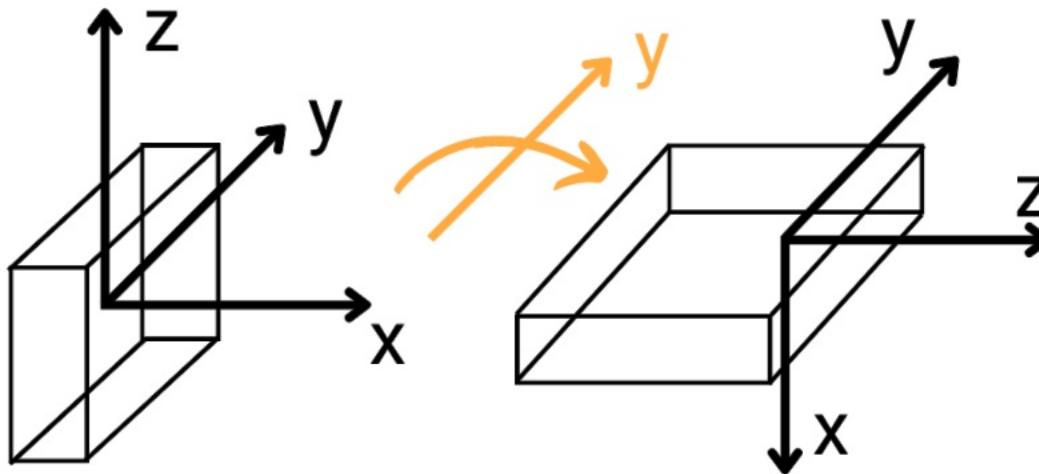
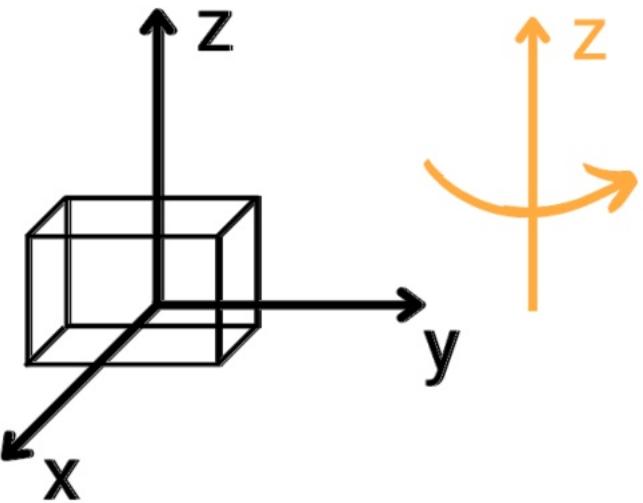


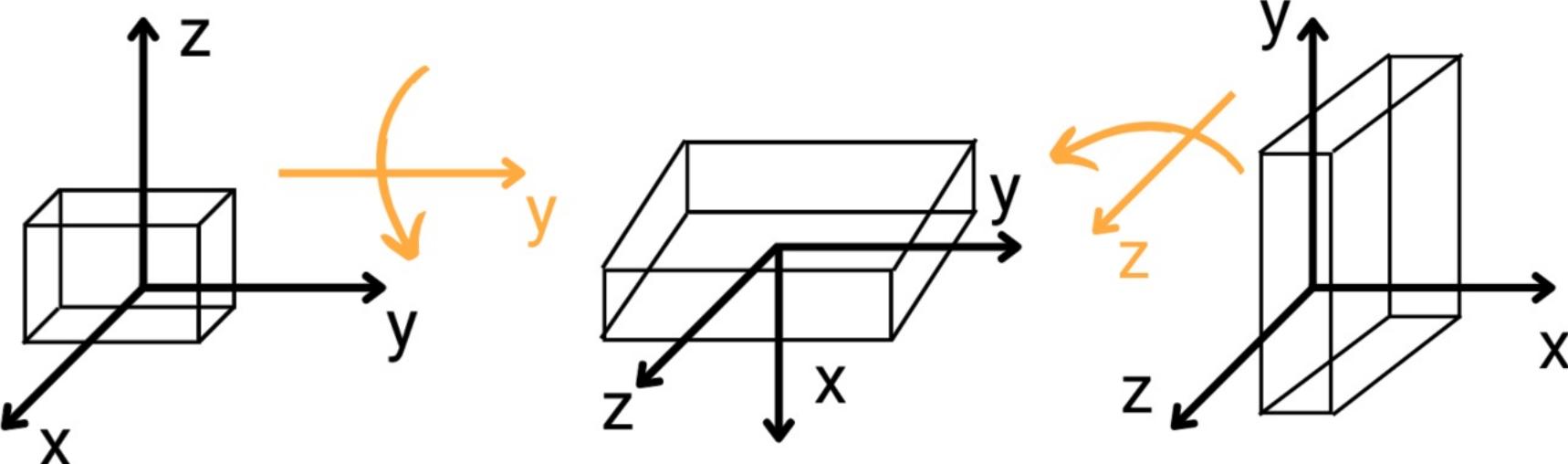
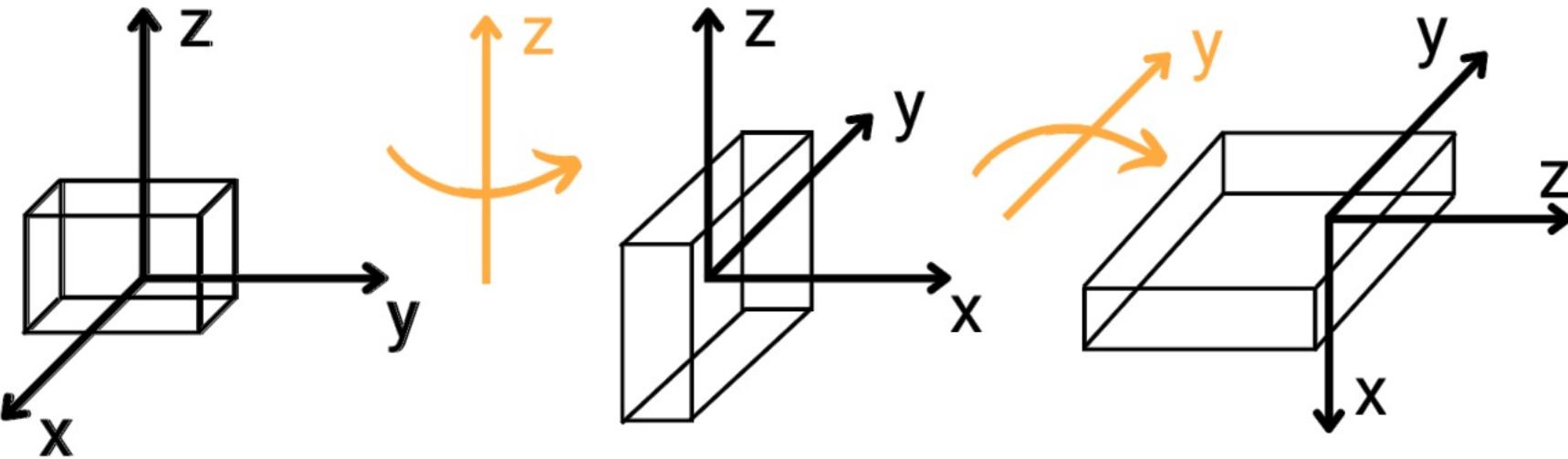


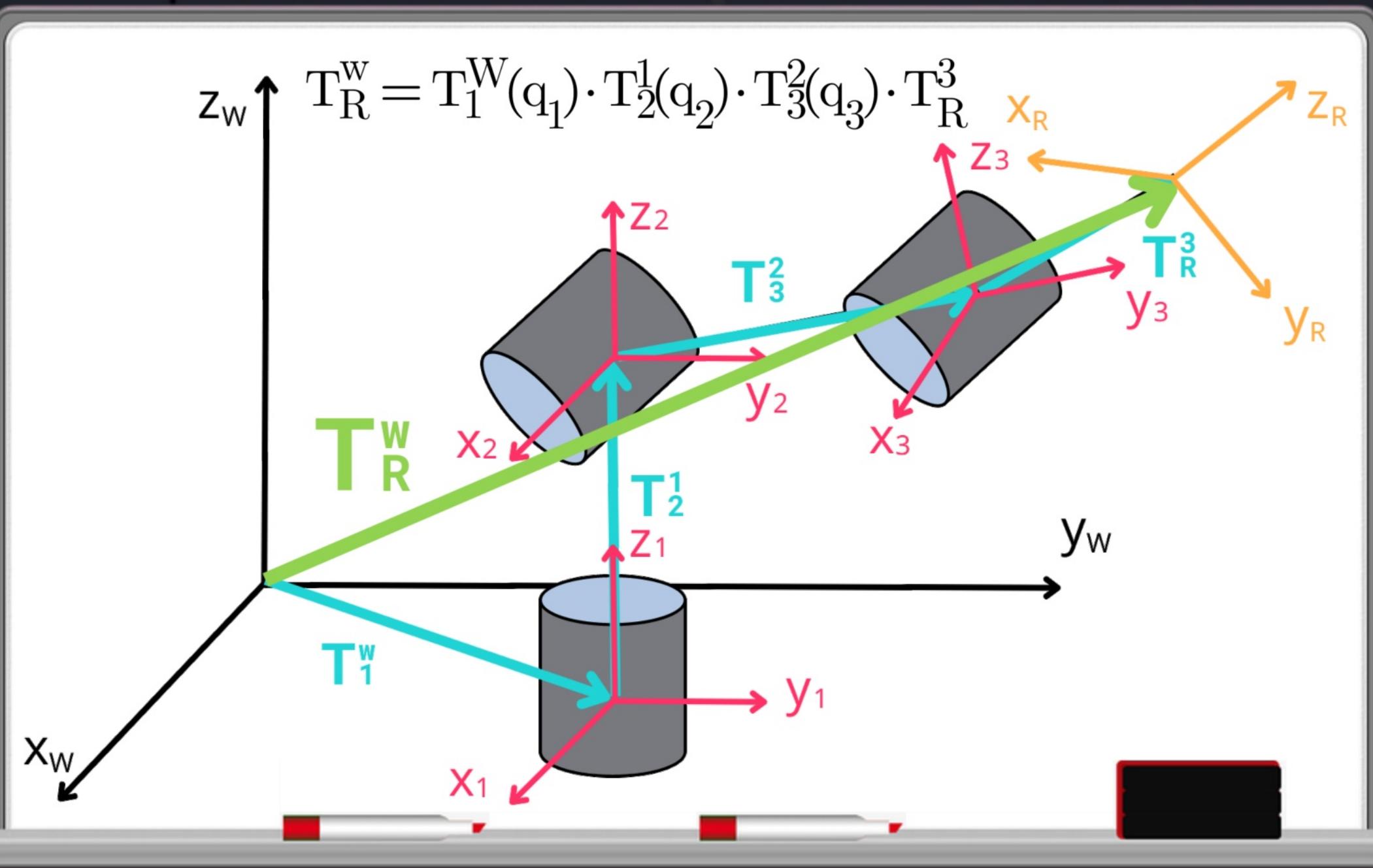




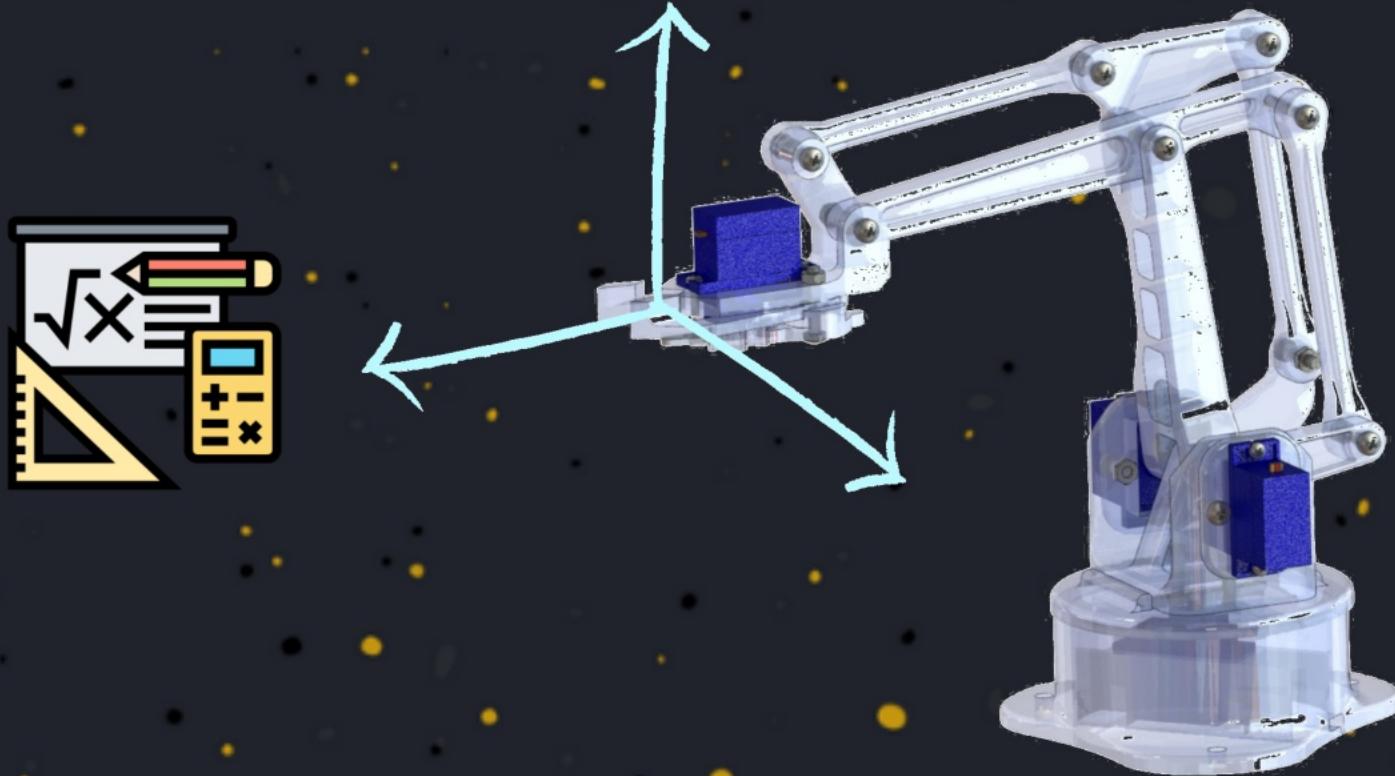
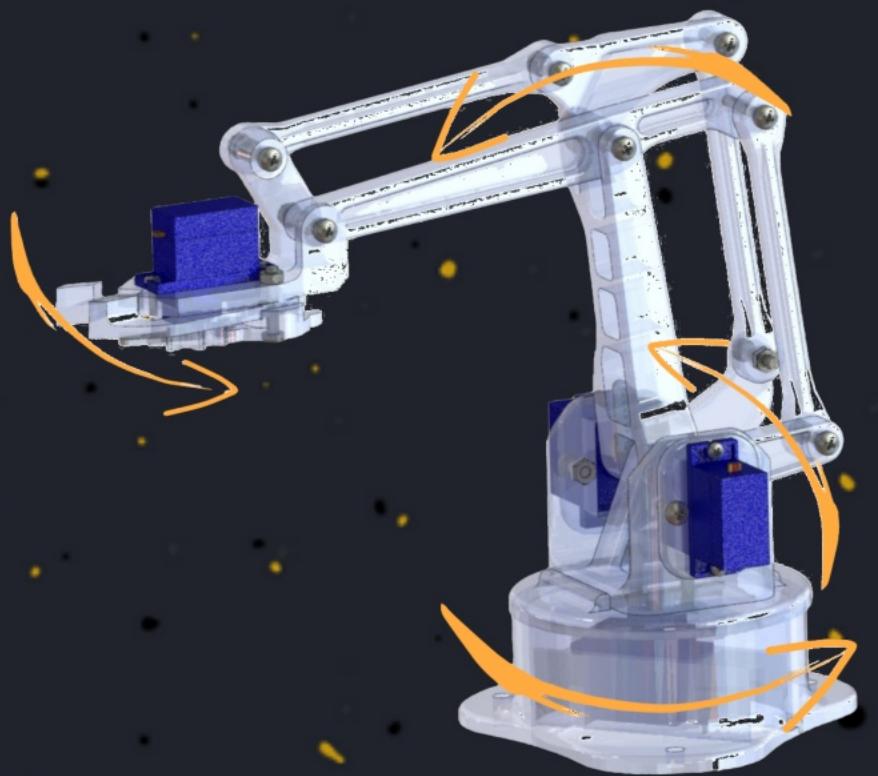








