Robotics

Spring 2023

Homework3

Arya Ebrahimi 9822762175

Part I:

Question1

$$T_{rot(x,90)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ T_{trans(d)} = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ T_{rot(z,90)} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

a)

$$T = T_{rot(z,90)} T_{trans(d)} T_{rot(x,90)}$$

$$= \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & -2 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & -2 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 2 \\ 0 \\ 1 \end{bmatrix} \Rightarrow x_p = -2, y_p = 3, z_p = 7$$

b)

The order of multiplication of Transform matrices from left to right in this case (fixed frame) is to start from last Transform matrix to the first one. The reason for this will be discussed in question3 of Part I.

c)

$$T = T_{rot(x,90)} T_{rot(z,90)} T_{trans(d)}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & -2 \\ 0 & 0 & -1 & -5 \\ 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & -2 \\ 0 & 0 & -1 & -5 \\ 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 2 \\ 0 \\ 1 \end{bmatrix} \Rightarrow x_p = -4, y_p = -5, z_p = 3$$

It can be concluded that the order of applying the transformations has an effect on the result point, and applying the same transformations in a different order causes different transformations.

Question2

$$T_{rot(x,90)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ T_{trans(d)} = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ T_{rot(z,90)} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

a)

$$T = T_{rot(x,90)} T_{trans(d)} T_{rot(z,90)}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & -1 \\ 0 & 0 & -1 & -5 \\ 1 & 0 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & -1 \\ 0 & 0 & -1 & -5 \\ 1 & 0 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 2 \\ 0 \\ 1 \end{bmatrix} \Rightarrow x_p = -3, y_p = -5, z_p = 6$$

b)

The order of multiplication of Transform matrices from left to right in this case (current frame) is to start from first Transform matrix to the last one (opposite of the fixed frame situation).

Question3

In the current frame situation, we multiply transform matrices based on the matrix composition one after the other. But, in the case of the fixed frame, because we want to do a transformation based on the previous frame, we should use Similarity Transformations to find the transform in the current frame based on the desired transform on the base frame.

For the current frame situation we have $R_2^0 = R_1^0 R_2^1$, that can be generalized to: $T_2^0 = T_1^0 T_2^1$.

But for the fixed frame we have to use similarity transformations, so the formula can be written as follows: $R_2^0 = R_1^0[(R_1^0)^{-1} R_2^1 (R_1^0)] = R_2^1 R_1^0$, and $T_2^0 = T_2^1 T_1^0$.

Qustion4

1. For the revolute joints, $\hat{z}$ should be the axis of rotation, and for prismatic joints, it should be in the direction of movement.

2. There are two joint parameters in the DH system which are d_i and θ_i. d_i is the distance between x_{i-1} and x_i relative to the z_i axis. θ_i is the angle between x_{i-1} and x_i about the z_i axis. we consider one of them to be variable in each joint. If the joint is revolute, then θ_i will be variable, and d_i should be a constant, and if we have a prismatic joint, then it is most likely for d_i to be variable.

3. Skew lines are two lines that do not intersect and are not parallel. Consider the rectangular parallelepiped shown in Figure 1, The line through segment AD and the line through segment B_1B are skew lines, but there is a line AB which is perpendicular to both lines.

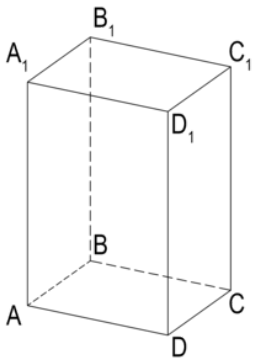

Figure 1: Rectangular parallelepiped.

4. x_i must be perpendicular to both z_i and z_{i-1}, and its direction have to be from smaller i to the larger i. If z_i and z_{i-1} intersect at some point, then the direction of x_i is not important, and it just needs to satisfy the orthogonality condition.

5. α_i and a_i are the link parameters and are constant. If the joint is revolute, then d should be constant as well, and if the joint is prismatic, then θ would be constant, and d is variable. α_i is the angle between z_i and z_{i+1} about the x_i. The Right-hand rule can be used to find if the value is positive or negative. The thumb should be in line with x_i, and the direction of hand closure determines the positive angle. a_i is the link's length, which is the distance between z_i and z_{i+1} alongside the x_i axis.

