BumperBot Simulation Testing Guide

Hands-On Tutorial: Test the Three AI Agent Packages

Author: RoboShire Team
Version: 1.0
Date: October 23, 2025
Estimated Time: 2-3 hours

Visual Robot Testing Guide

See Your Robot Move & Control It With Keyboard!

Version: 1.0 Date: October 24, 2025 Estimated Time: 30-45 minutes per package Difficulty: Fun & Interactive!

What You'll Experience

This guide shows you how to: - SEE your robot in 3D using MuJoCo physics simulation - CONTROL the robot with keyboard (W/A/S/D keys) - WATCH it move in real-time (60 FPS rendering) - MONITOR live data (velocity, position, sensors) - TEST all three AI agent packages visually

Everything visual. Everything interactive. Everything in the GUI!

Table of Contents

1. Setup & Launch

Step 1: Launch RoboShire

Windows:

```
Double-click: d:\ROS2_PROJECT\launch_roboshire.bat
```

OR:

```
cd d:\ROS2_PROJECT
python -m roboshire
```

What You'll See:

```
RoboShire v2.2.0 - Visual IDE for ROS2
Loading interface...
Main window opens with all tabs
```

Step 2: Verify MuJoCo Viewer is Available

1. Look at the left panel tabs
2. You should see: Robot Design tab
3. Click on Robot Design tab
4. You'll see the MuJoCo Viewer section

MuJoCo Viewer Shows: - Large 3D viewport (black background) - Play/Pause/Reset buttons - FPS counter - "No URDF loaded" message initially

2. Load Robot in 3D Viewer

Step 1: Import a Robot Model

Option A: Load Example Robot (Quickest)

1. Click File New from Example...
2. Select: Differential Drive Robot
3. Click Create Project

What Happens: - Example project loads - URDF file imported - MuJoCo viewer shows 3D robot!

Option B: Load BumperBot Model (For AI Agent Packages)

1. Click Robot Import URDF...
2. Navigate to: d:\ROS2_PROJECT\roboshire\examples\differential_drive\robot.urdf
3. Click Open

What Happens: - URDF loads into MuJoCo - 3D model appears in viewport - You see a two-wheeled robot!

Step 2: Interact with 3D View

Camera Controls: - Left-click + Drag Rotate camera - Right-click + Drag Pan camera - Mouse Wheel Zoom in/out - Double-click Reset camera view

Try It: 1. Left-click and drag Rotate around robot 2. Zoom in to see details 3. Zoom out to see full robot 4. Double-click to reset view

Step 3: Start Simulation

1. Click the Play button () in MuJoCo Viewer
2. Watch the FPS counter start: "60.0 FPS"
3. Robot is now in "simulation mode"

What You See: - Robot standing still (no commands yet) - Wheels are ready to move - Physics simulation active at 60 FPS

3. Build & Run (GUI Only)

Now let's make the robot controllable!

Step 1: Open Sarah's Package (Example)

1. File Open Project...
2. Navigate to: d:\ROS2_PROJECT\workspace\src\sarah_bumperbot_pkg\
3. Click Open

Status Bar Shows:

```
Project: sarah_bumperbot_pkg | Ready
```

Step 2: Build the Package

1. Build Build (or press Ctrl+B)
2. Dialog: Select sarah_bumperbot_pkg
3. Click OK

Watch Build: - Switches to Build & Deploy tab - Build Output shows progress: ``` Starting >>> sarah_bumperbot_pkg [Processing: sarah_bumperbot_pkg] Finished <<< sarah_bumperbot_pkg [10.2s]

Summary: 1 package finished [10.3s] ``` - Green checkmark appears - Status: "Build successful!"

