Mobile

Function

This section teaches you to be able to have Pepper move to a location by implementing it in code. It is often used to have Pepper try to find a path and reach a location at a target point.

Remember: Pepper robots can avoid obstacles and reach the target point safely.

The basic QiSDK API call interface is as follows:

```
// Get the pin rate of the target point
Frame targetFrame = ... ;java // Create the GoTo behaviour.

// Create the GoTo behaviour
GoTo goTo = GoToBuilder.with(qiContext)
.withFrame(targetFrame)
.build(); // Create the GoTo behaviour.

// Execute the GoTo behaviour asynchronously
goTo.run();
```

Preparation

What is Frame?

Frame is a concept in the QiSDK API, which is generally used in indoor navigation of Pepper robots. Frame is used to record the position information of the target point in navigation, which generally includes the x, y, z spatial values of the robot's position.

What is Transform?

Transform is also a concept in the QiSDK API, it represents a geometric change, which includes a Vector3 and a Quaternion. at first glance, you may not understand it, but it is designed to have some knowledge of high level maths.

To better understand its use in applications, here is a brief introduction: In the geometric space of a robot, it contains x, y, z three-dimensional spatial information, as well as a rotation information, then this information is represented by a quaternion, as to why the quaternion is used to represent the rotation, please refer to here. We can draw a perfect analogy with the Offset function in Excel, which serves to define an offset, not relative to any reference.

What is Mapping?

Mapping is a local mapping service provided by Pepper.

Case

Scenario: Have Pepper move to a specified target point

Have Pepper move to a location one metre in front of you. The example is as follows:

```
Actuation actuation = qiContext.getActuation();

Frame robotFrame = actuation.robotFrame();

// Define an offset of 1 metre along the X axis.
Transform transform = TransformBuilder.create().fromXTranslation(1);

Mapping mapping = qiContext.getMapping();

FreeFrame targetFrame = mapping.makeFreeFrame();

// 0L is the timestamp, so you can specify a time to update the FreeFrame.
targetFrame.update(robotFrame, transform, 0L);

GoTo goTo = GoToBuilder.with(qiContext)
.withFrame(targetFrame.frame())
.build();

goTo.run();
```

Performance and limitations

Obstacle avoidance

Pepper robot is able to avoid obstacles while moving. It means that when it encounters an obstacle in its route to reach the target point, it will choose a suitable path to bypass the obstacle and reach the target point.

Exception handling

Specific environments can interfere with the accuracy of the robot reaching the goal and cause GoTo execution errors, such as:

- Lights can affect the sensors.
- Obstacles cannot be recognised.
- Rough ground.
- Interference of temporary obstacles, such as people or animals.

Choose GoTo or Animate?

Both GoTo and Animate allow the Pepper robot to move, but how do you choose which to use:

- Animate is suitable for unchanging orbital scenarios, excluding obstacle avoidance.
- GoTo is suitable for scenarios where the path to the goal is randomly shifted.

Reference

To learn more about using the GoTo class, please refer to:

- How to use and understand the use of GoTo
- GoTo's javadoc