6. The first method is the one that is used by Oussama Khatib and considers the link's frame on the starting part of the link. For example, if we have a simple RRR manipulator robot, then it is likely for the f_1 and f_0 origins to be the same. The second method that is used by the textbook is to consider a link's frame at the end of the link, which results in a more straightforward DH table. The order of transforms differs in these methods.
   Oussama Khatib method: $T = T_{rot(x,\alpha_{i-1})}T_{trans(x,a_{i-1})}T_{rot(z,\theta_i)}T_{trans(z,d_i)}$
   Textbook method: $T = T_{rot(z,\theta_i)}T_{trans(z,d_i)}T_{rot(x,\alpha_i)}T_{trans(x,a_i)}$

Question5

a)

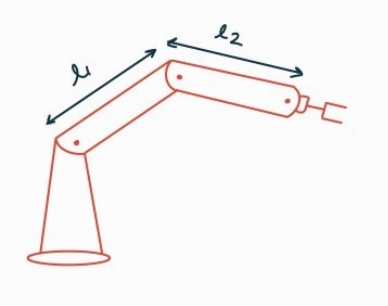

Figure 2: RRR robot.

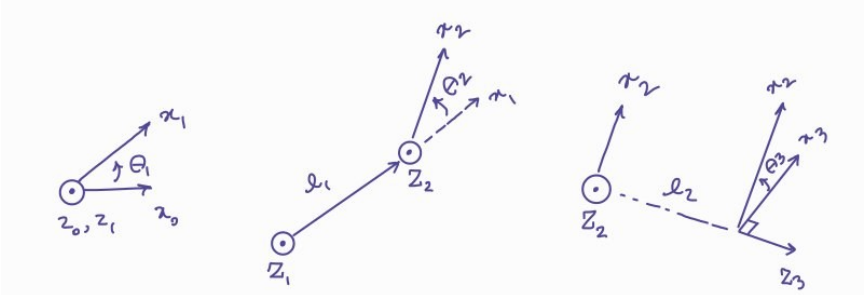

Figure 3: Proposed frames for the RRR robot.

Based on the slides,
1) a_i is the distance between z_i and z_{i+1} along x_i
2) α_i is the angle between z_i and z_{i+1} about x_i
3) d_i is the distance between x_{i-1} and x_i along z_i
4) θ_i is the angle between x_{i-1} and x_i along z_i

Since z_0 and z_1 are on top of each other, then a_{i-1} and α_{i-1} would be zero, and because it is a revolute joint, we haveθ_1 that indicates the rotation.
The distance between z_1 and z_2 relative to the x_1 is l_1 , so a in the second row would be equal to l_1. Since this joint is also revolute, the value of theta would be variable(θ_2).

Table 1: DH parameters for the RRR robot.

i	a_{i-1}	α_{i-1}	d_i	θ_i
1	0	0	0	θ_1^*
2	l_1	0	0	θ_2^*
3	0	90°	l_2	θ_3^*

For the next joint, the rotation's direction is changed, so z_3 differs from previous ones. z_3 is the result of rotating z_2 about x_2 for 90°, so the value of α_{i-1} is 90°. The distance between x_2 and x_3 along z_3 is l_2; thus, the d_3 is equal tol_2, and again since this is a revolute joint θ_3 would be variable.

b)

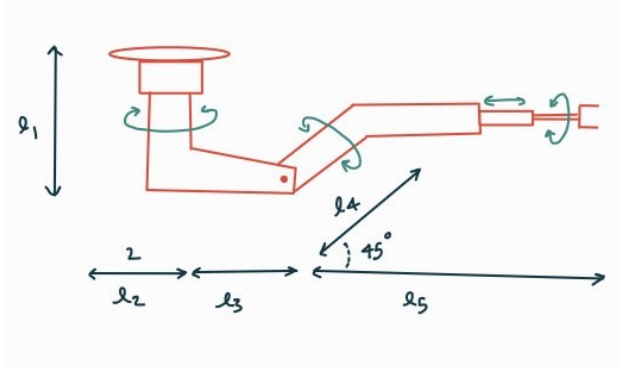

Figure 4: RRPR robot.