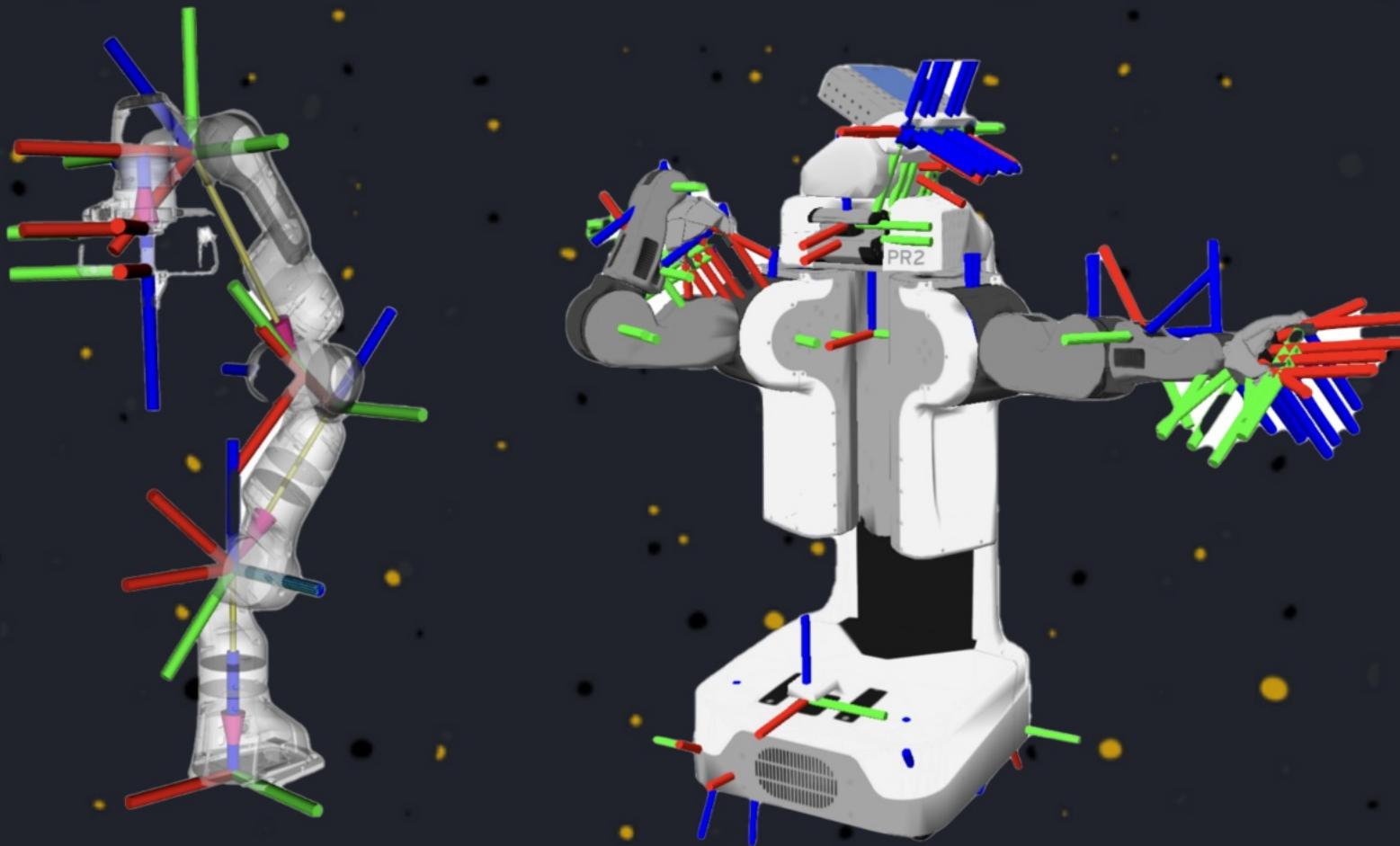
Forward



Inverse

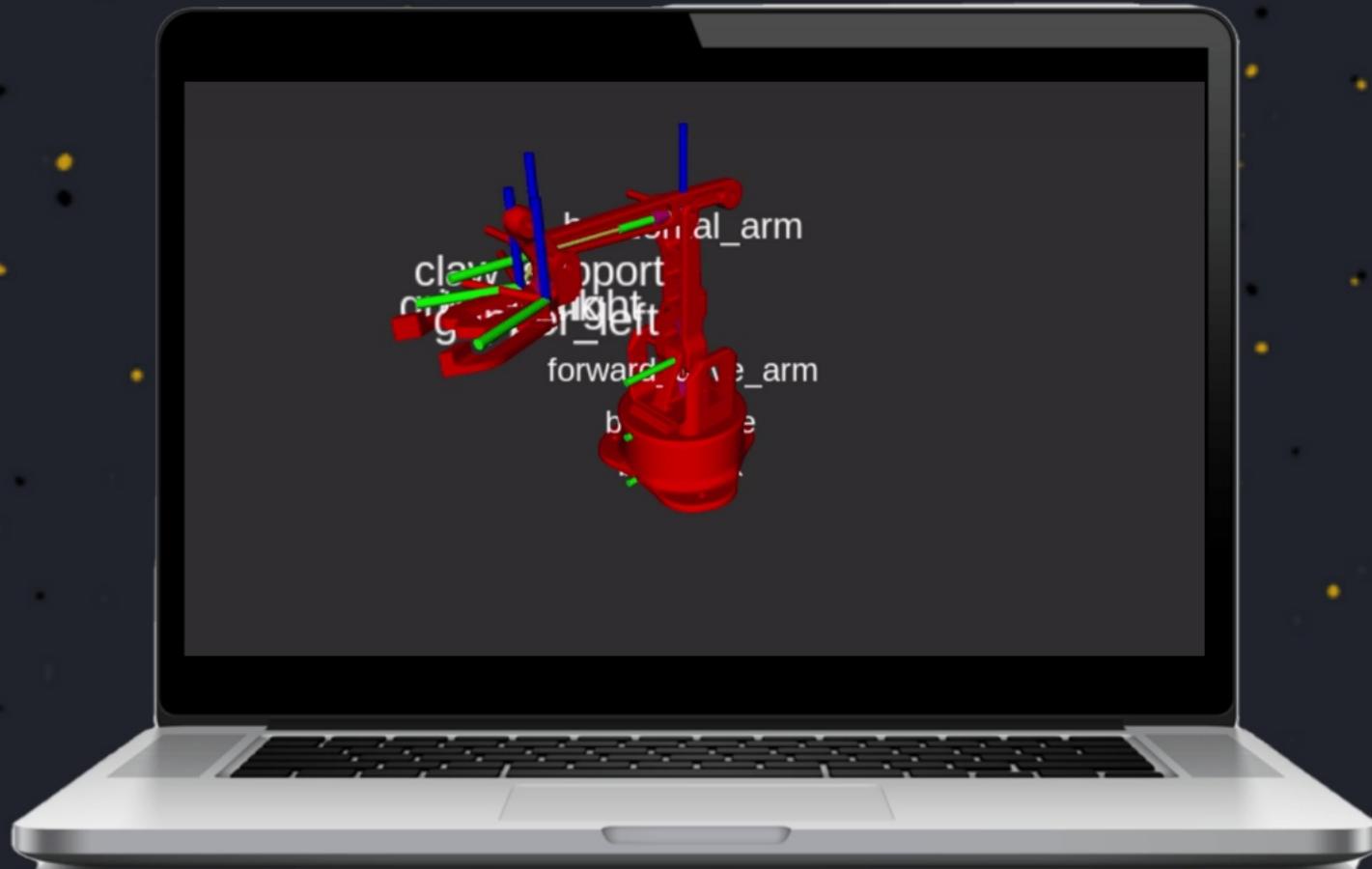


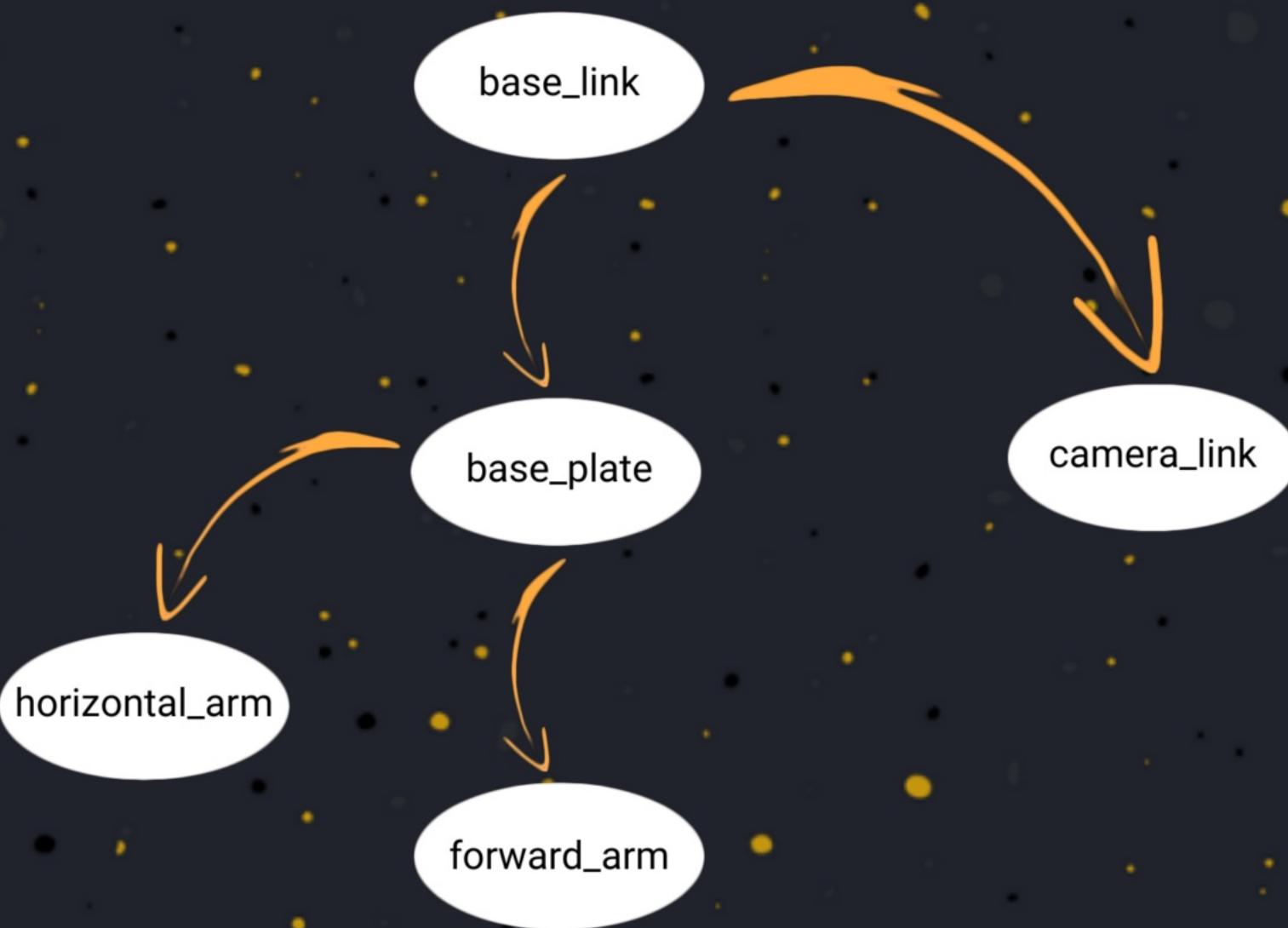
TF2 Library



TF2 Library

TF2 Library





```
<robot name="arduinobot">

  <link name="world"/>

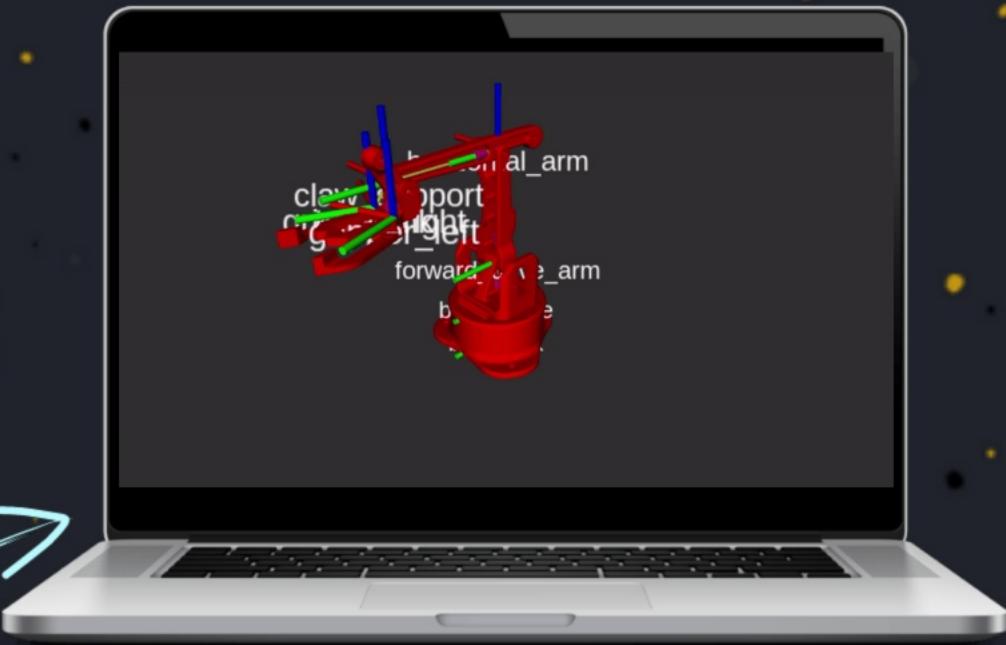
  <link name="base_link">
    <visual>
      <origin rpy="0 0 0" xyz="-0.5 -0.5 0"/>
      <geometry>
        <mesh filename="basement.STL"/>
      </geometry>
    </visual>
    <collision>
      <origin rpy="0 0 0" xyz="-0.5 -0.5 0"/>
      <geometry>
        <mesh filename="basement.STL"/>
      </geometry>
    </collision>
  </link>

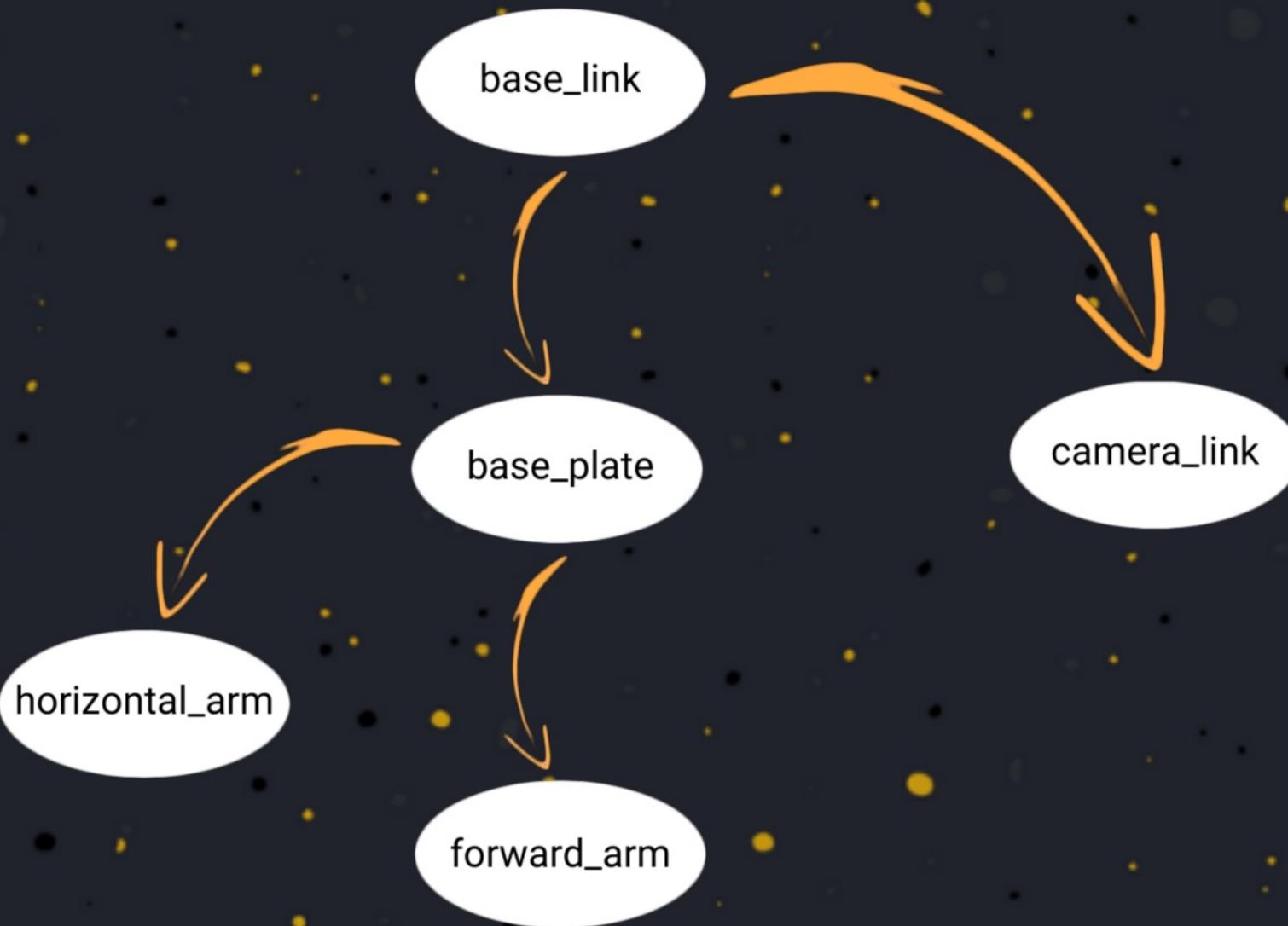
  <joint name="virtual_joint" type="fixed">
    <parent link="world"/>
    <child link="base_link"/>
    <origin xyz="0 0 0" rpy="0 0 0"/>
  </joint>

</robot>
```

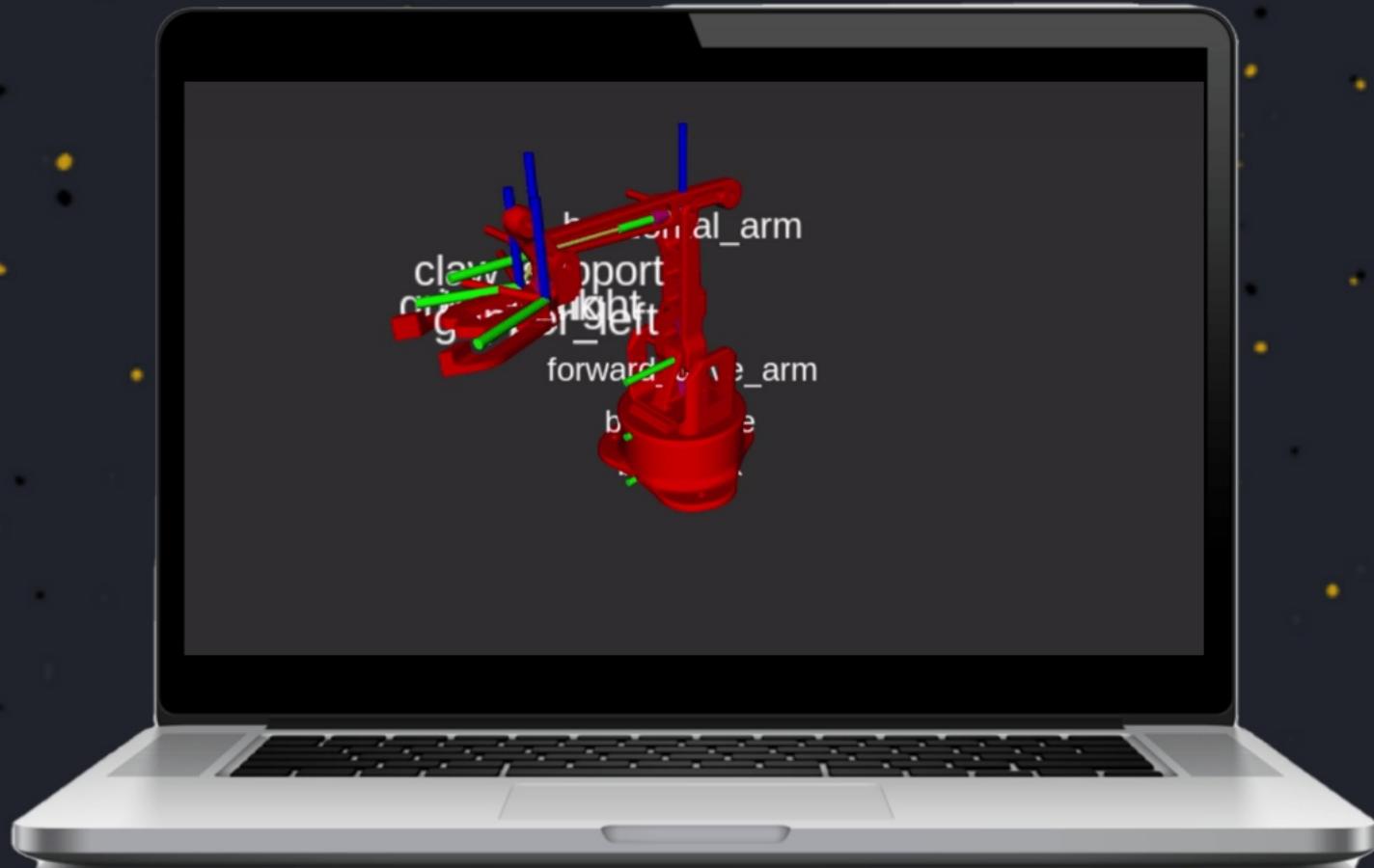
robot_state_publisher

/tf





TF2 Library



```
<robot name="arduinobot">

<link name="world"/>

<link name="base_link">
  <visual>
    <origin rpy="0 0 0" xyz="-0.5 -0.5 0"/>
    <geometry>
      <mesh filename="basement.STL"/>
    </geometry>
  </visual>
  <collision>
    <origin rpy="0 0 0" xyz="-0.5 -0.5 0"/>
    <geometry>
      <mesh filename="basement.STL"/>
    </geometry>
  </collision>
</link>

<joint name="virtual_joint" type="fixed">
  <parent link="world"/>
  <child link="base_link"/>
  <origin xyz="0 0 0" rpy="0 0 0"/>
</joint>

<joint name = "joint_1" type="revolute">
  <parent link="base_link"/>
  <child link = "base_plate"/>
  <origin xyz="0 0 0.307"/>
  <axis xyz="0 0 1"/>
</joint>

</robot>
```