Step 3: Run Keyboard Teleop Node

1. Run Run (or press Ctrl+R)
2. Dialog: Select package: sarah_bumperbot_pkg
3. Dialog: Select executable: keyboard_teleop
4. Click OK

What Happens: - Switches to Logs & Monitor tab - Log Viewer shows: [INFO] Keyboard Teleop started! [INFO] ========== [INFO] KEYBOARD TELEOP CONTROLS [INFO] ========== [INFO] W: Move forward [INFO] S: Move backward [INFO] A: Turn left [INFO] D: Turn right [INFO] Space: Stop [INFO] Q: Quit [INFO] ==========

Node Status Widget Shows: - keyboard_teleop (Running) - State: ACTIVE

4. Keyboard Control

The Control Keys

```
Keyboard Teleop Controls


          W
      Forward

     A    S    D
    Left  Back  Right

     SPACE = STOP
       Q = Quit
```

Step 1: Focus on Terminal (Important!)

The keyboard_teleop node runs in a terminal window that RoboShire launched. You need to:

1. Look for a terminal/console window that appeared
2. It shows: "KEYBOARD TELEOP CONTROLS"
3. Click on that terminal window to focus it
4. Now you can press keys!

If You Don't See Terminal: - Check taskbar for console window - Look behind RoboShire main window - On Windows: Black command prompt window

Step 2: Test the Controls

With terminal focused, press:

/cmd_vel publishes: linear.x = 0.2

S Robot moves backward

/cmd_vel publishes: linear.x = -0.2

A Robot turns left

/cmd_vel publishes: angular.z = 0.5

D Robot turns right

/cmd_vel publishes: angular.z = -0.5

SPACE Robot stops

1. W Robot moves forward
2. Log shows: [INFO] Moving forward
3. Log shows: [INFO] Moving backward
4. Log shows: [INFO] Turning left
5. Log shows: [INFO] Turning right
6. Log shows: [INFO] STOP
7. /cmd_vel publishes: linear.x = 0.0, angular.z = 0.0

5. Watch Robot Move

Now the exciting part - see your robot move in 3D!

Step 1: Arrange Windows

Optimal Layout:

```
RoboShire        Terminal
   (MuJoCo 3D)      (Keyboard)

   [Robot View]     W A S D
```

1. Drag RoboShire window to left half of screen
2. Drag terminal window to right half of screen
3. Keep both visible at same time

Step 2: Connect MuJoCo to ROS2

For the robot to move in MuJoCo based on ROS2 commands:

1. In RoboShire, go to MuJoCo Viewer
2. Look for "Subscribe to /cmd_vel" checkbox
3. Check it to enable ROS2 integration

OR:

If that's not available yet, you'll see movement via: - Running odometry nodes that subscribe to /cmd_vel - Wheel positions updating in joint states - TF transforms moving

Step 3: Watch It Move!

Test Forward Movement: 1. Terminal: Press and hold W 2. RoboShire 3D View: Watch robot wheels spin! 3. Log Viewer: See velocity commands streaming 4. Release W: Robot keeps moving (publish continues) 5. Press SPACE: Robot stops

Test Turning: 1. Terminal: Press A (turn left) 2. 3D View: Watch robot rotate counterclockwise 3. Try D: Robot rotates clockwise

Test Backward: 1. Terminal: Press S 2. 3D View: Wheels spin reverse 3. Robot moves backward

Step 4: Test Combinations

Drive in a Square: 1. Press W for 2 seconds (forward) 2. Press SPACE (stop) 3. Press A for 1 second (turn 90) 4. Press SPACE (stop) 5. Repeat 4 times!

Drive in a Circle: 1. Press W (forward) 2. While still pressed, also press A (turn left) 3. Robot drives in a circle! 4. Release both, press SPACE to stop

6. Monitor Live Data

While controlling the robot, monitor everything in real-time!