Table 2: DH parameters for the RRPR robot.

i	a_{i-1}	α_{i-1}	d_i	θ_i
1	0	0	l_1	θ_1^*
2	l_3	-90°	0	θ_2^*
3	0	-90°	$\frac{\sqrt{2}}{2}l_4$	-90°
4	0	90°	$l_5 + d^*$	0
5	0	0	0	$-90 + \theta_5^*$

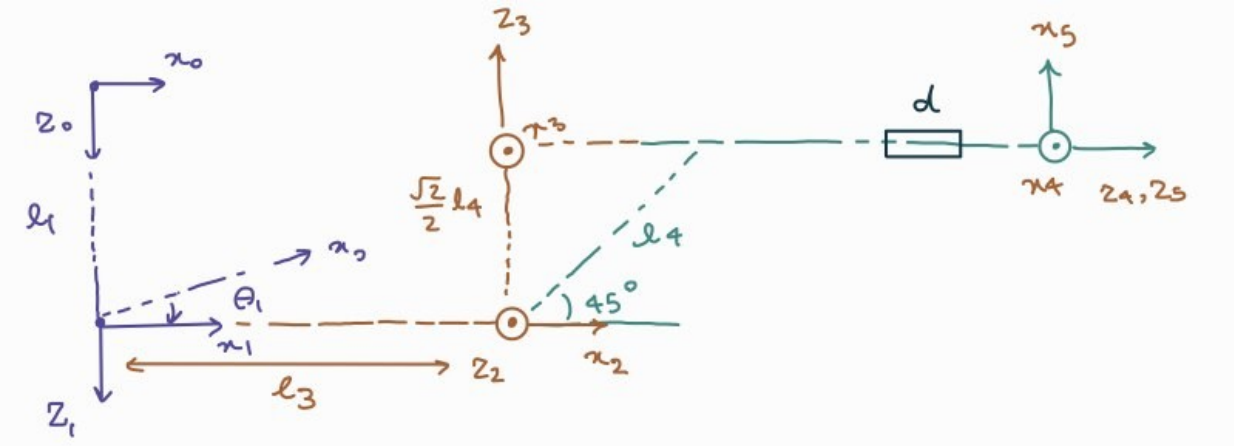

Figure 5: Proposed frames for the RRPR robot.

z_1 should be the rotation axis of the first joint, so the direction of it would be DOWN. The axis of rotation for the second joint differs from the first one, so z_1 should be rotated. z_2 is the result of rotating z_1 about x_1 for -90°. The distance between z_1 and z_2 along x_1 (a_{i-1}) is l_3.

The next joint is prismatic, so the direction of the next z should be alongside the prismatic joint, which is RIGHT, but x_2 must be perpendicular to the next z. It is impossible to satisfy both conditions, so in this step, we only add an offset (based on the l_4 and $\cos 45^\circ$) for the sake of calculations, and 90° rotation of z, making it possible for the following steps to rotate z direction to the RIGHT.

By rotating z_3 about x_3 for 90°, z_4 would be in the desired direction. The distance between x_3 and x_4 along the z_4 is l_5, and there is a prismatic joint on the link so that d_4 would be variable.the origins of $f4$ and $f5$ intersects which makes a_4 and d_5 zero, since z_4 and z_5 are on top of each other, α_4 is also zero and only the θ_5 has a value. the angle between x_4 and x_5 is -90°, and since it is a revolute joint, θ_5 considers the joint's rotation.

Question6

1) The Jacobian matrix links the robot's joint velocities to the velocity of the end-effector. It achieves this by expressing the end-effector's linear and angular velocities in terms of the joint velocities.
The basic Jacobian matrix is a $6 \times n$ matrix, where n is the number of joints in the robot. The last three rows of the matrix represent the angular velocity, while the first three rows of the matrix reflect the end-effector's linear velocity in the x, y, and z directions.