robot_state_publisher



```
<robot name="arduinobot">

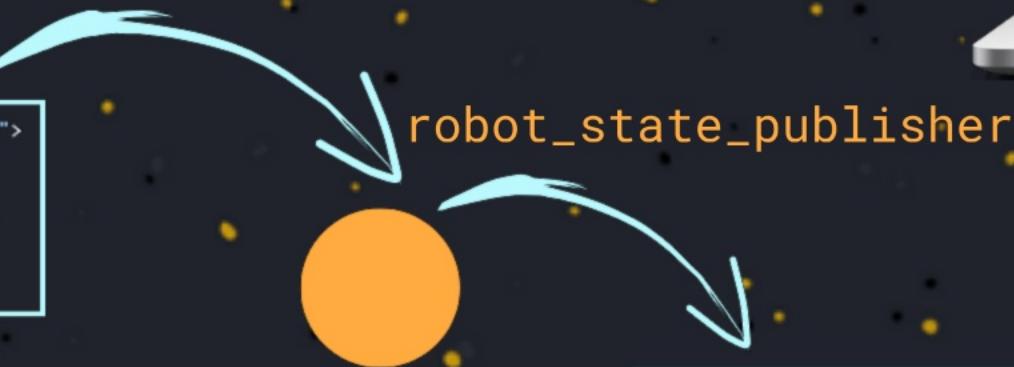
<link name="world"/>

<link name="base_link">
  <visual>
    <origin rpy="0 0 0" xyz="-0.5 -0.5 0"/>
    <geometry>
      <mesh filename="basement.STL"/>
    </geometry>
  </visual>
  <collision>
    <origin rpy="0 0 0" xyz="-0.5 -0.5 0"/>
    <geometry>
      <mesh filename="basement.STL"/>
    </geometry>
  </collision>
</link>

<joint name="virtual_joint" type="fixed">
  <parent link="world"/>
  <child link="base_link"/>
  <origin xyz="0 0 0" rpy="0 0 0"/>
</joint>

<joint name = "joint_1" type="revolute">
  <parent link="base_link"/>
  <child link = "base_plate"/>
  <origin xyz="0 0 0.307"/>
  <axis xyz="0 0 1"/>
</joint>

</robot>
```



```
<robot name="arduinobot">

<link name="world"/>

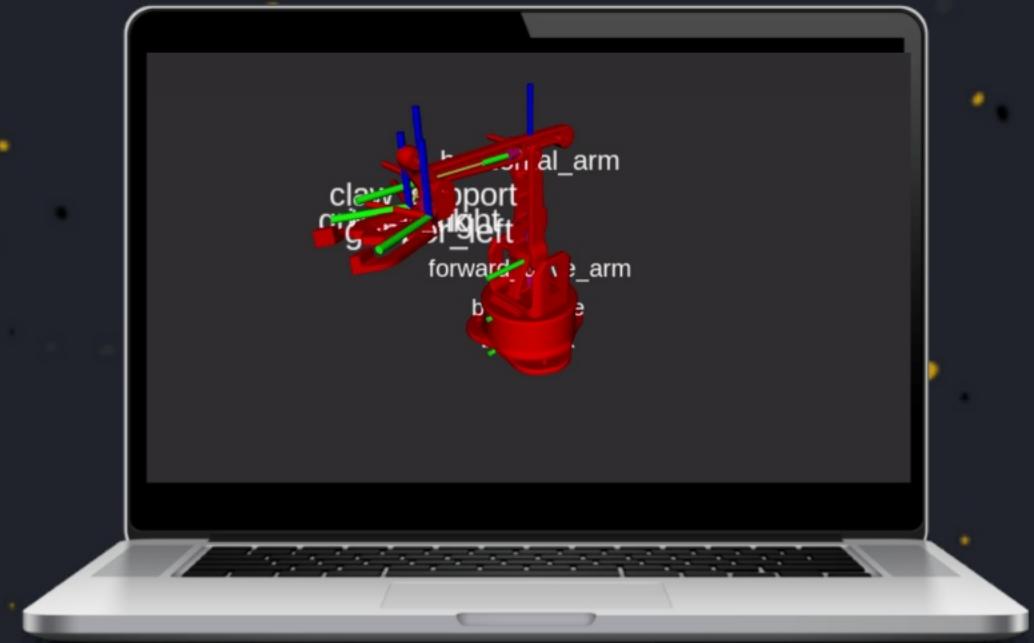
<link name="base_link">
  <visual>
    <origin rpy="0 0 0" xyz="-0.5 -0.5 0"/>
    <geometry>
      <mesh filename="basement.STL"/>
    </geometry>
  </visual>
  <collision>
    <origin rpy="0 0 0" xyz="-0.5 -0.5 0"/>
    <geometry>
      <mesh filename="basement.STL"/>
    </geometry>
  </collision>
</link>

<joint name="virtual_joint" type="fixed">
  <parent link="world"/>
  <child link="base_link"/>
  <origin xyz="0 0 0" rpy="0 0 0"/>
</joint>

<joint name = "joint_1" type="revolute">
  <parent link="base_link"/>
  <child link = "base">
  <origin xyz="0 0 0">
  <axis xyz="0 0 1"/>
  </joint>
</robot>
```

robot_state_publisher

/tf_static



```
<robot name="arduinobot">

<link name="world"/>

<link name="base_link">
  <visual>
    <origin rpy="0 0 0" xyz="-0.5 -0.5 0"/>
    <geometry>
      <mesh filename="basement.STL"/>
    </geometry>
  </visual>
  <collision>
    <origin rpy="0 0 0" xyz="-0.5 -0.5 0"/>
    <geometry>
      <mesh filename="basement.STL"/>
    </geometry>
  </collision>
</link>

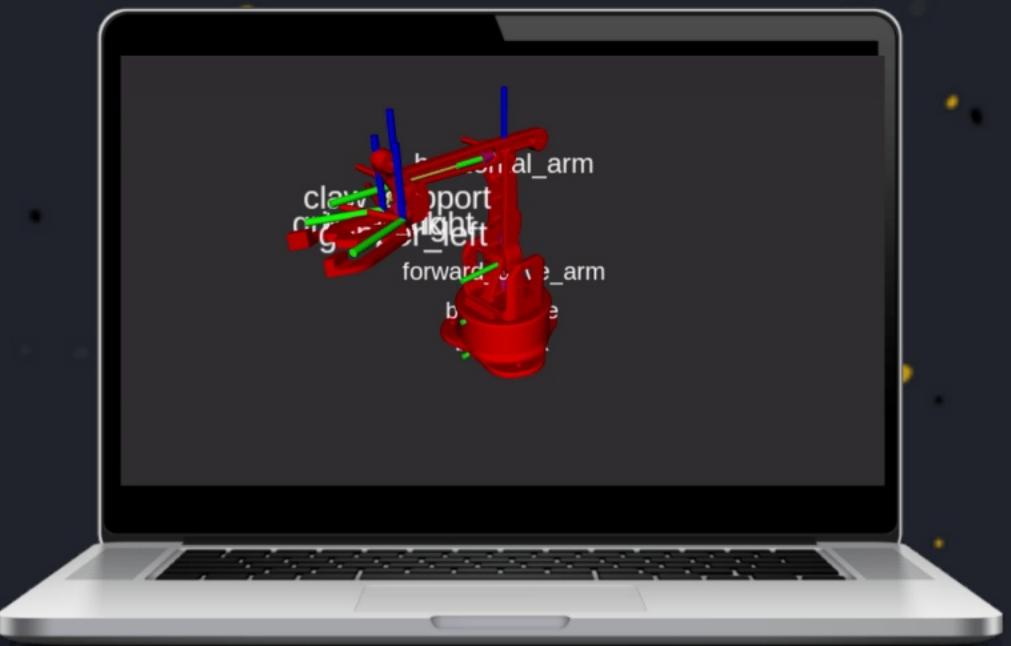
<joint name="virtual_joint" type="fixed">
  <parent link="world"/>
  <child link="base_link"/>
  <origin xyz="0 0 0" rpy="0 0 0"/>
</joint>

<joint name = "joint_1" type="revolute">
  <parent link="base_link"/>
  <child link = "base">
  <origin xyz="0 0 0" rpy="0 0 0"/>
  <axis xyz="0 0 1"/>
</joint>
</robot>
```

Robot State Publisher

The diagram illustrates the data flow from the URDF XML code to a ROS node. A large orange circle labeled "robot_state_publisher" receives input from two boxes: one containing the XML code for the "virtual_joint" and another for the "joint_1". Arrows point from these boxes to the central publisher node.

The diagram shows a configuration interface with four sliders labeled "joint_1" through "joint_4", each with a value of 0.00. A "Randomize" button is also present. Arrows point from this interface to the "joint_1" section of the URDF code and to the "tf" topic.

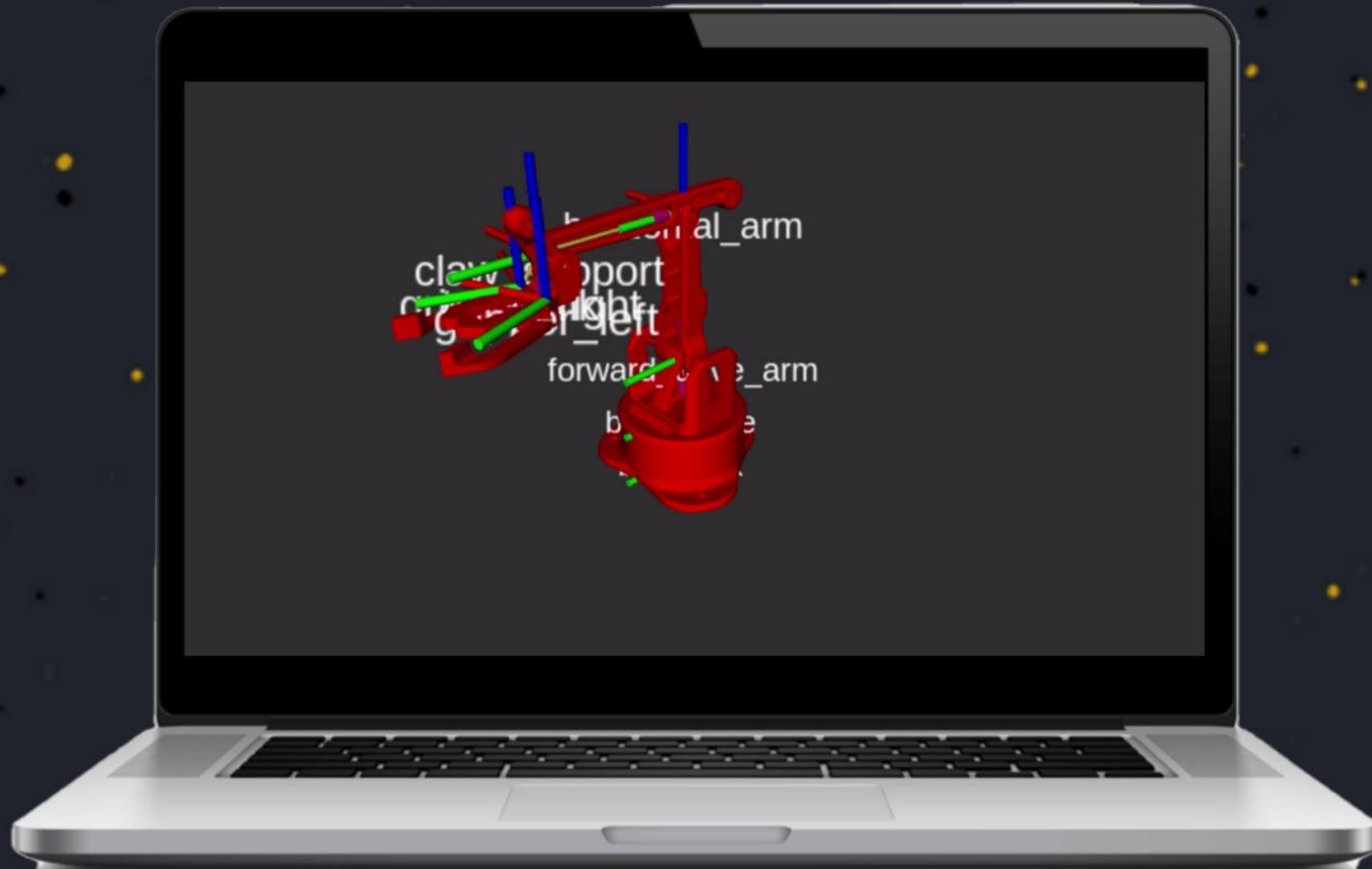


robot_state_publisher

/tf_static

/tf

TF2 Library



$f(\mathbf{x})$

$f(\mathbf{x}) \quad f(\mathbf{x})$

$f(\mathbf{x}) \quad f(\mathbf{x})$

face_recognition

$f(\mathbf{x})$ $f(\mathbf{x})$ $f(\mathbf{x})$ $f(\mathbf{x})$ $f(\mathbf{x})$ face_recognition

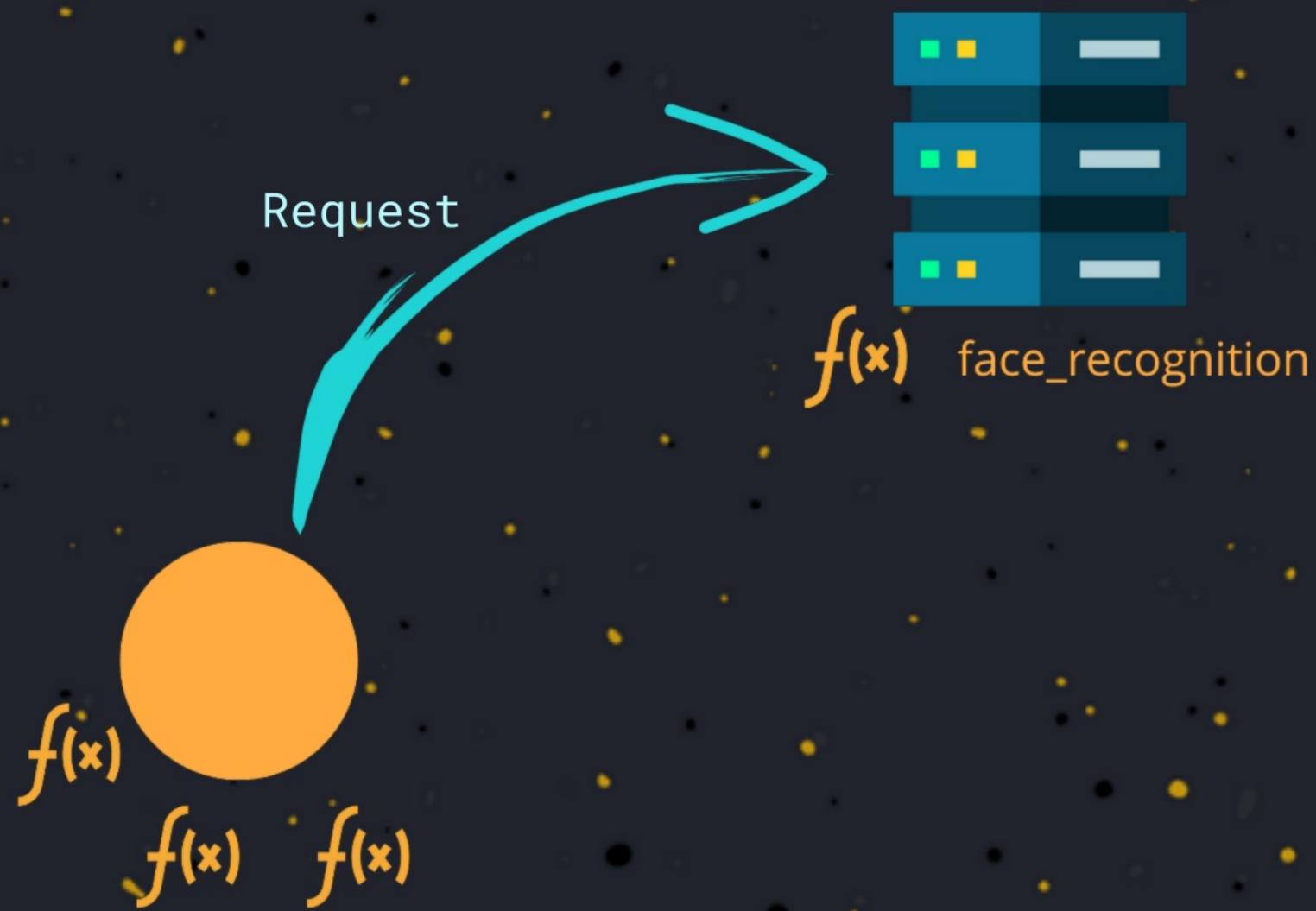
$f(\mathbf{x})$ $f(\mathbf{x})$ $f(\mathbf{x})$ $f(\mathbf{x})$ $f(\mathbf{x})$ face_recognition

