Bluetooth joystick control

Scene Example:

In a stage scene, the joystick of the gamepad is used to control Pepper to move or rotate in different directions so that Pepper can ‘walk’ to the right position on the stage and start hosting or performing an entertainment programme.

Project description:

The project uses Bluetooth to provide communication between the gamepad and the Pepper robot. Before you start using it, make sure that the Bluetooth input device is connected to the Pepper's Bluetooth (turn on Bluetooth in the Pepper tablet and actively connect the input device). And, close the power hatch (charging port cover) or Pepper will not move.

You can directly clone the project to open it within AndroidStudio and run it directly on Pepper.

You can also refer to the following steps to integrate this feature into your project.

Step one:

Import the Library pepper-gamepad into your project.

Step 2 (Make sure your project is already a robot project):

Disable BasicAwareness in onRobotFocusGained and instantiate the RemoteRobotController.

```
override fun onRobotFocusGained(qiContext: QiContext) {
    val basicAwarenessHolder = HolderBuilder.with(qiContext)
        .withAutonomousAbilities(AutonomousAbilitiesType.BASIC_AWARENESS)
        .build()

    basicAwarenessHolder.async().hold().thenConsume {
        when {
            it.isSuccess -> Log.i(TAG, "BasicAwareness held with success")
            it.hasError() -> Log.e(TAG, "holdBasicAwareness error: " + it.errorMessage)
            it.isCancelled -> Log.e(TAG, "holdBasicAwareness cancelled")
        }
    }

    remoteRobotController = RemoteRobotController(qiContext)
}
```

Step three:

Implement a method to get the value back from the Bluetooth input device (the left lever enables Pepper to move in different directions and the right lever enables Pepper to rotate)。

```
private fun getCenteredAxis(event: MotionEvent, device: InputDevice, axis: Int): Float {
    val range: InputDevice.MotionRange? = device.getMotionRange(axis, event.source)

    // A joystick at rest does not always report an absolute position of
    // (0,0). Use the getFlat() method to determine the range of values bounding the joystick axis

    range?.apply {
        val value = event.getAxisValue(axis)

        // Ignore axis values that are within the 'flat' region of the joystick axis center.
        if (Math.abs(value) > flat) {
            return value
        }
    }
    return 0f
}
```

Step four:

Rewrite Activity's onGenericMotionEvent method and get the value passed back when operating the joystick through different TAGs of MotionEvent, and finally implement Pepper motion update through remoteRobotController.updateTarget.

```
override fun onGenericMotionEvent(event: MotionEvent): Boolean {
    // Add null protection for when the controller disconnects
    val inputDevice = event.device ?: return super.onGenericMotionEvent(event)

    // Get left joystick coordinates
    val leftJoystickX = getCenteredAxis(event, inputDevice, MotionEvent.AXIS_X)
    val leftJoystickY = getCenteredAxis(event, inputDevice, MotionEvent.AXIS_Y)

    // Get right joystick coordinates
    val rightJoystickX = getCenteredAxis(event, inputDevice, MotionEvent.AXIS_Z)
    val rightJoystickY = getCenteredAxis(event, inputDevice, MotionEvent.AXIS_RZ)

    if (::remoteRobotController.isInitialized) {
        thread {
            remoteRobotController.updateTarget(leftJoystickX, leftJoystickY, rightJoystickX, rightJ
        }
    }
    return true
}
```