2) The elements of the basic Jacobian matrix are calculated by taking partial derivatives of the forward kinematics equations that describe the position and orientation of the end-effector as a function of the joint angles.

$$\dot{x}_P = J_{x_P}(q)\dot{q} \qquad \begin{pmatrix} \dot{x}_P \\ \dot{x}_R \end{pmatrix} = \begin{pmatrix} J_{X_P}(q) \\ J_{X_R}(q) \end{pmatrix} \dot{q}$$
$$\dot{x}_R = J_{x_R}(q)\dot{q}$$

3) Angular velocity is the rate at which a link rotates around an axis, whereas linear velocity is the speed of the link moving in a straight line. Since there is more than one link in a robot, the velocities will propagate through the links. The first link may have a slight angular or linear velocity, but the velocities of previous links affect the current velocity in the further links.

$$v_{i+1} = v_i + \omega_i \times P_{i+1} + \dot{d}_{i+1} \cdot Z_{i+1}$$

Where v_i is the linear velocity of the previous link, ω_i is the angular velocity of the previous link, P_{i+1} is the link between f_i and f_{i+1} and $\dot{d}_{i+1} \cdot Z_{i+1}$ indicates the current linear velocity if there is a prismatic joint.

$$\omega_{i+1} = \omega_i + \Omega_{i+1}$$
$$\Omega_{i+1} = \dot{\theta}_{i+1} \cdot Z_{i+1}$$

Where Ω_{i+1} is the angular velocity of the current frame and is non-zero if the joint is revolute.

Part II:
Exercise1

a) YAML launcher for Alicesim and Bobsim packages:

```
(base) arya@rosbox:~/ros2_ws/src/multi_language_publisher_subscriber/launch$ ros2 launch albob.yaml
[INFO] [launch]: All log files can be found below /home/arya/.ros/log/2023-04-20-15-41-40-888947-rosbox.pop-os-607797
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [alice_node-1]: process started with pid [607802]
[INFO] [bob_node-2]: process started with pid [607804]
[alice_node-1] [INFO] [1681992702.162352406] [alice]: I heard: "N"
[alice_node-1] [INFO] [1681992703.124267572] [alice]: I heard: "E"
[alice_node-1] [INFO] [1681992704.125289708] [alice]: I heard: "E"
[alice_node-1] [INFO] [1681992705.125080544] [alice]: I heard: "D"
[alice_node-1] [INFO] [1681992706.125381007] [alice]: I heard: "_"
[alice_node-1] [INFO] [1681992707.125990054] [alice]: I heard: "H"
[alice_node-1] [INFO] [1681992708.126505563] [alice]: I heard: "E"
[alice_node-1] [INFO] [1681992709.125675054] [alice]: I heard: "L"
[alice_node-1] [INFO] [1681992710.126093696] [alice]: I heard: "P"
[alice_node-1] [INFO] [1681992711.125362814] [alice]: I heard: "!"
[alice_node-1] [INFO] [1681992712.125344521] [alice]: END OF PM
[alice_node-1] [INFO] [1681992713.124960558] [alice]: I heard: "N"
[alice_node-1] [INFO] [1681992714.125227350] [alice]: I heard: "E"
[alice_node-1] [INFO] [1681992715.124841364] [alice]: I heard: "E"
[alice_node-1] [INFO] [1681992716.125105494] [alice]: I heard: "D"
[alice_node-1] [INFO] [1681992717.123385894] [alice]: I heard: "_"
[alice_node-1] [INFO] [1681992718.124718529] [alice]: I heard: "H"
[alice_node-1] [INFO] [1681992719.124738615] [alice]: I heard: "E"
```

Figure 6: The result of YAML launcher for Alice and Bob nodes.