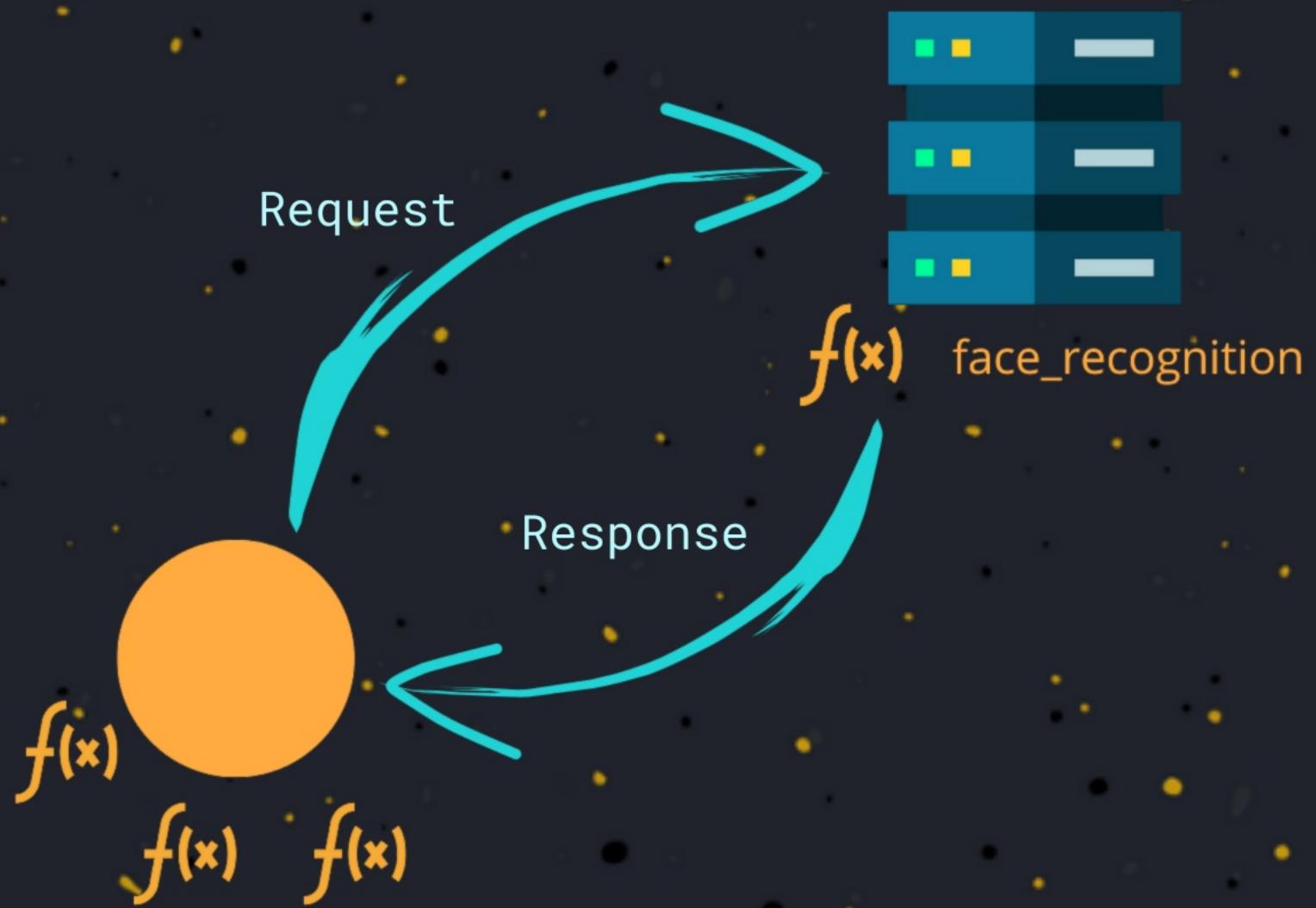


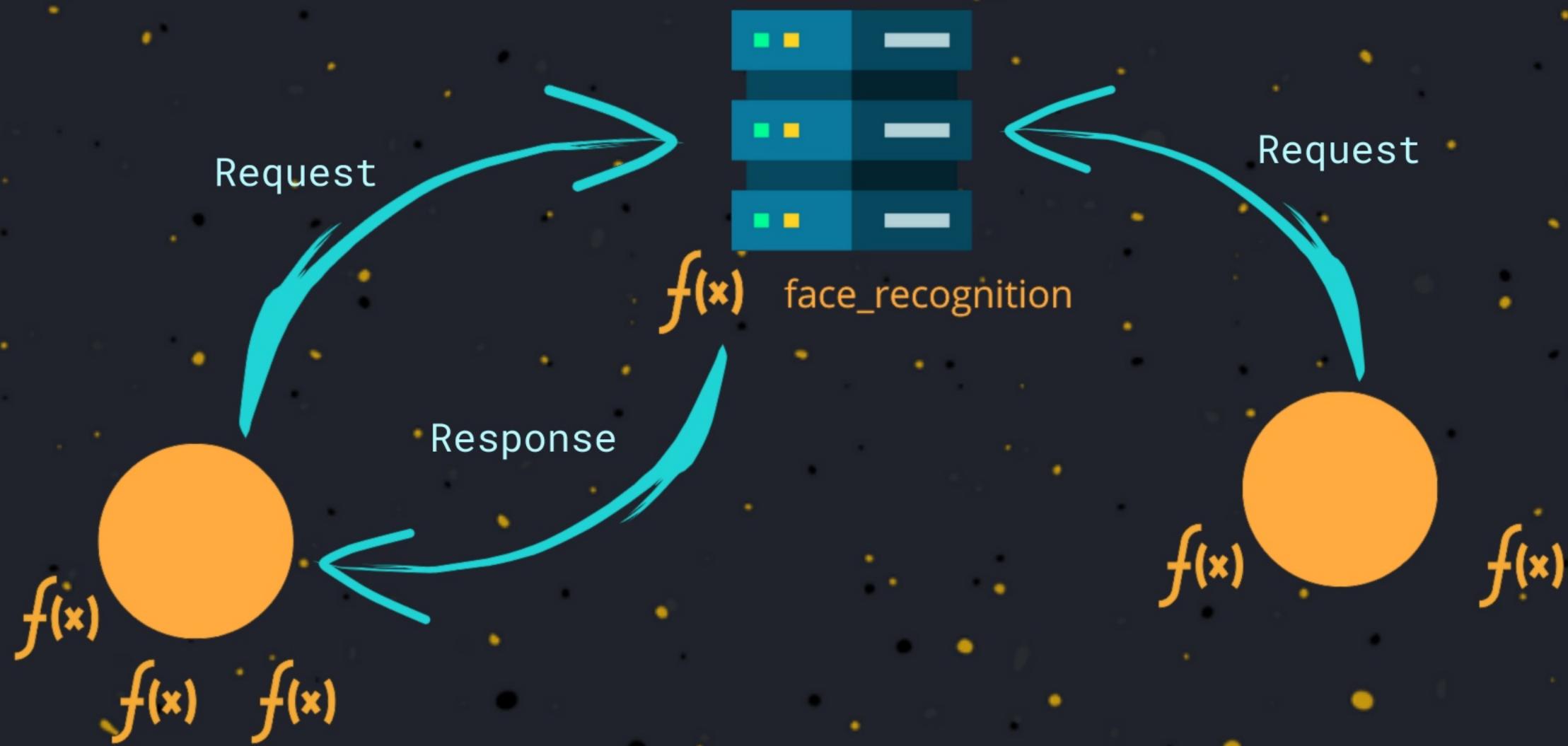


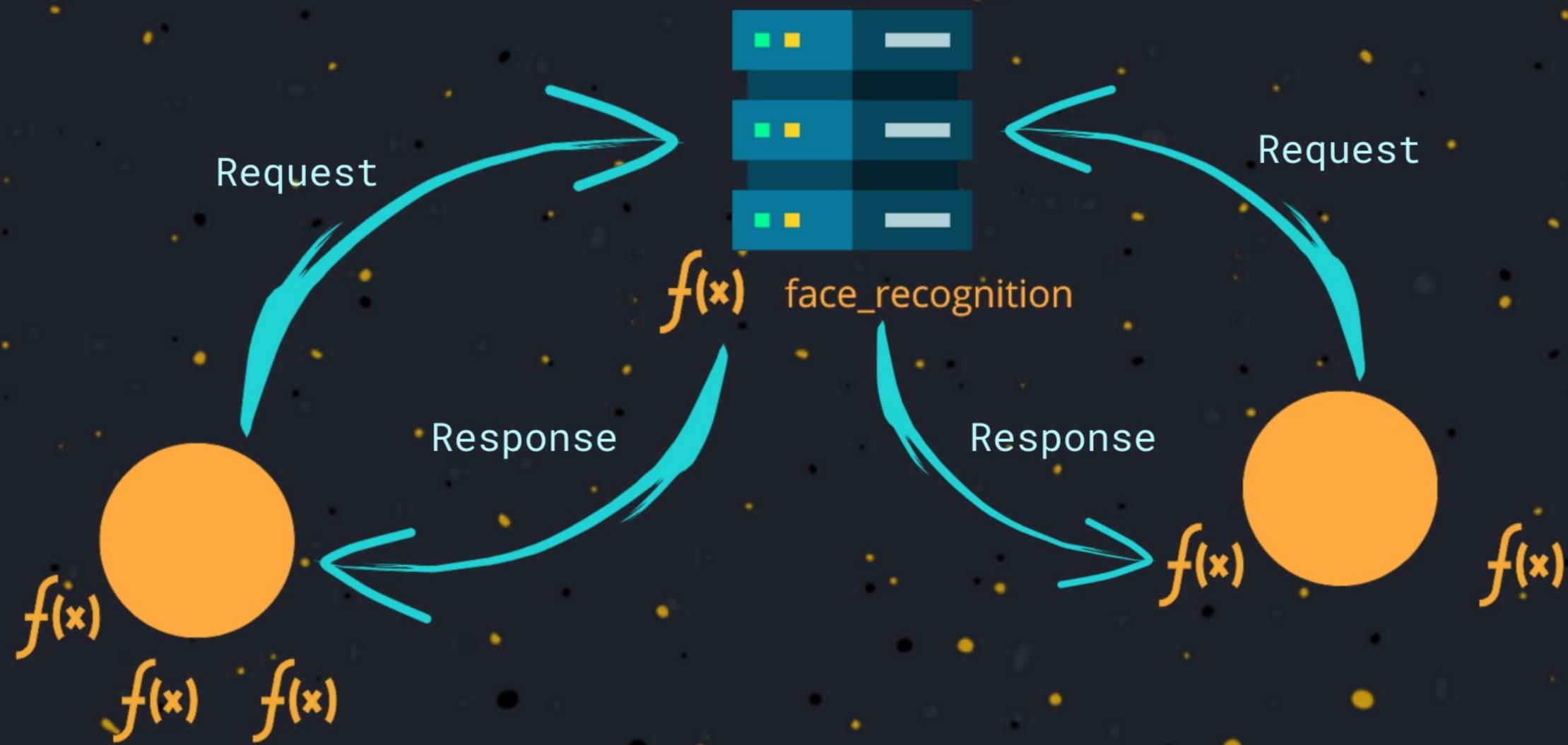
$f(\mathbf{x})$
 $f(\mathbf{x})$ $f(\mathbf{x})$

$f(\mathbf{x})$ $f(\mathbf{x})$









Service

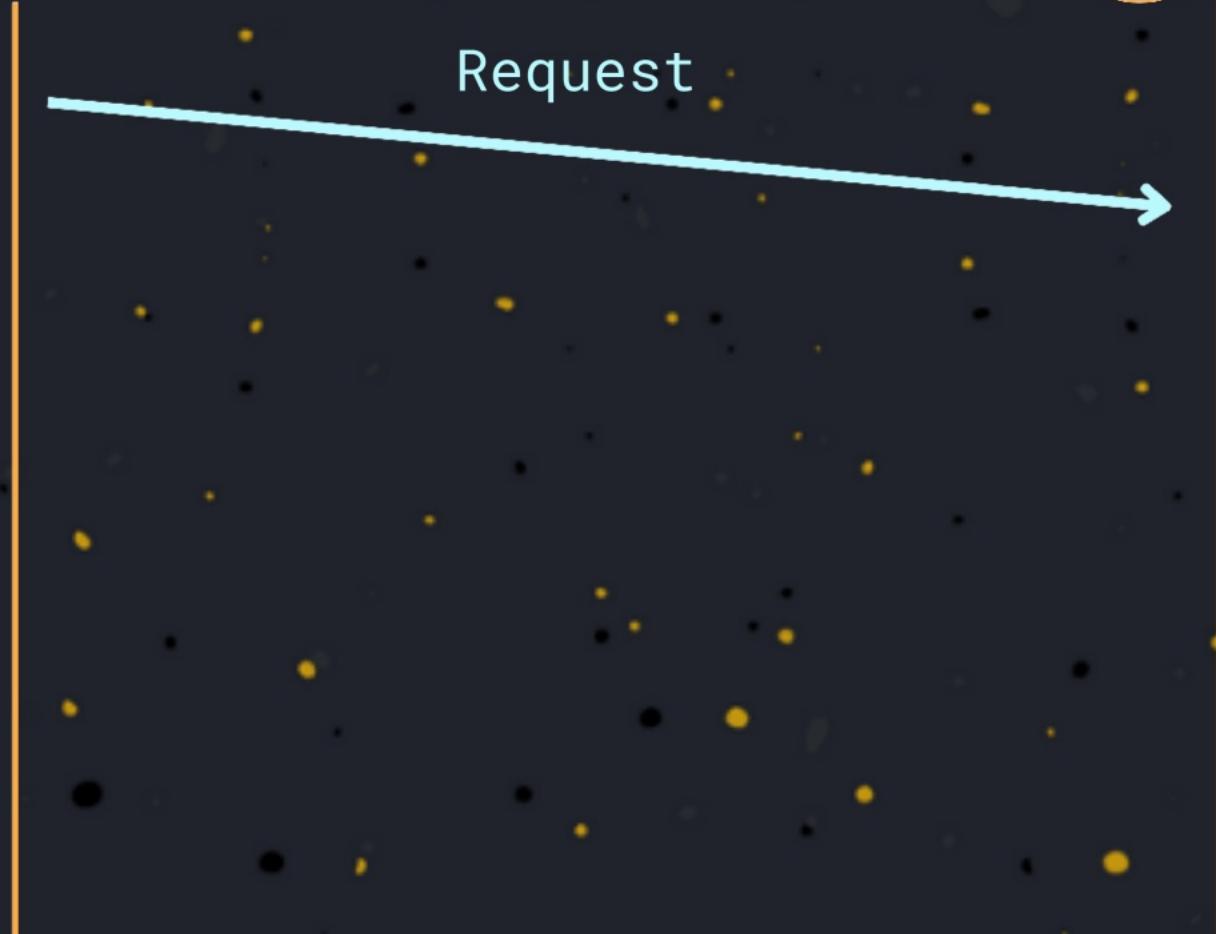


Client



Server

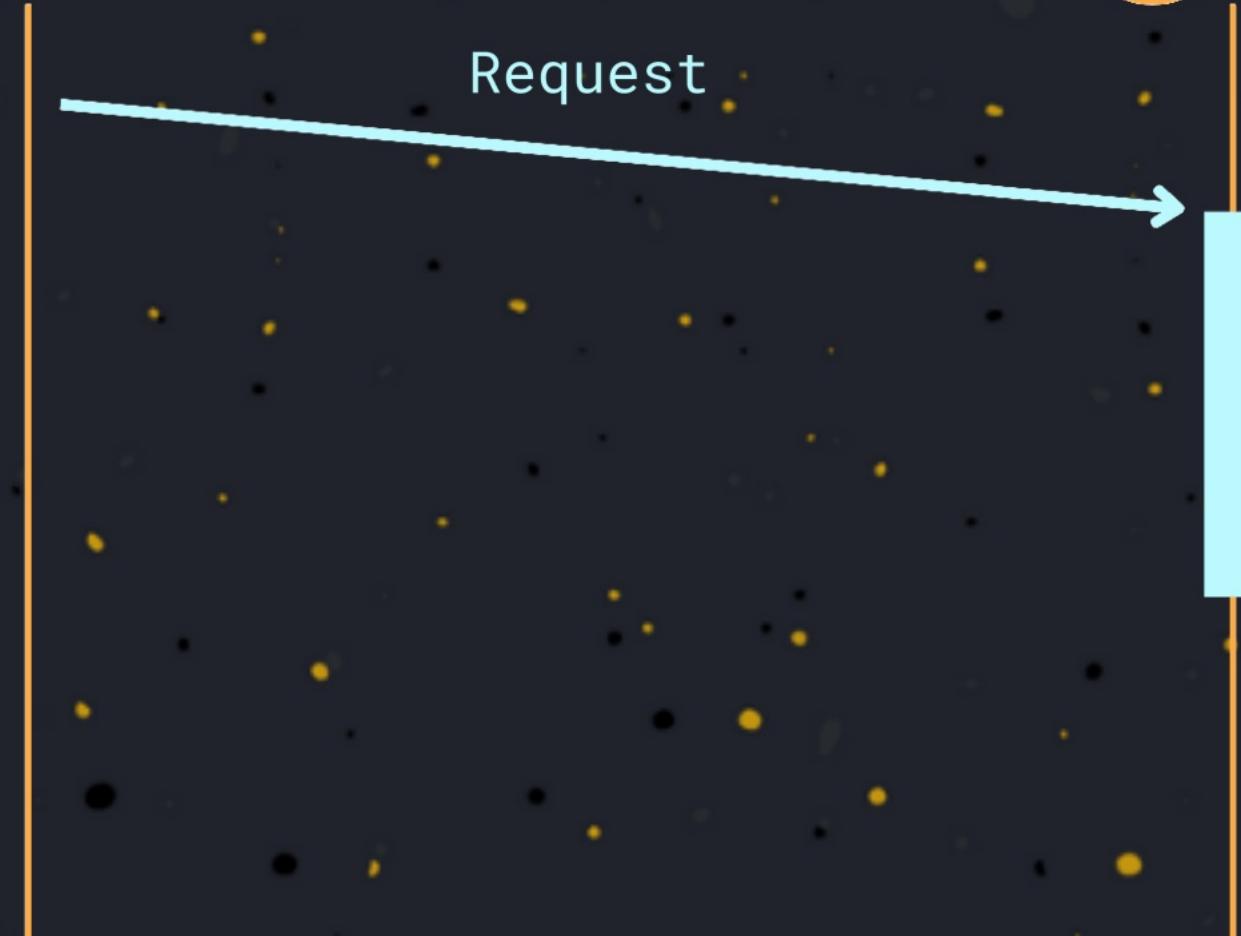
Service



Client

Server

Service



Client



Server

Service



Request



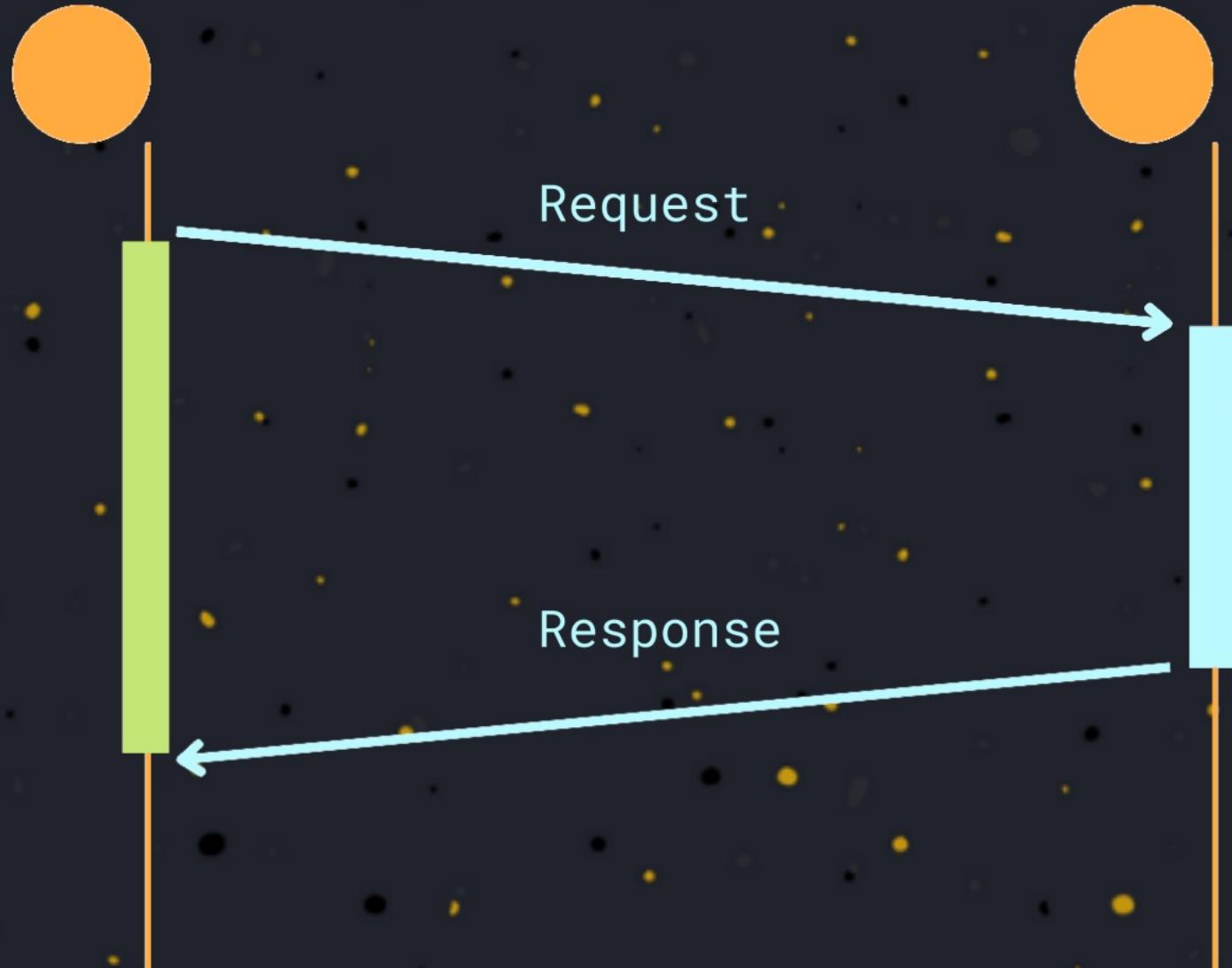
Client



Server

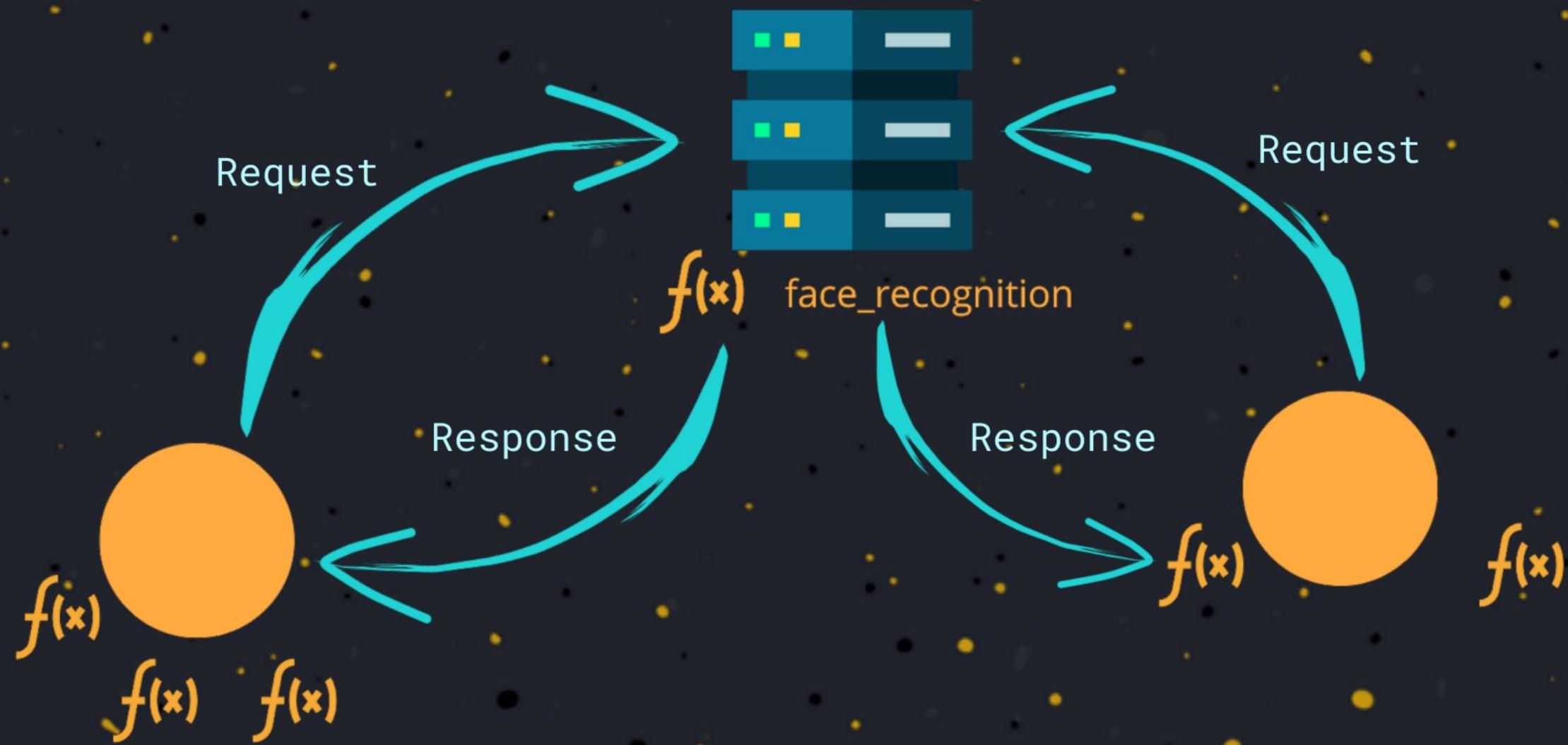


Service

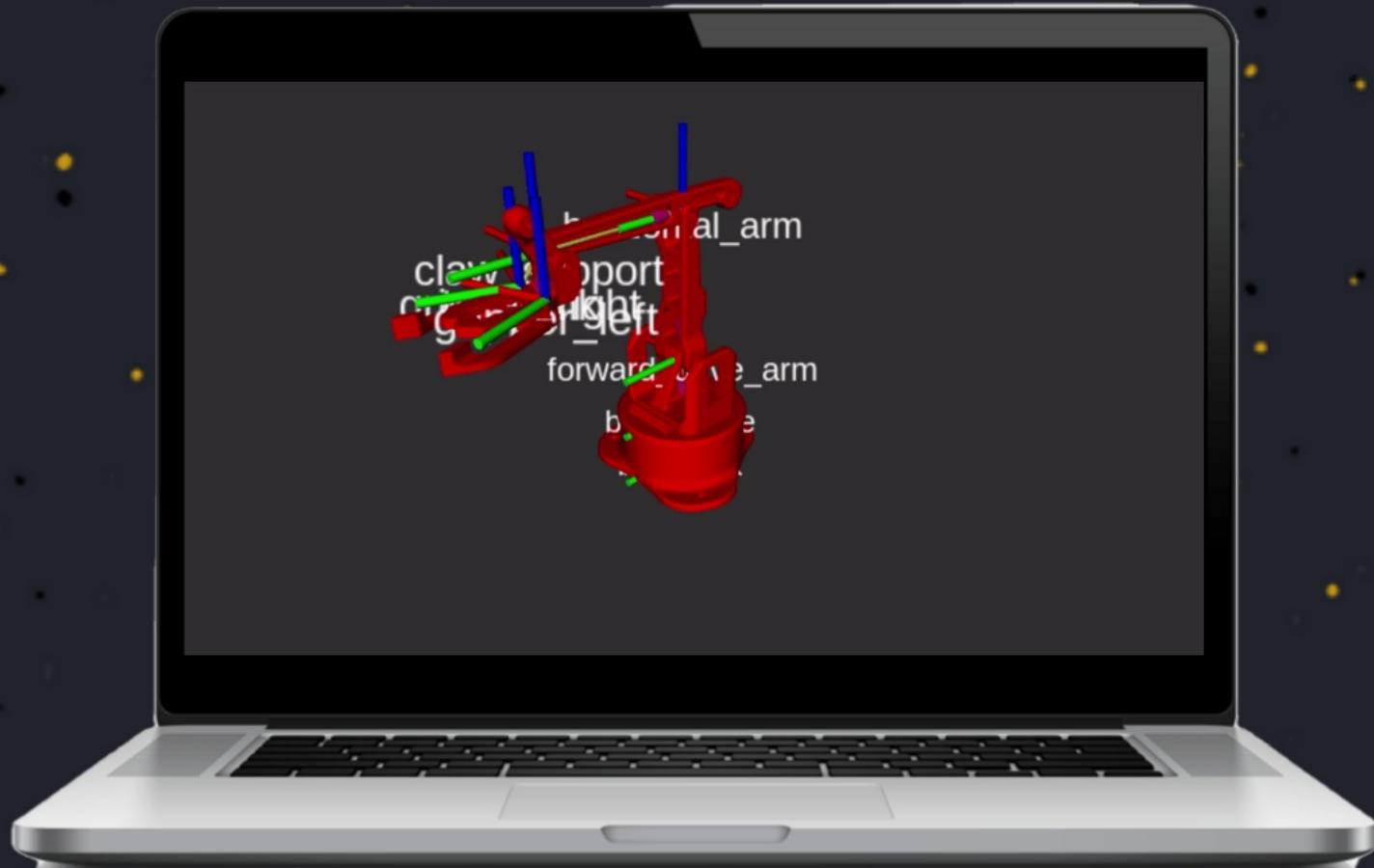


Client

Server

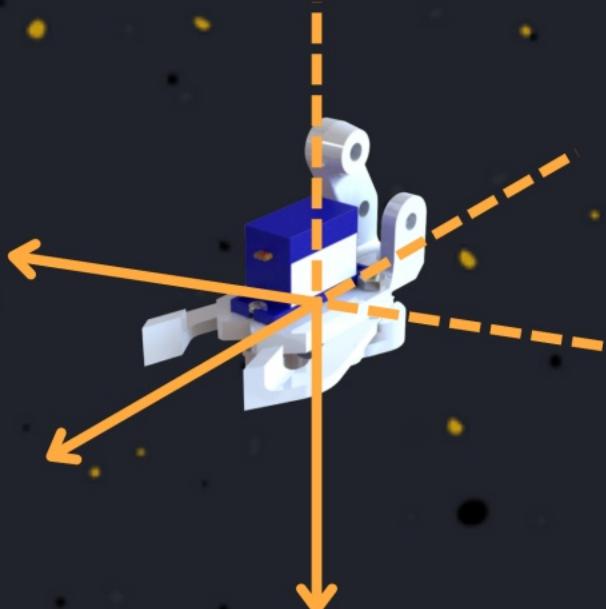


TF2 Library

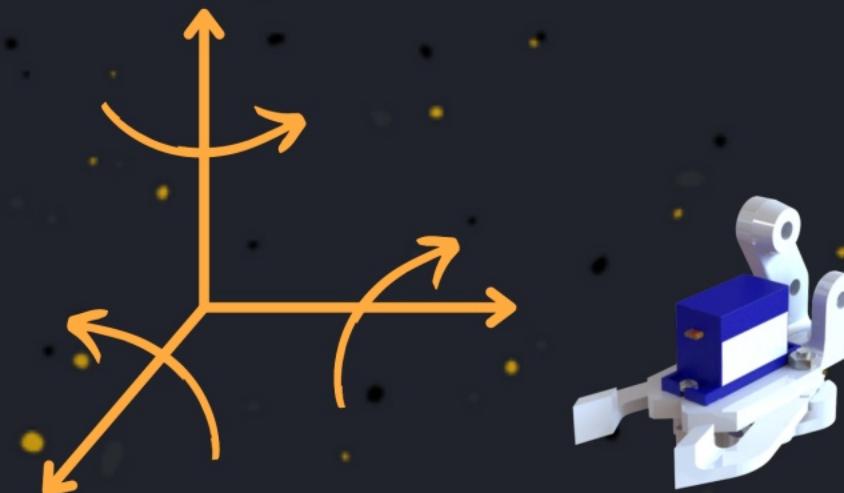


Angle Representations

Euler Angles



Quaternion



Euler Angles



Euler Angles



$$\begin{bmatrix} \cos(\vartheta) & -\sin(\vartheta) & 0 \\ \sin(\vartheta) & \cos(\vartheta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Euler Angles



$$\begin{bmatrix} \cos(\vartheta) & -\sin(\vartheta) & 0 \\ \sin(\vartheta) & \cos(\vartheta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Euler Angles



$$\begin{bmatrix} \cos(\vartheta) & -\sin(\vartheta) & 0 \\ \sin(\vartheta) & \cos(\vartheta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



$$\begin{bmatrix} \cos(\varphi) & 0 & \sin(\varphi) \\ 0 & 1 & 0 \\ -\sin(\varphi) & 0 & \cos(\varphi) \end{bmatrix}$$

Euler Angles



$$\begin{bmatrix} \cos(\vartheta) & -\sin(\vartheta) & 0 \\ \sin(\vartheta) & \cos(\vartheta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



$$\begin{bmatrix} \cos(\varphi) & 0 & \sin(\varphi) \\ 0 & 1 & 0 \\ -\sin(\varphi) & 0 & \cos(\varphi) \end{bmatrix}$$



Euler Angles



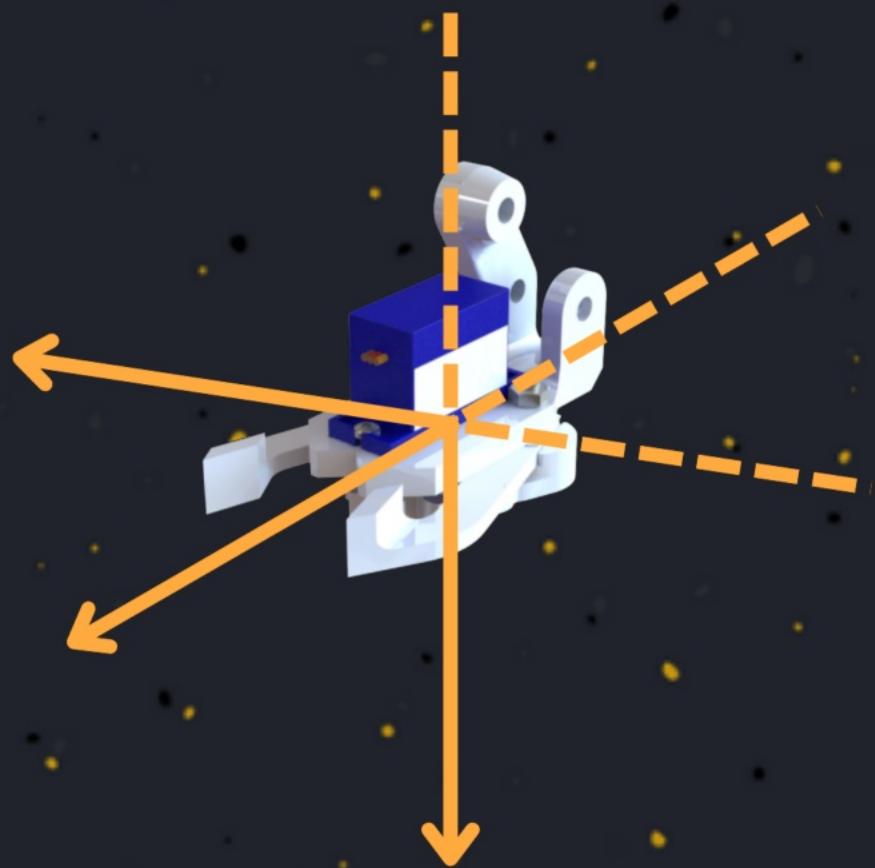
$$\begin{bmatrix} \cos(\vartheta) & -\sin(\vartheta) & 0 \\ \sin(\vartheta) & \cos(\vartheta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

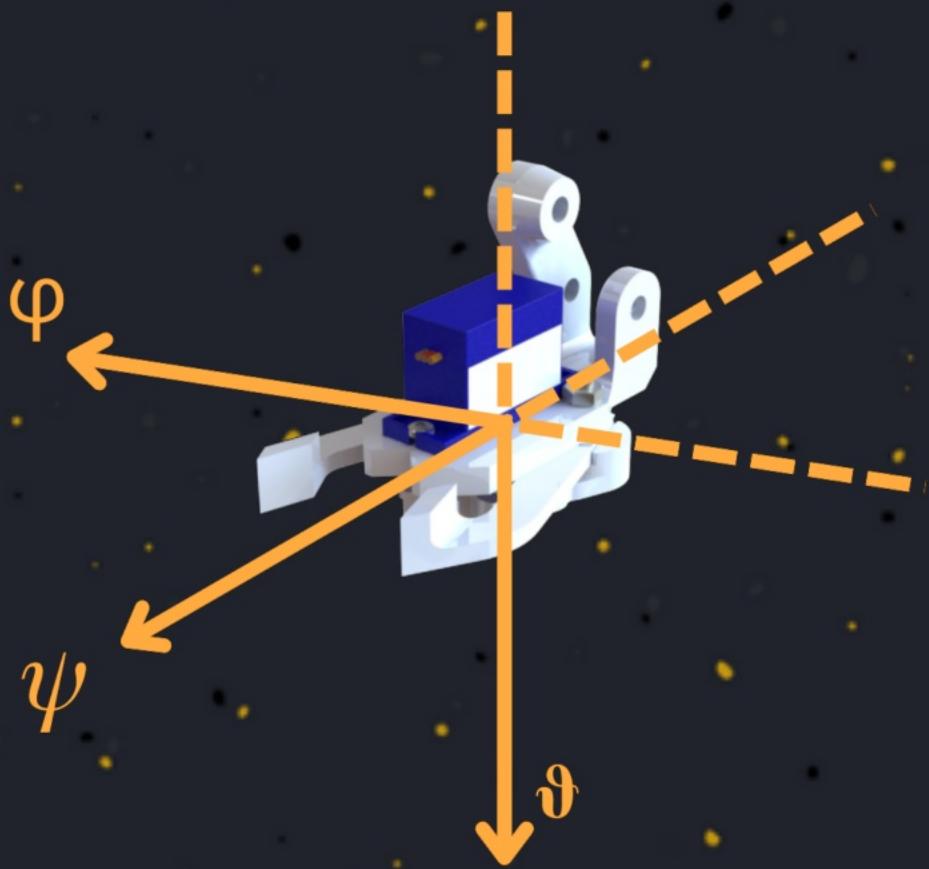
$$\begin{bmatrix} \cos(\varphi) & 0 & \sin(\varphi) \\ 0 & 1 & 0 \\ -\sin(\varphi) & 0 & \cos(\varphi) \end{bmatrix}$$

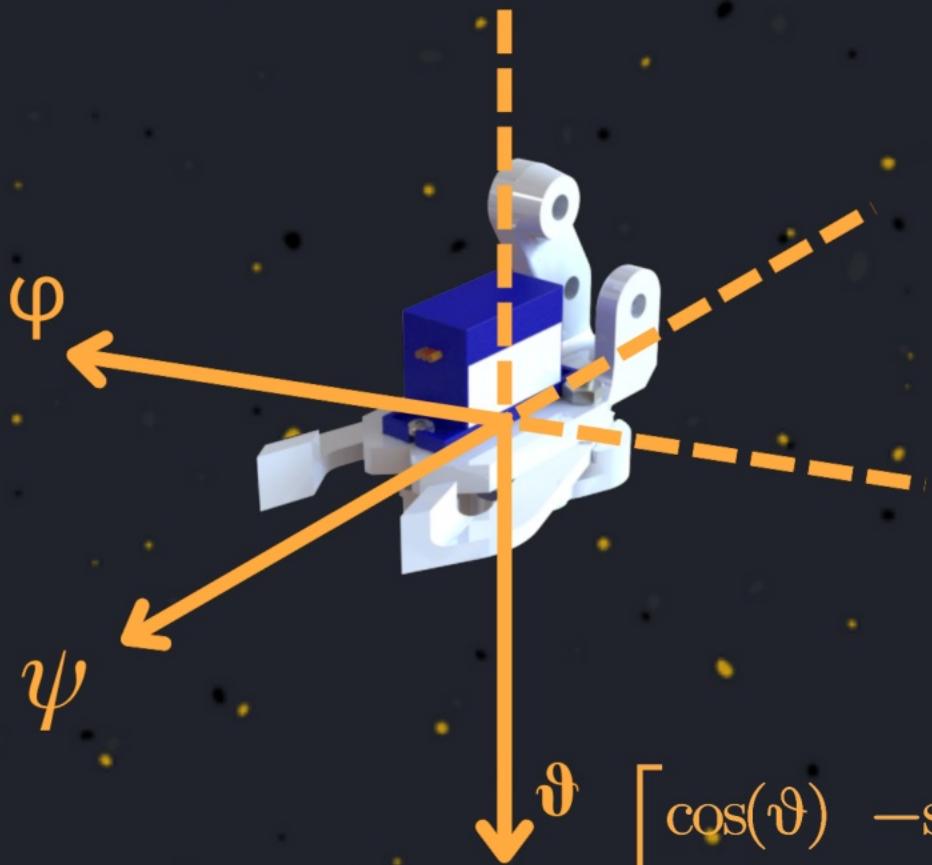


$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi) & -\sin(\psi) \\ 0 & \sin(\psi) & \cos(\psi) \end{bmatrix}$$

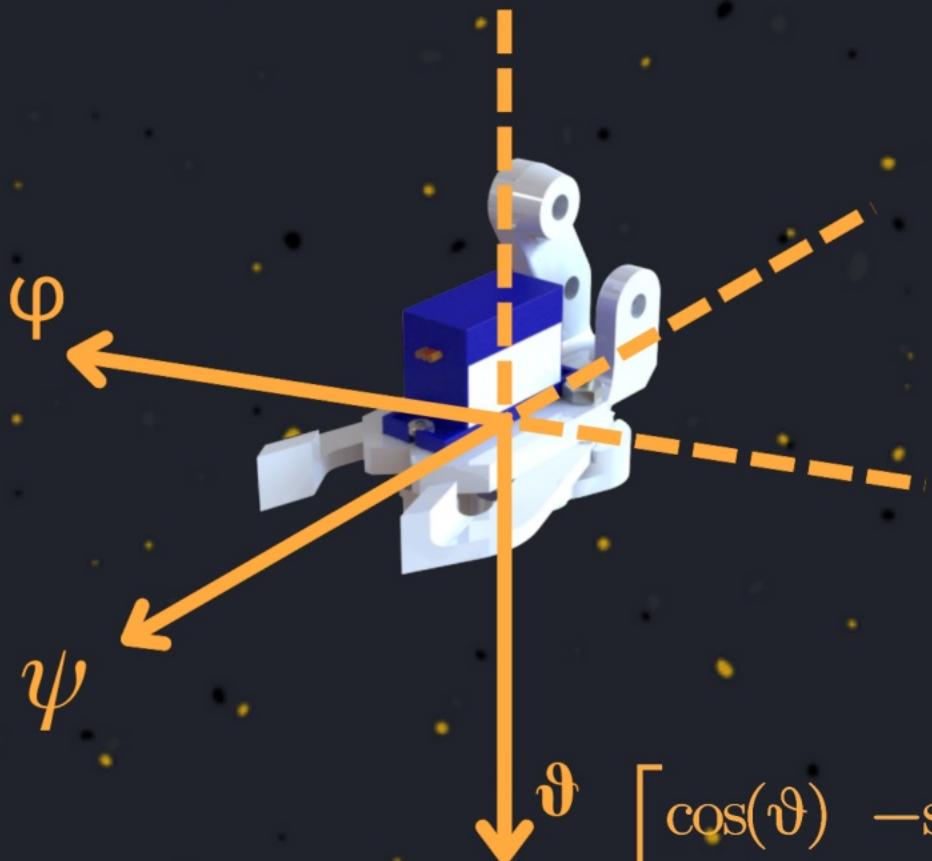








$$\begin{bmatrix} \cos(\vartheta) & -\sin(\vartheta) & 0 \\ \sin(\vartheta) & \cos(\vartheta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\varphi) & 0 & \sin(\varphi) \\ 0 & 1 & 0 \\ -\sin(\varphi) & 0 & \cos(\varphi) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi) & -\sin(\psi) \\ 0 & \sin(\psi) & \cos(\psi) \end{bmatrix}$$