b) YAML launcher for Yinsim and Yangsim packages:

```
(base) arya@rosbox:~/ros2_ws/src/yinyang/launch$ ros2 launch yinyang_launch.yaml
[INFO] [launch]: All log files can be found below /home/arya/.ros/log/2023-04-20-08-43-45-344157-rosbox.pop-os-585088
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [yinnode-1]: process started with pid [585092]
[INFO] [yangnode-2]: process started with pid [585094]
[yangnode-2] [INFO] [1681967626.954438835] [yang]: I am Yin, some mistake me for an actual material entity but I am more of a con
cept
[yinnode-1] [INFO] [1681967626.987541053] [yin]: request sent
[yangnode-2] [INFO] [1681967627.584612247] [yang]: request sent
[yinnode-1] [INFO] [1681967627.587257602] [yin]: Hi Yin, I am Yang the opposite of you.
[yangnode-2] [INFO] [1681967627.954191247] [yang]: Interesting Yang, so one could say, in a philosophical sense, we are two polar
 elements
[yinnode-1] [INFO] [1681967627.955524453] [yin]: request sent
[yangnode-2] [INFO] [1681967628.584234811] [yang]: request sent
[yinnode-1] [INFO] [1681967628.586633334] [yin]: Yes, Yin; we ourselves, do not mean anything since we are only employed to expre
ss a relation
[yangnode-2] [INFO] [1681967628.954214460] [yang]: We, Yang, are therefore the balancing powers in the universe.
[yinnode-1] [INFO] [1681967628.955747899] [yin]: request sent
[yangnode-2] [INFO] [1681967629.584487786] [yang]: request sent
[yinnode-1] [INFO] [1681967629.587075906] [yin]: Precisely, Yin; we are used to describe how things function in relation to each
other and to the universe.
[yangnode-2] [INFO] [1681967629.954900905] [yang]: Difficult and easy complete each other.
[yinnode-1] [INFO] [1681967629.956288412] [yin]: request sent
[yangnode-2] [INFO] [1681967630.584292500] [yang]: request sent
[yinnode-1] [INFO] [1681967630.586790006] [yin]: For what is and what is not beget each other.
[yangnode-2] [INFO] [1681967630.954101772] [yang]: Long and short show each other.
[yinnode-1] [INFO] [1681967630.955551445] [yin]: request sent
[yangnode-2] [INFO] [1681967631.584168428] [yang]: request sent
[yinnode-1] [INFO] [1681967631.587118222] [yin]: High and low place each other.
[yangnode-2] [INFO] [1681967631.954631588] [yang]: Noise and sound harmonize each other.
[yinnode-1] [INFO] [1681967631.955978415] [yin]: request sent
[yangnode-2] [INFO] [1681967632.584496896] [yang]: request sent
[yinnode-1] [INFO] [1681967632.587099626] [yin]: Before and behind follow each other.
[yangnode-2] [INFO] [1681967632.954177472] [yang]: You shine your light.
[yinnode-1] [INFO] [1681967632.956054450] [yin]: request sent
[yangnode-2] [INFO] [1681967633.584123927] [yang]: request sent
```

Figure 7: Result of the YAML launcher for Yin and Yang nodes.

```
[yangnode-2] [INFO] [1681967633.823083802] [yang]: 28
[yangnode-2] [INFO] [1681967633.833427104] [yang]: 27
[yangnode-2] [INFO] [1681967633.843729473] [yang]: 26
[yangnode-2] [INFO] [1681967633.854151021] [yang]: 25
[yangnode-2] [INFO] [1681967633.864475918] [yang]: 24
[yangnode-2] [INFO] [1681967633.874719588] [yang]: 23
[yangnode-2] [INFO] [1681967633.885126063] [yang]: 22
[yangnode-2] [INFO] [1681967633.895362112] [yang]: 21
[yangnode-2] [INFO] [1681967633.905959072] [yang]: 20
[yangnode-2] [INFO] [1681967633.916458676] [yang]: 19
[yangnode-2] [INFO] [1681967633.926860355] [yang]: 18
[yangnode-2] [INFO] [1681967633.937126946] [yang]: 17
[yangnode-2] [INFO] [1681967633.947363785] [yang]: 16
[yangnode-2] [INFO] [1681967633.957555690] [yang]: 15
[yangnode-2] [INFO] [1681967633.967893994] [yang]: 14
[yangnode-2] [INFO] [1681967633.978133414] [yang]: 13
[yangnode-2] [INFO] [1681967633.988808284] [yang]: 12
[yangnode-2] [INFO] [1681967633.999199841] [yang]: 11
[yangnode-2] [INFO] [1681967634.009721494] [yang]: 10
[yangnode-2] [INFO] [1681967634.020189588] [yang]: 9
[yangnode-2] [INFO] [1681967634.030541819] [yang]: 8
[yangnode-2] [INFO] [1681967634.040828360] [yang]: 7
[yangnode-2] [INFO] [1681967634.051185968] [yang]: 6
[yangnode-2] [INFO] [1681967634.061755540] [yang]: 5
[yangnode-2] [INFO] [1681967634.072365086] [yang]: 4
[yangnode-2] [INFO] [1681967634.082410337] [yang]: 3
[yangnode-2] [INFO] [1681967634.092751845] [yang]: 2
[yangnode-2] [INFO] [1681967634.103140455] [yang]: 1
[yangnode-2] [INFO] [1681967634.113455368] [yang]: 0
[yangnode-2] [INFO] [1681967634.125564202] [yang]: Result received: farewell
[yangnode-2] terminate called after throwing an instance of 'int'
[ERROR] [yangnode-2]: process has died [pid 585094, exit code -6, cmd '/home/arya/ros2_ws/install/yangsim/lib/yangsim/yangnode --
ros-args -r __node:=yang --params-file /tmp/launch_params_lydbv5qk'].
[yinnode-1] [INFO] [1681967636.953814647] [Quitting]: Done
[INFO] [yinnode-1]: process has finished cleanly [pid 585092]
(base) arya@rosbox:~/ros2_ws/src/yinyang/launch$
```