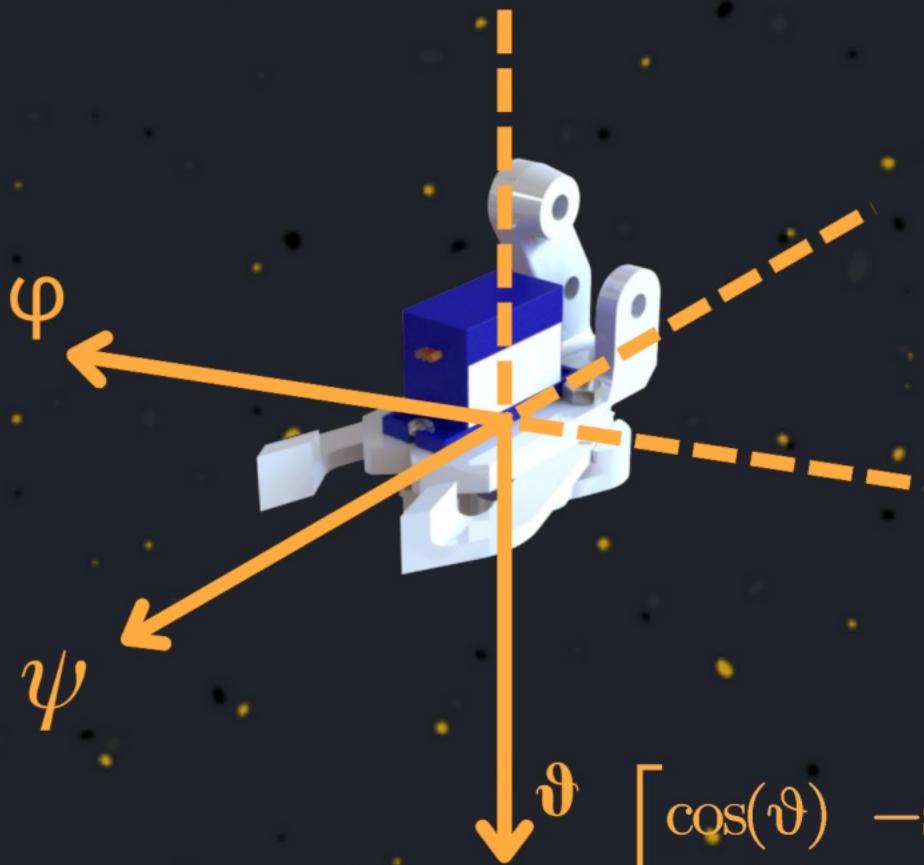


$$\begin{bmatrix} \cos(\vartheta) & -\sin(\vartheta) & 0 \\ \sin(\vartheta) & \cos(\vartheta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\varphi) & 0 & \sin(\varphi) \\ 0 & 1 & 0 \\ -\sin(\varphi) & 0 & \cos(\varphi) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi) & -\sin(\psi) \\ 0 & \sin(\psi) & \cos(\psi) \end{bmatrix}$$

Yaw

Pitch

Roll



$$\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

$$\begin{bmatrix} \cos(\vartheta) & -\sin(\vartheta) & 0 \\ \sin(\vartheta) & \cos(\vartheta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\varphi) & 0 & \sin(\varphi) \\ 0 & 1 & 0 \\ -\sin(\varphi) & 0 & \cos(\varphi) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi) & -\sin(\psi) \\ 0 & \sin(\psi) & \cos(\psi) \end{bmatrix}$$