Figure 8: Result of the YAML launcher for Yin and Yang nodes.

c) Tf2 uses the idea of scene graphs, but scene graphs are designed to be iterated across, and in the case of tf2, using graphs will create transform loops. To prevent this problem, tf2 uses a tree structure which is more efficient both in the case of search computations and preventing loops.

d) static_transform_publisher is the proper way to define static transformations that can be used either as a command line tool or a node that can be added to the launch files.
If the static_transform_publisher was not available, we had to use StaticTransformBroadcaster to publish the static transforms in the code.

e) Static transformations represent the fixed relationship between coordinate frames that do not change during the operation of the robot system. For example, if there is a fixed sensor and a manipulator, the transformation between the two frames is static because both frames are fixed and do not change over time.
On the other hand, Dynamic transformations can change over time as the robot moves. For example, the transformation between a manipulator's base frame and arm frame changes over time, so it is dynamic.

f) The URDF file describing the robot's kinematics and geometry is read by the Robot State Publisher package, which then creates the static transforms needed by the TF2 library to show the relationship between the system's different coordinate frames. “Robot State Publisher is supplied with a kinematic tree model (URDF) of the robot. It then subscribes to the joint_states topic (of type sensor_msgs/msg/JointState) to get individual joint states. These joint states are used to update the kinematic tree model, and the resulting 3D poses are then published to tf2.”

Exercise2
Turtlemania

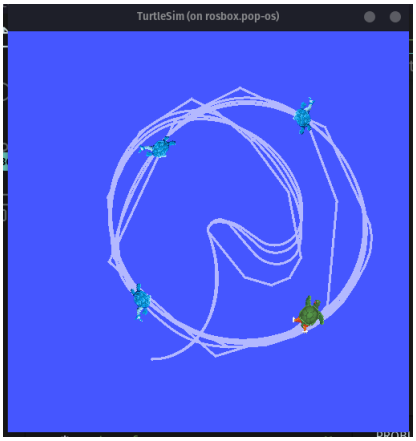

Figure 9: Turtlemania result.