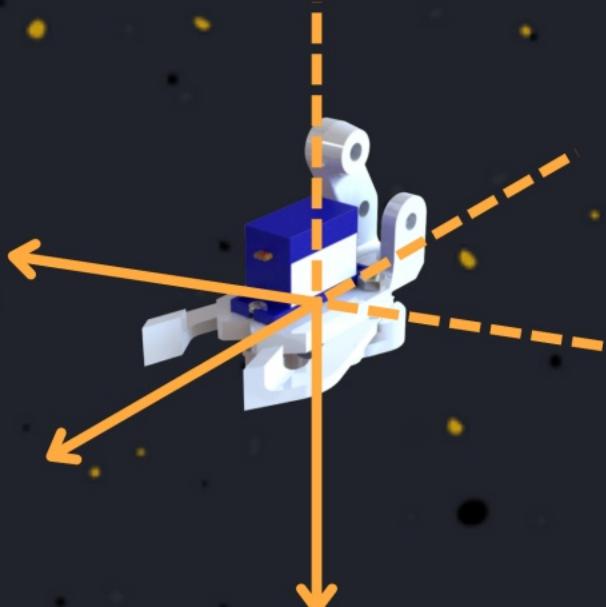
Yaw

Pitch

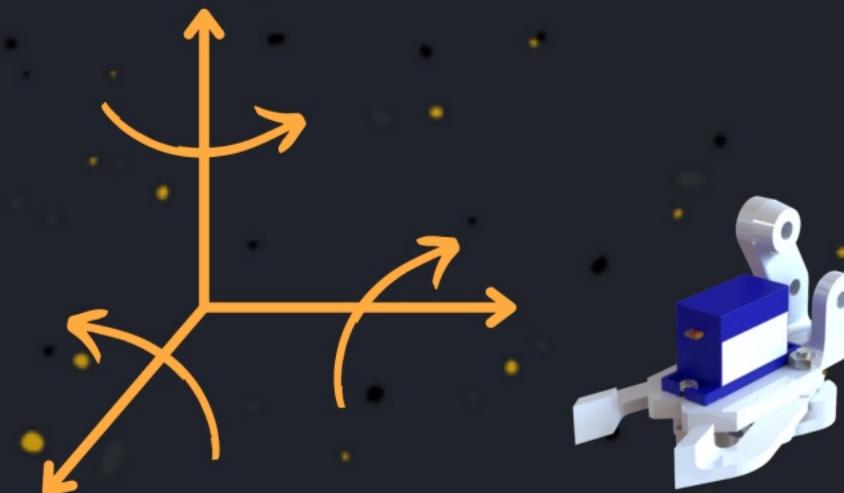
Roll

Angle Representations

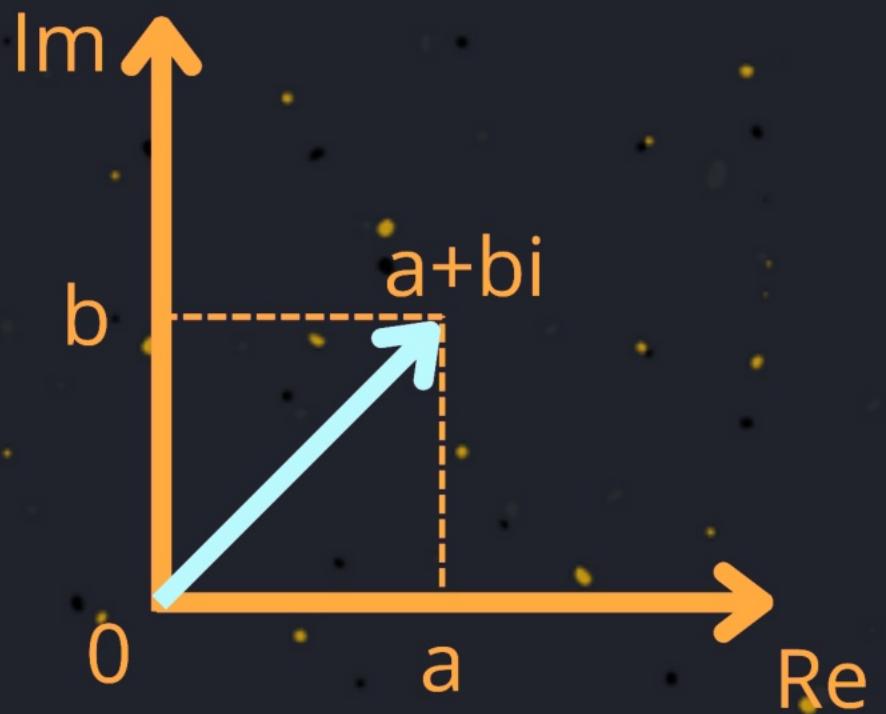
Euler Angles



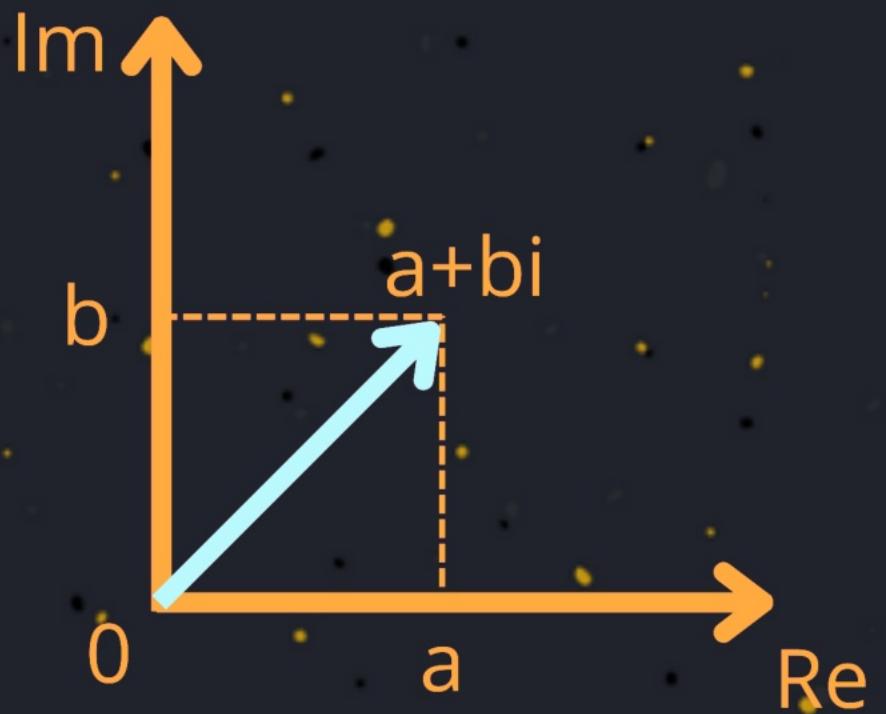
Quaternion



Quaternion

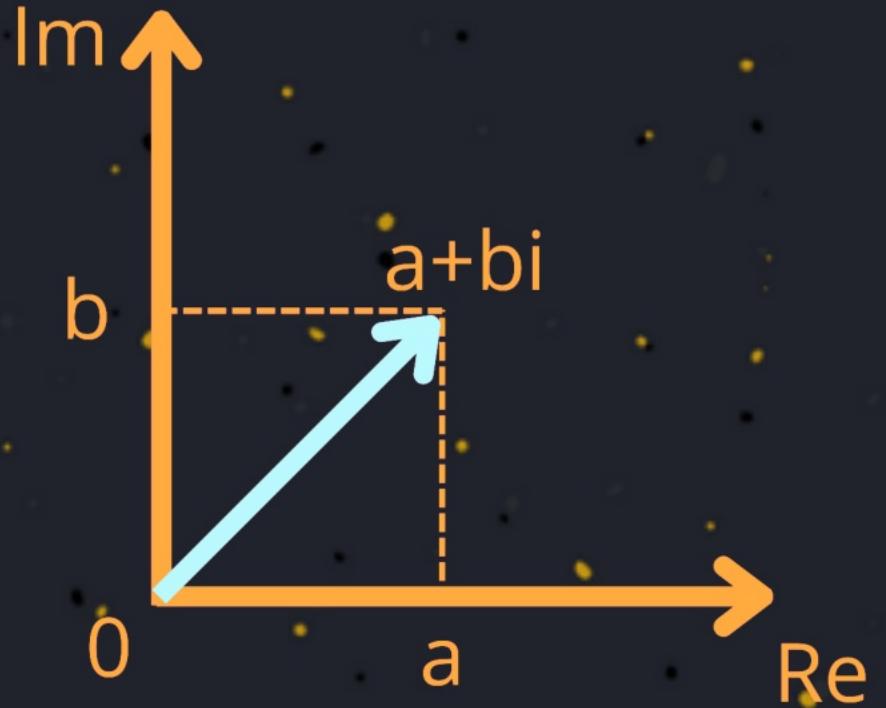


Quaternion



$$q = a + bi + cj + dk$$

Quaternion

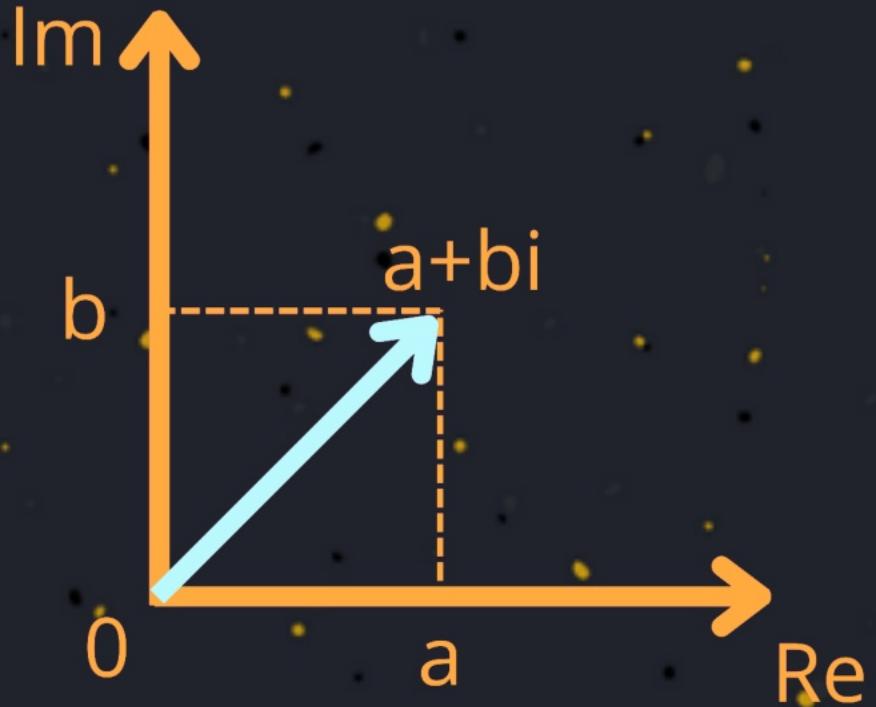


$$q = a + bi + cj + dk$$

Unitary

$$a^2 + b^2 + c^2 + d^2 = 1$$

Quaternion



$$q = a + bi + cj + dk$$

Unitary

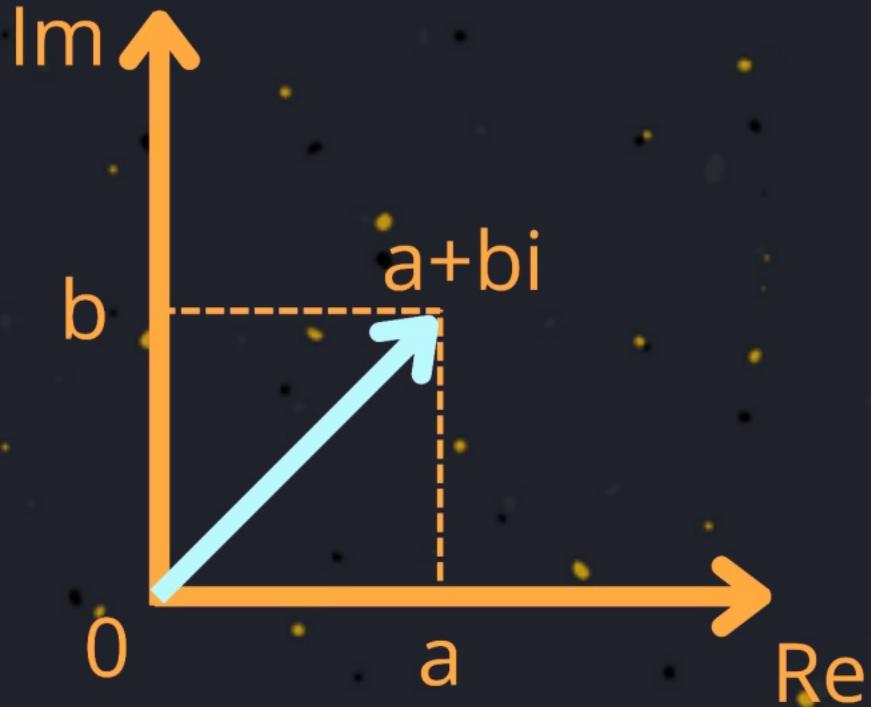
$$a^2 + b^2 + c^2 + d^2 = 1$$

Rotation Composition

$$q_1 = \begin{bmatrix} a_1 \\ b_1 \\ c_1 \\ d_1 \end{bmatrix}$$

$$q_2 = \begin{bmatrix} a_2 \\ b_2 \\ c_2 \\ d_2 \end{bmatrix}$$

Quaternion



$$q = a + bi + cj + dk$$

Unitary

$$a^2 + b^2 + c^2 + d^2 = 1$$

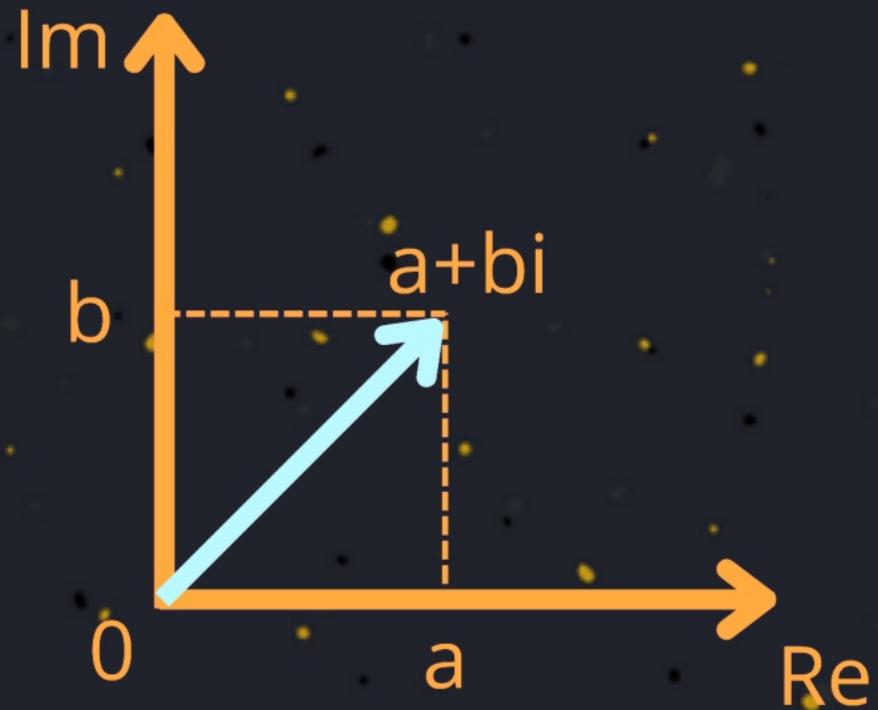
Rotation Composition

$$q_1 = \begin{bmatrix} a_1 \\ b_1 \\ c_1 \\ d_1 \end{bmatrix}$$

$$q_2 = \begin{bmatrix} a_2 \\ b_2 \\ c_2 \\ d_2 \end{bmatrix}$$

$$q_1 \cdot q_2$$

Quaternion



$$q = a + bi + cj + dk$$

Unitary

$$a^2 + b^2 + c^2 + d^2 = 1$$

Rotation Composition

$$q_1 = \begin{bmatrix} a_1 \\ b_1 \\ c_1 \\ d_1 \end{bmatrix}$$

$$q_2 = \begin{bmatrix} a_2 \\ b_2 \\ c_2 \\ d_2 \end{bmatrix}$$

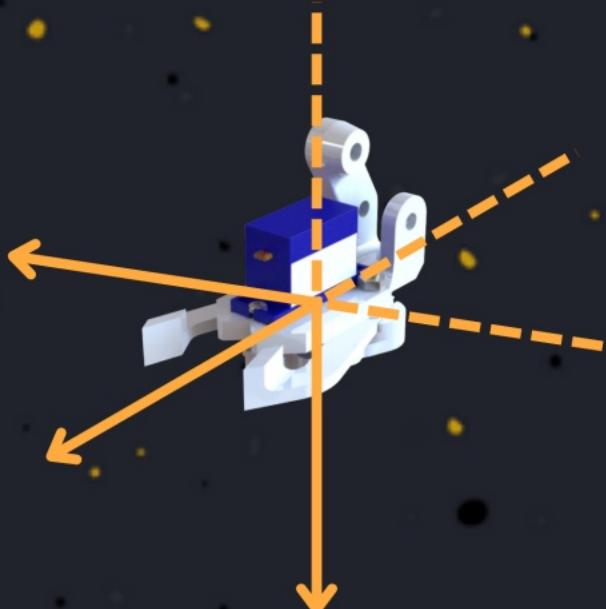
$$q_1 \cdot q_2$$

Inverse

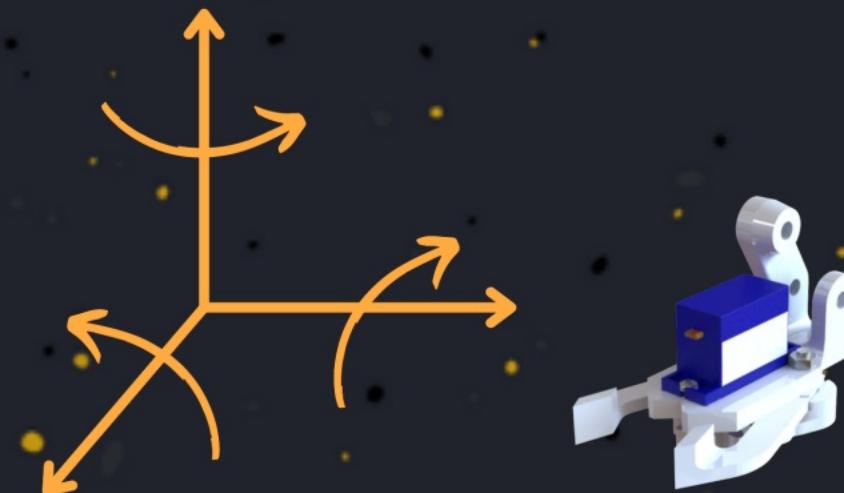
$$q^{-1} = a - bi - cj - dk$$

Angle Representations

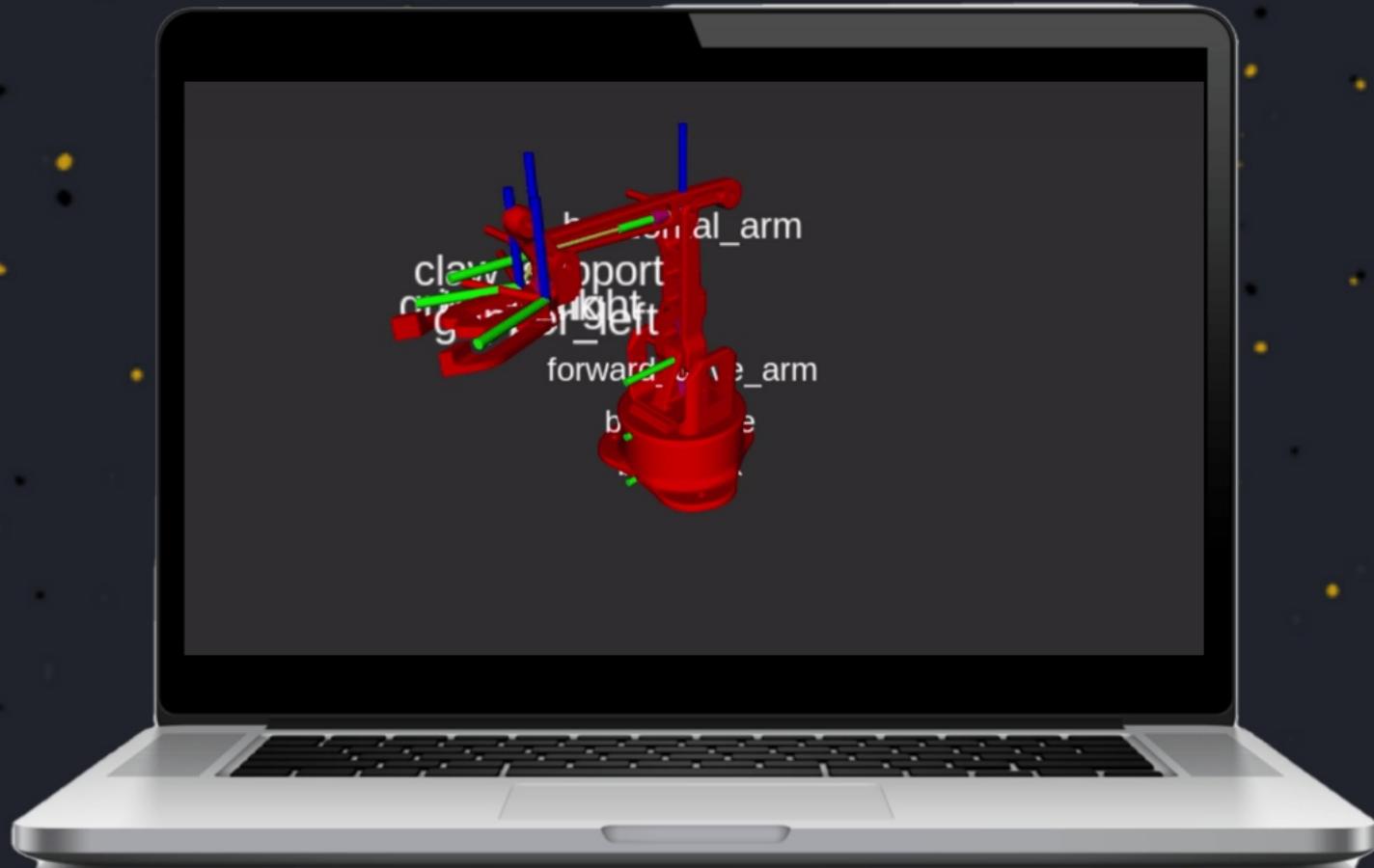
Euler Angles



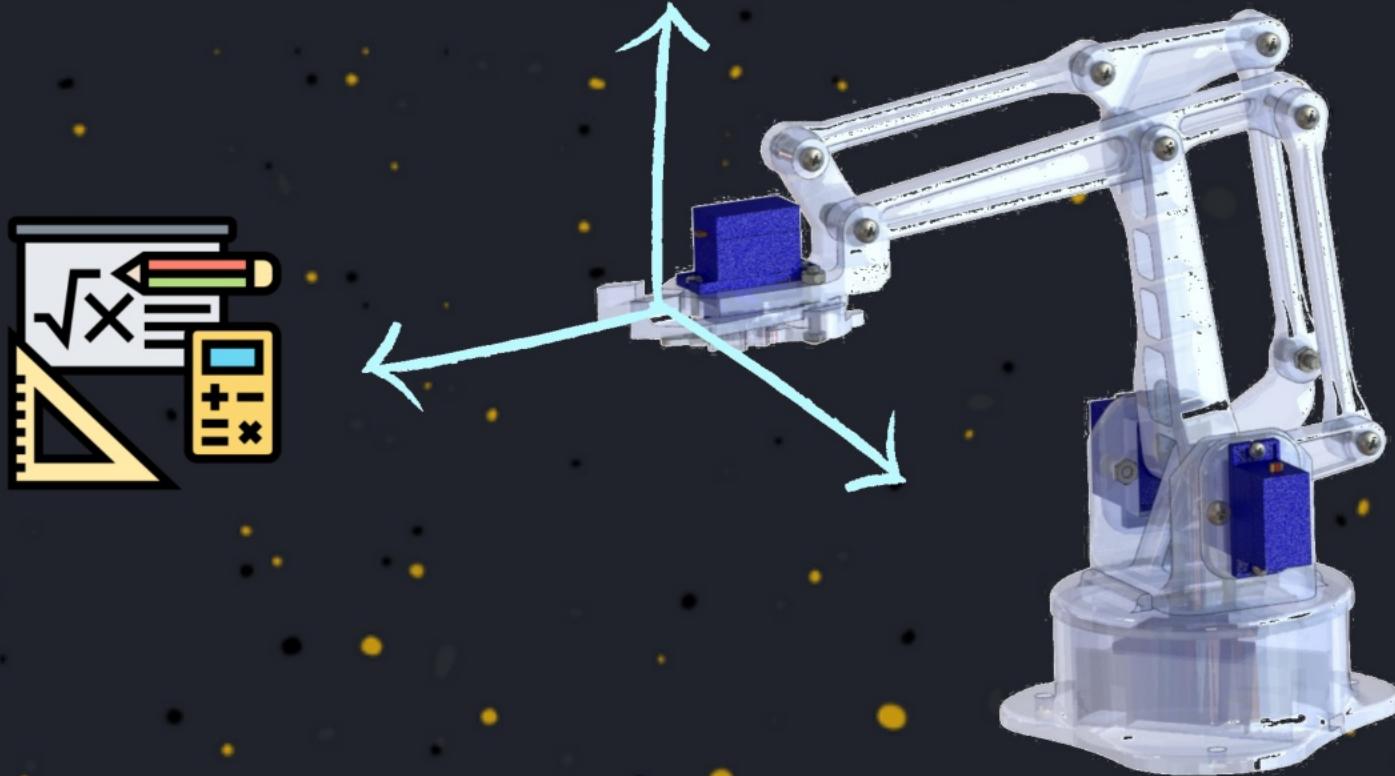
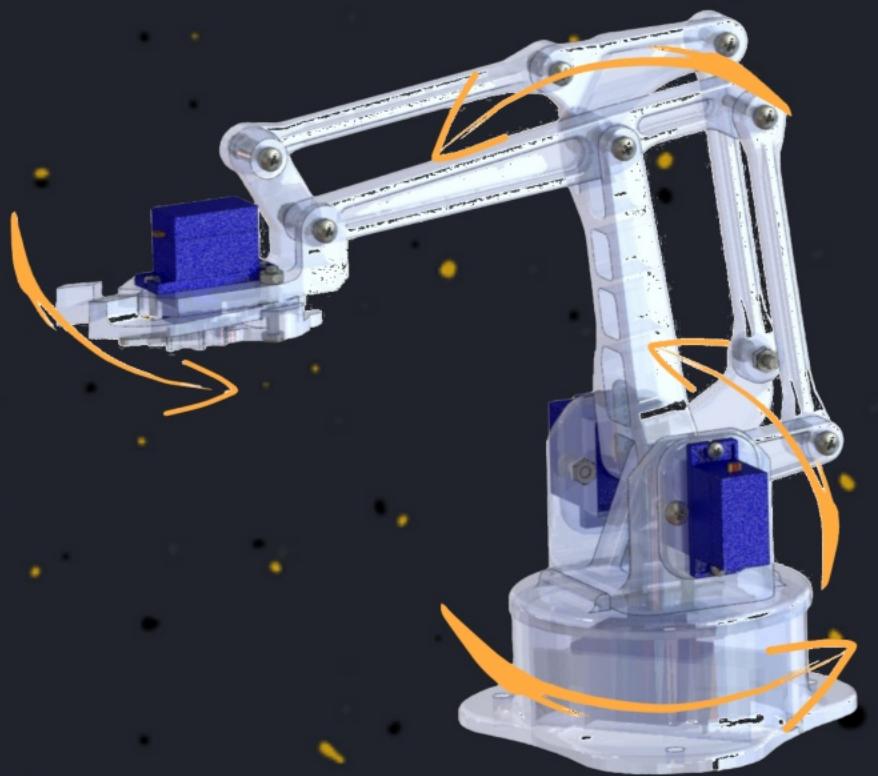
Quaternion



TF2 Library



Forward

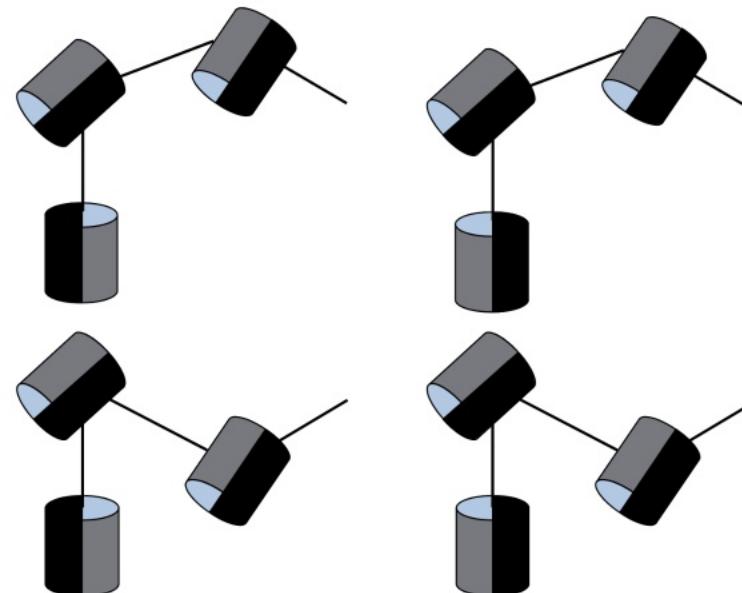


Inverse



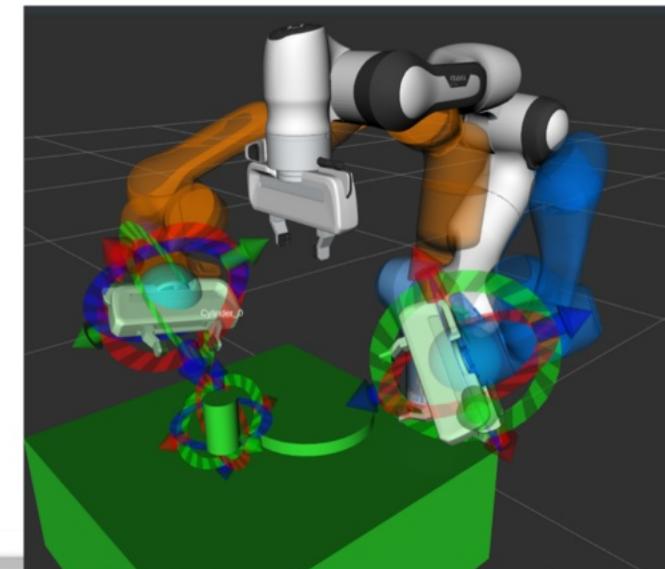
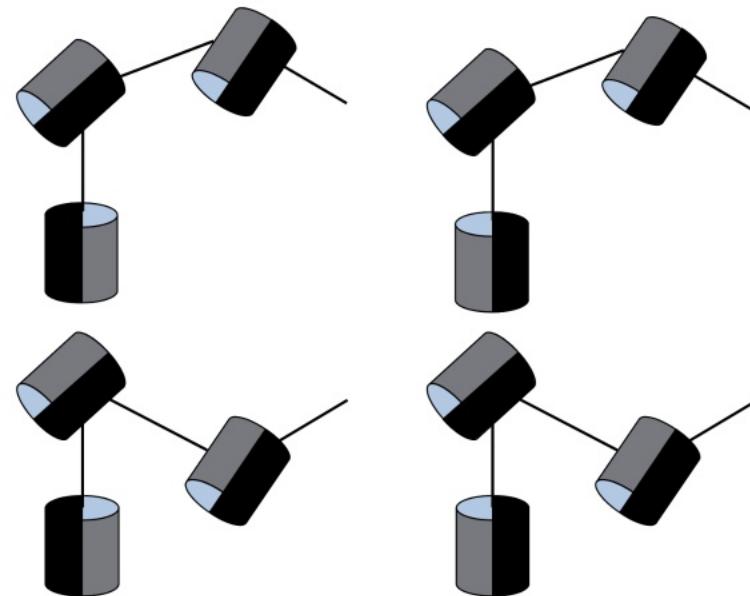
Inverse Kinematics

- Multiple Solutions
- Infinite Solutions
- No Solutions



Inverse Kinematics

- Multiple Solutions
- Infinite Solutions
- No Solutions

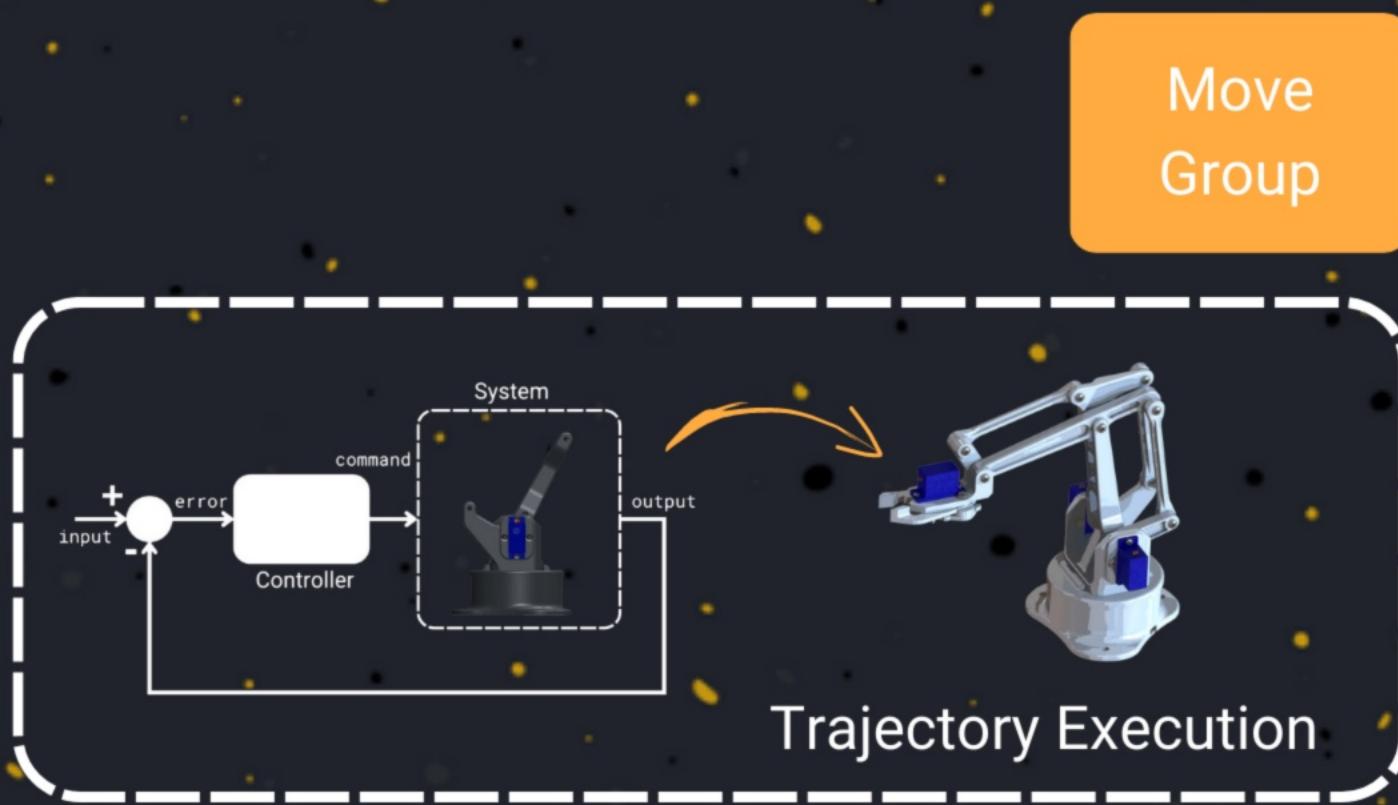


> **MoveIt2**

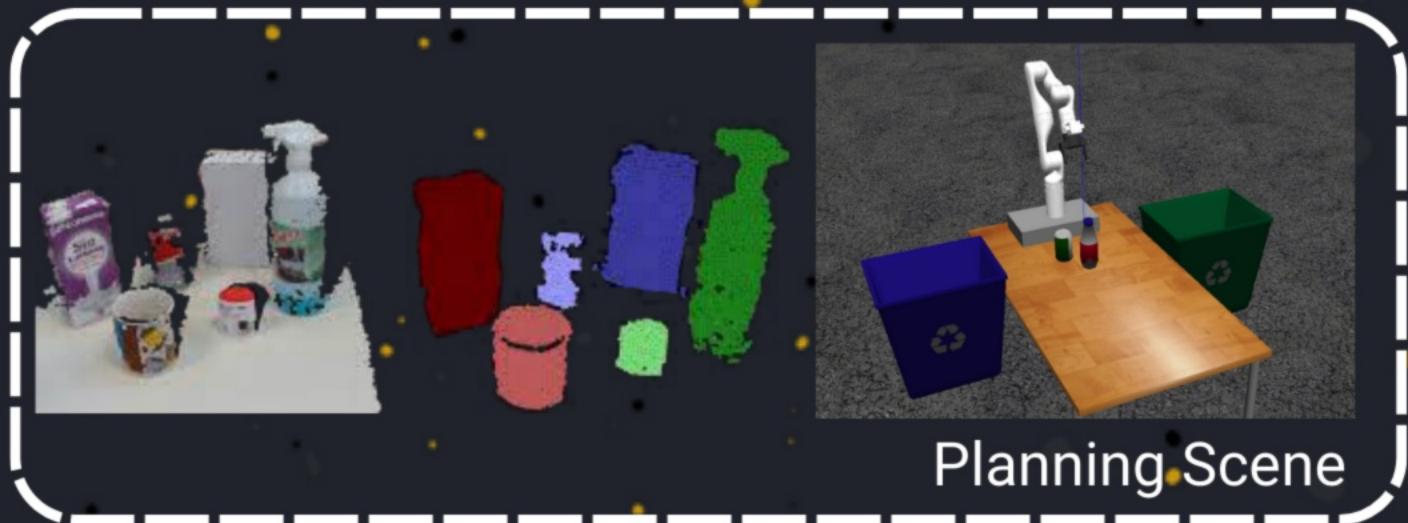
> MoveIt2

Move
Group

> MoveIt2

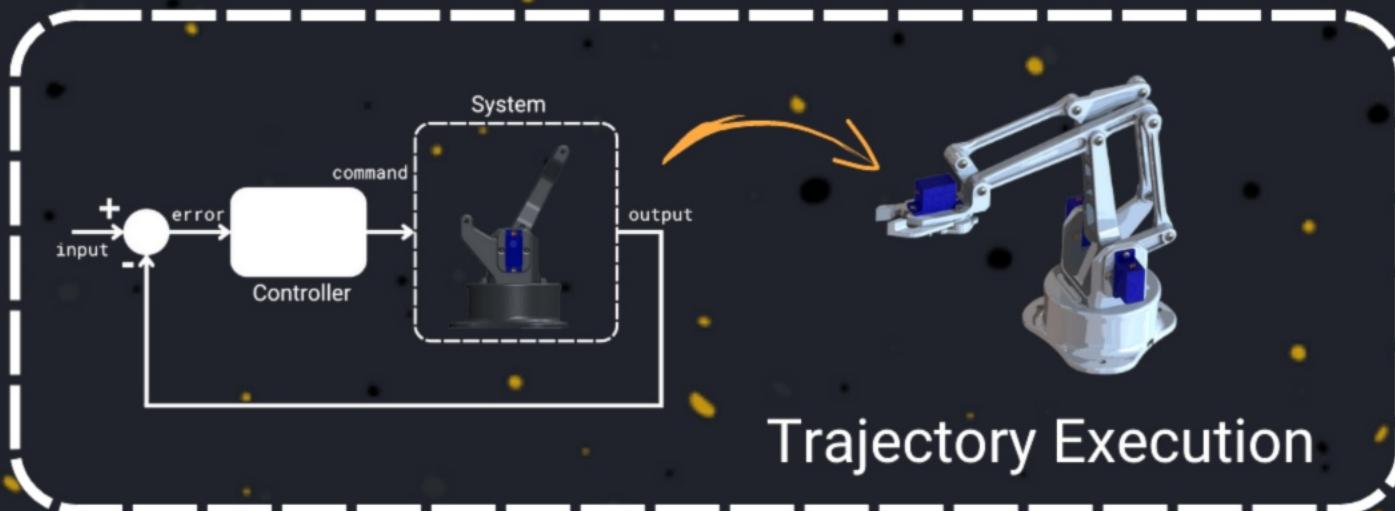


> MoveIt2



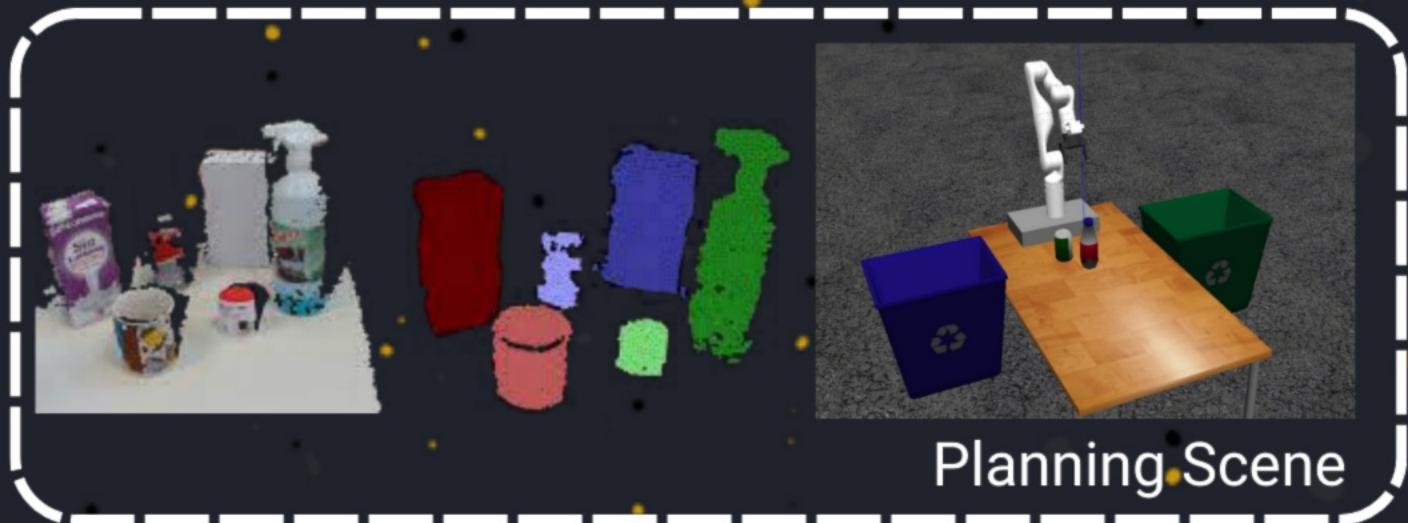
Planning Scene

Move
Group



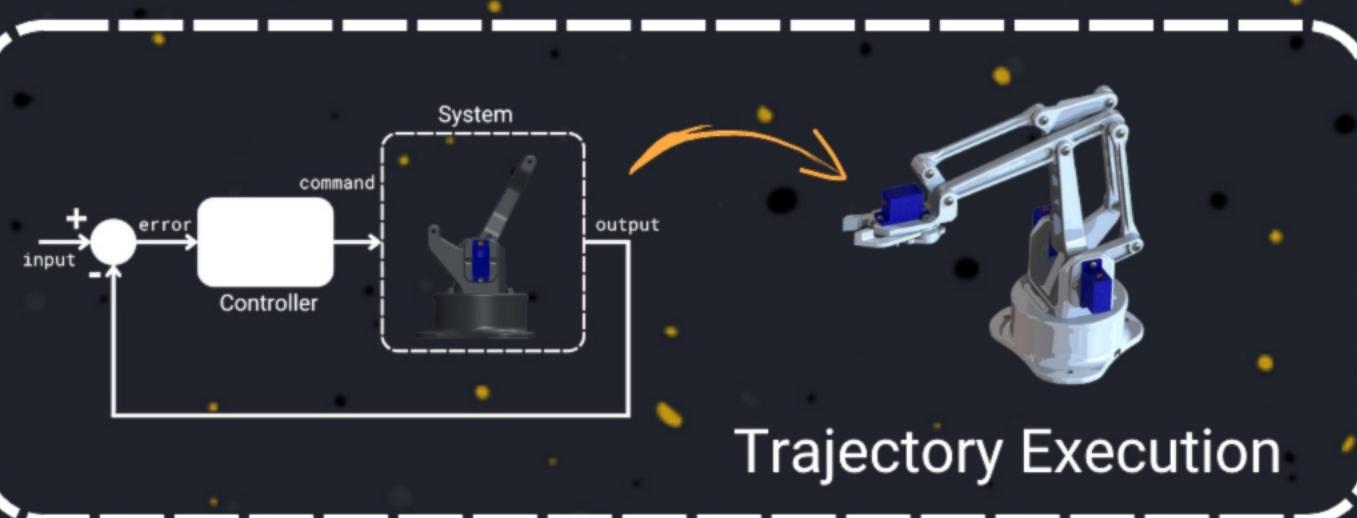
Trajectory Execution

> MoveIt2

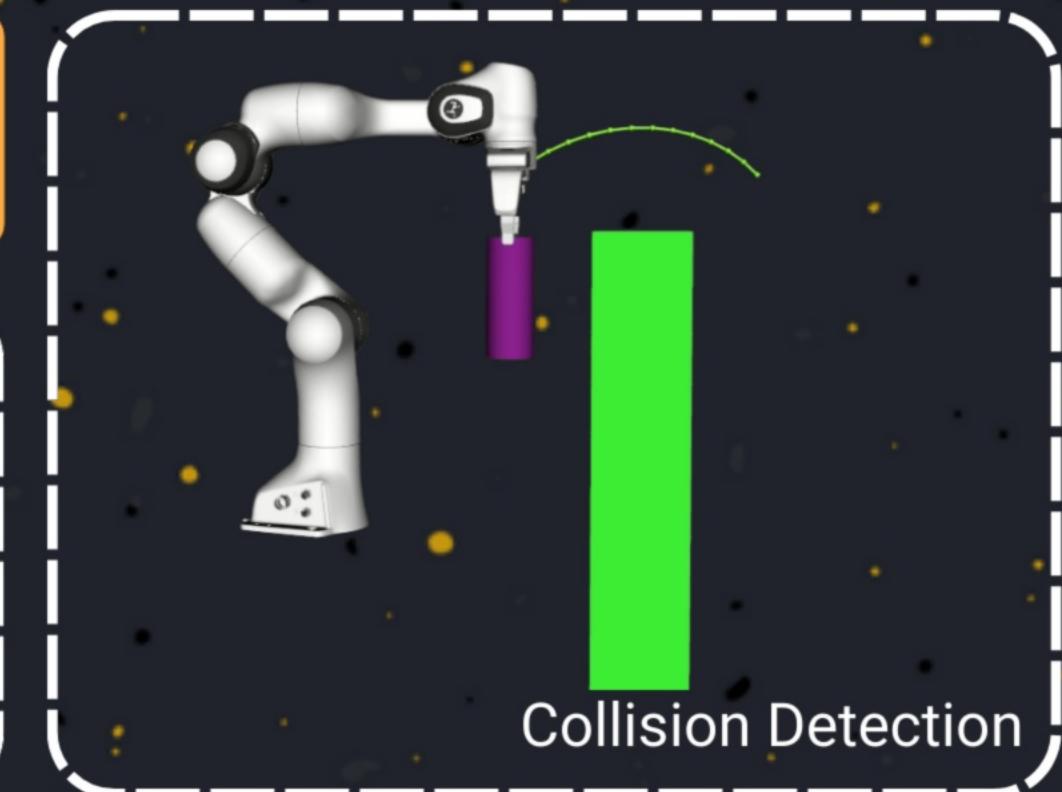


Planning Scene

Move Group



Trajectory Execution



Collision Detection

> MoveIt2

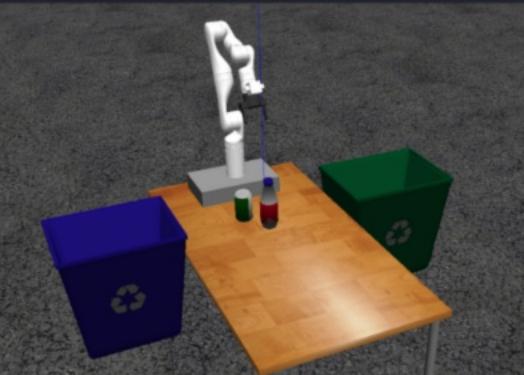


User Interface / API

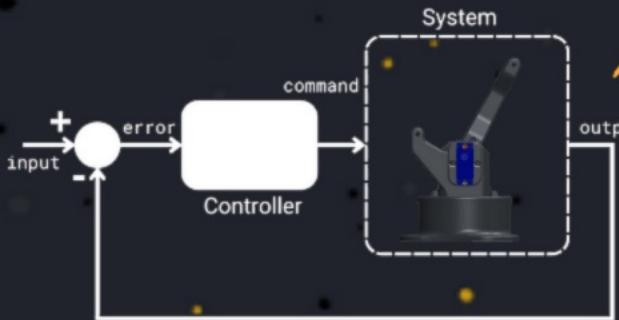
alien@alien: ~
alien@alien: ~

http://localhost:1333

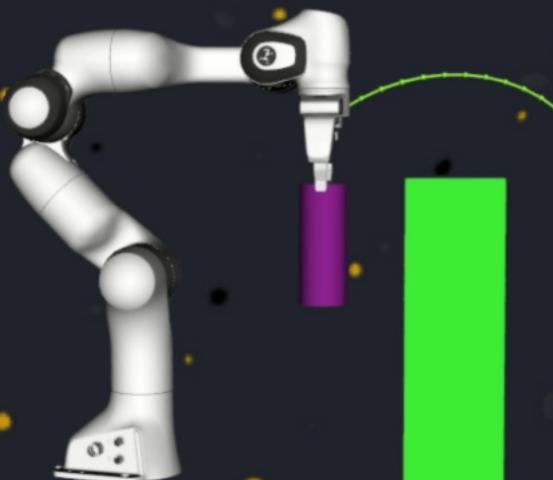
Move
Group



Planning Scene



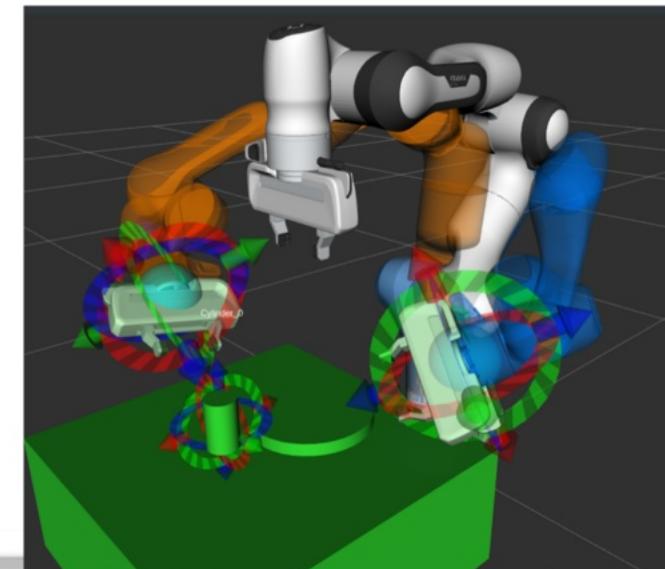
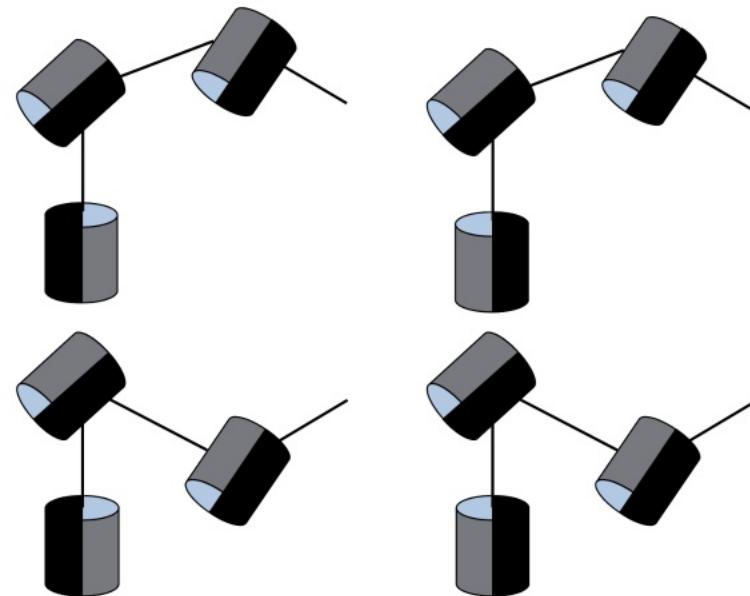
Trajectory Execution



Collision Detection

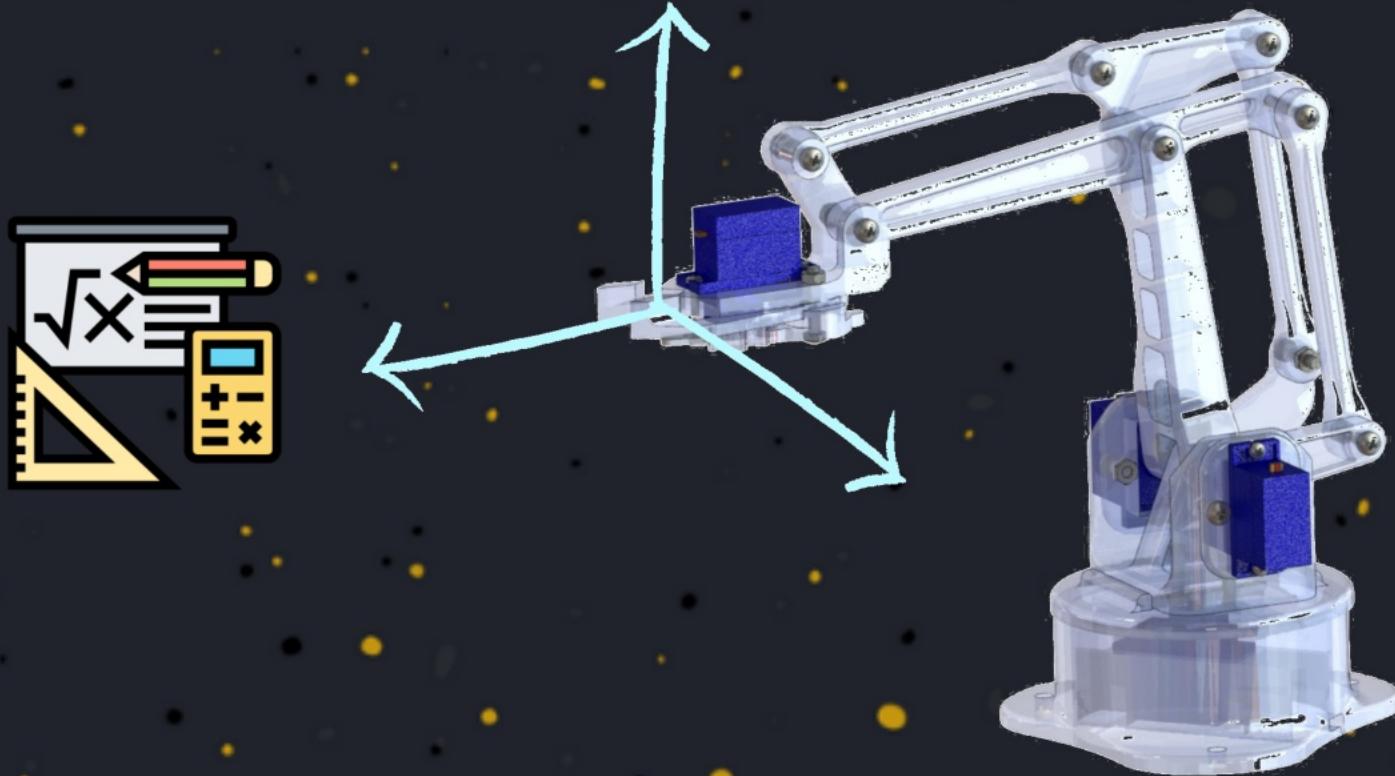
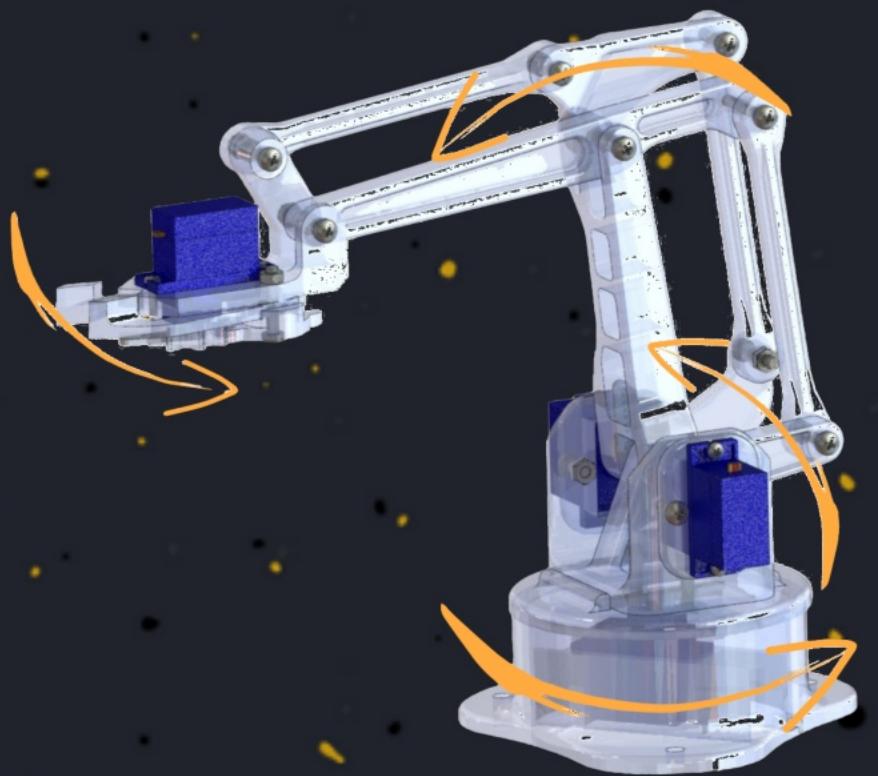
Inverse Kinematics

- Multiple Solutions
- Infinite Solutions
- No Solutions



> **MoveIt2**

Forward



Inverse



Arduinobot