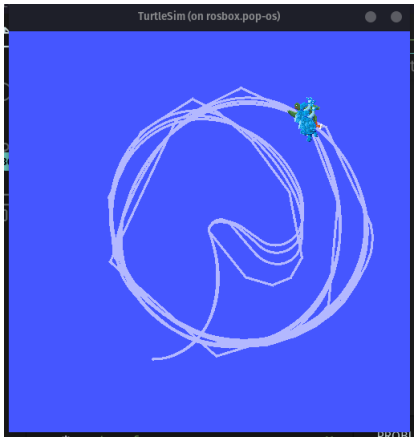

Figure 10: Turtlemania result.

```
(base) arya@rosbox:~/ros2_ws/src/turtlemania/launch$ ros2 launch turtlemania turtlemania_launch.py
[INFO] [launch]: All log files can be found below /home/arya/.ros/log/2023-04-21-15-48-10-240795-rosbox.pop-os-1756
8
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [turtlesim_node-1]: process started with pid [17569]
[INFO] [tf2_broadcaster-2]: process started with pid [17571]
[INFO] [tf2_broadcaster-3]: process started with pid [17573]
[INFO] [tf2_broadcaster-4]: process started with pid [17575]
[INFO] [tf2_broadcaster-5]: process started with pid [17577]
[INFO] [tf2_listener-6]: process started with pid [17579]
[INFO] [tf2_listener-7]: process started with pid [17581]
[INFO] [tf2_listener-8]: process started with pid [17583]
[turtlesim_node-1] Gtk-Message: 15:48:10.511: Failed to load module "appmenu-gtk-module"
[turtlesim_node-1] Gtk-Message: 15:48:10.615: Failed to load module "canberra-gtk-module"
[turtlesim_node-1] Gtk-Message: 15:48:10.617: Failed to load module "canberra-gtk-module"
[turtlesim_node-1] _IceTransSocketUNIXConnect: Cannot connect to non-local host pop-os
[turtlesim_node-1] _IceTransSocketUNIXConnect: Cannot connect to non-local host pop-os
[turtlesim_node-1] Qt: Session management error: Could not open network socket
[turtlesim_node-1] [INFO] [1682079490.790039571] [sim]: Starting turtlesim with node name /sim
[turtlesim_node-1] [INFO] [1682079490.799411582] [sim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], th
eta=[0.000000]
[turtlesim_node-1] [INFO] [1682079492.366558602] [sim]: Spawning turtle [turtle2] at x=[4.000000], y=[2.000000], th
eta=[0.000000]
[turtlesim_node-1] [INFO] [1682079492.448259930] [sim]: Spawning turtle [turtle3] at x=[4.000000], y=[2.000000], th
eta=[0.000000]
[turtlesim_node-1] [INFO] [1682079492.472654659] [sim]: Spawning turtle [turtle4] at x=[4.000000], y=[2.000000], th
eta=[0.000000]
[tf2_listener-6] [INFO] [1682079493.454526931] [listener1]: Successfully spawned turtle2
[tf2_listener-8] [INFO] [1682079493.482594845] [listener3]: Successfully spawned turtle4
[tf2_listener-7] [INFO] [1682079493.483515189] [listener2]: Successfully spawned turtle3
[tf2_listener-8] [INFO] [1682079494.438553216] [listener3]: transform not ready
```

Figure 11: Turtlemania launch terminal.

Exercise3
Installing Gazeebo

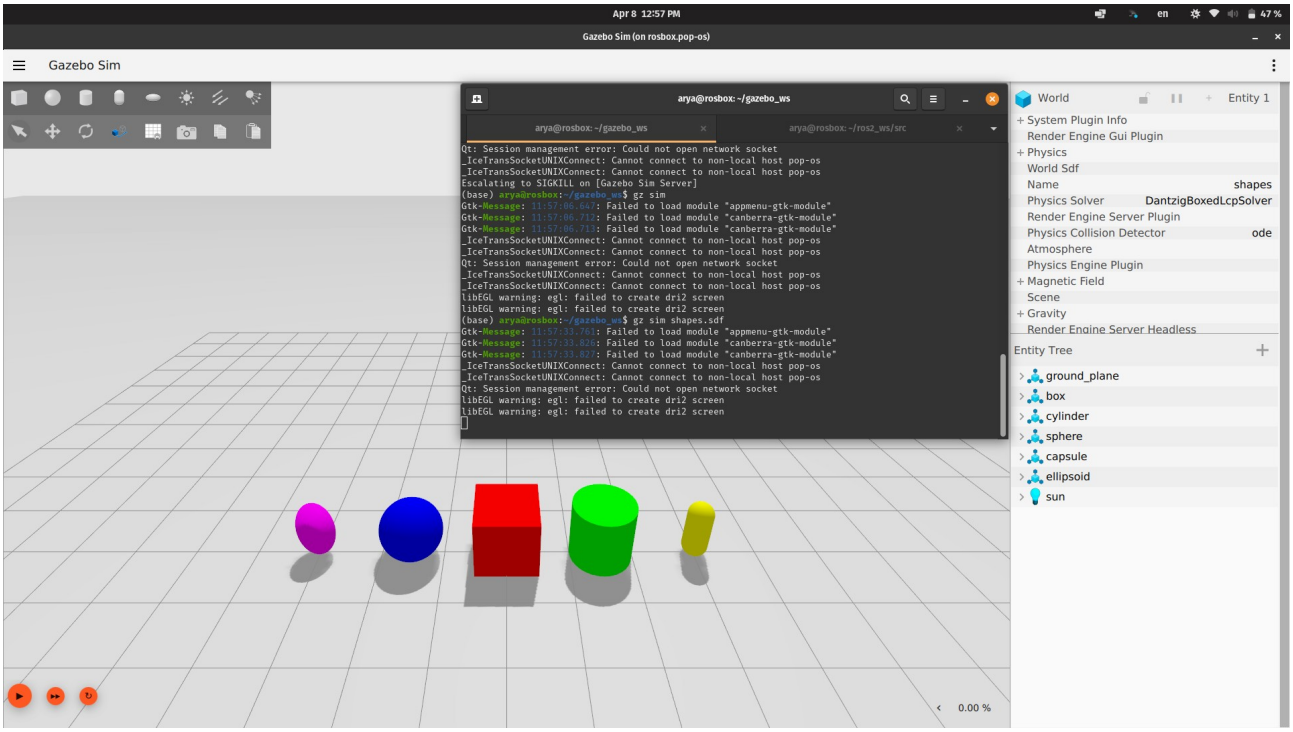

Figure 12: Shapes.sdf

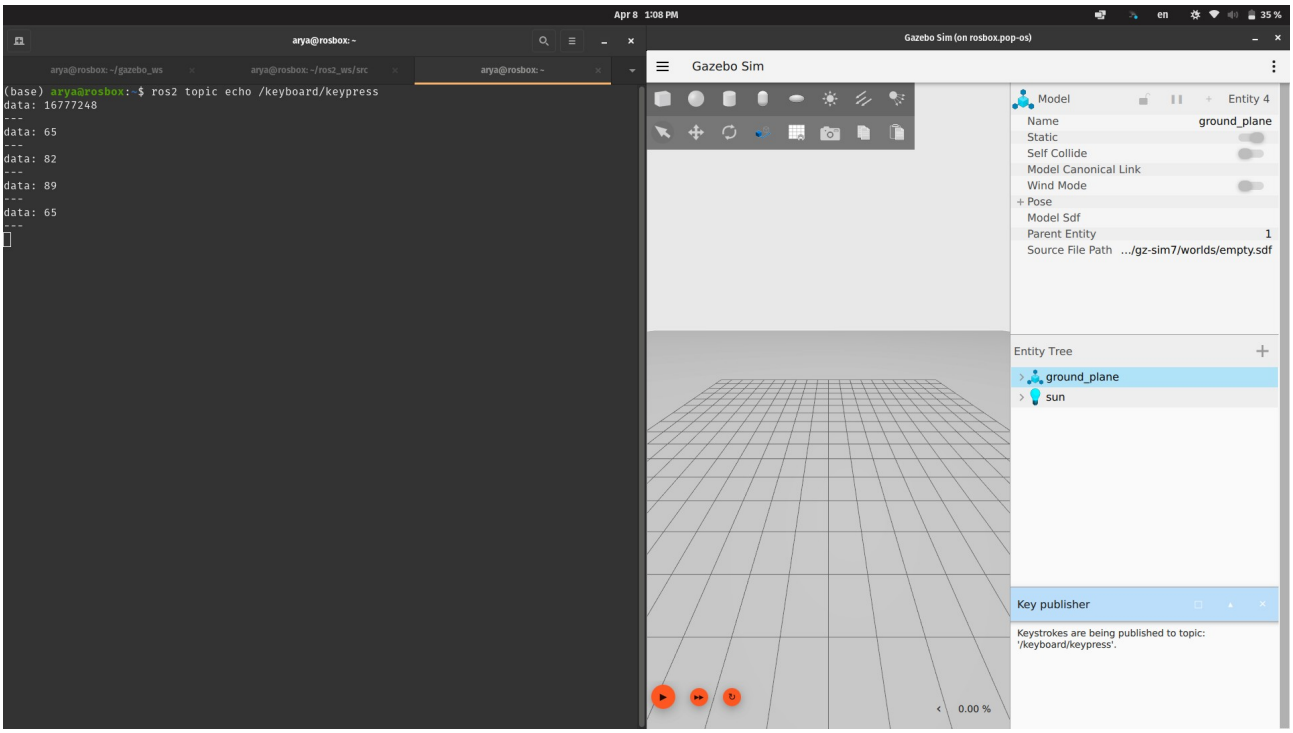

Figure 13: Typing first name ascii result.