Topic Inspector: See Commands

1. Tools Topic Inspector
2. Double-click: /cmd_vel
3. Watch values change as you press keys:

```
# When pressing W:
linear:
  x: 0.2   Moving forward!
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0   Not turning

# When pressing A:
linear:
  x: 0.0   Not moving
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.5   Turning left!
```

Launch Odometry for Position Tracking

To see WHERE the robot has moved:

1. Run Run (Ctrl+R)
2. Select: sarah_bumperbot_pkg
3. Select: odometry_calculator
4. Click OK

Now Monitor Odometry: 1. Tools Topic Inspector 2. Double-click: /odom 3. Watch position change as robot moves:

```
pose:
  pose:
    position:
      x: 0.523   Moving forward!
      y: 0.087   Drifting slightly
      z: 0.0
    orientation:
      z: 0.123   Rotated 14 from start
      w: 0.992
```

TF Tree: See Frame Transforms

1. Tools TF Tree Visualizer
2. Watch transforms update in real-time:

```
odom
  base_link (moving!)
       left_wheel (spinning!)
       right_wheel (spinning!)
       imu_link (following base)
```

Performance: Monitor CPU/Memory

1. Click Performance Profiler tab
2. Watch metrics while driving:

```
Node: keyboard_teleop
CPU: 0.8%   Low overhead
Memory: 8 MB
Publish Rate: 10 Hz

Node: odometry_calculator
CPU: 2.1%   Computing position
Memory: 12 MB
Update Rate: 50 Hz
```

7. Test All Three Packages

Now test each AI agent's package and compare!

Test 1: Sarah's Package (Beginner)

Already done above! Summary:

Feature	Result
Build Time	_____ seconds
Keyboard Control	W/A/S/D working
Robot Movement	Visible in 3D

Responsiveness	Immediate
CPU Usage	_____ %
Fun Factor	_____ / 10

Your Notes: - How smooth was movement? __ - Any lag or delay? __ - Easy to control? ___

Test 2: Marcus's Package (Optimized)

Step 1: Stop Sarah's Nodes 1. Run Stop All Nodes 2. Wait 5 seconds for clean shutdown

Step 2: Load Marcus's Package 1. File Open Project... 2. Navigate: workspace/src/marcus_bumperbot_pkg/ 3. Click Open

Step 3: Build Marcus's Package 1. Build Build (Ctrl+B) 2. Select: marcus_bumperbot_pkg 3. Watch build complete (~8-12 seconds)

Step 4: Run Marcus's Keyboard Teleop 1. Run Run (Ctrl+R) 2. Select: marcus_bumperbot_pkg 3. Select: keyboard_teleop

Step 5: Control & Compare 1. Focus terminal window 2. Press W / A / S / D 3. Watch robot move in 3D

Marcus's Differences: - Optimized code Should feel slightly faster - Unified TF broadcaster Lower CPU usage - Performance monitoring See metrics in GUI

Fill Out:

Feature	Marcus's Result
Build Time	_____ seconds (faster than Sarah's?)
Keyboard Control	W/A/S/D working
Robot Movement	Visible in 3D
Responsiveness	Better / Same / Worse
CPU Usage	_____ % (lower than Sarah's?)
Fun Factor	_____ / 10

Comparison: - Is movement smoother? __ - Notice any performance improvements? __ - Preferred control feel: Sarah's / Marcus's

Test 3: Elena's Package (Enterprise)

Step 1: Stop Marcus's Nodes 1. Run Stop All Nodes 2. Wait 5 seconds

Step 2: Load Elena's Package 1. File Open Project... 2. Navigate: workspace/src/elena_bumperbot_pkg/ 3. Click Open

Step 3: Build Elena's Package 1. Build Build (Ctrl+B) 2. Select: elena_bumperbot_pkg 3. Watch build (~15-20 seconds - has tests!)

Step 4: Run Elena's Velocity Controller (Lifecycle)

Elena uses lifecycle nodes - different process!

1. Run Run (Ctrl+R)
2. Select: elena_bumperbot_pkg
3. Select: velocity_controller (lifecycle version)

Step 5: Activate the Lifecycle Node 1. Go to Build & Deploy tab 2. Click Lifecycle Manager sub-tab 3. See node: velocity_controller (State: UNCONFIGURED) 4. Click Configure button State: INACTIVE 5. Click Activate button State: ACTIVE 6. Now it's ready!

Step 6: Run Keyboard Teleop 1. Run Run (Ctrl+R) 2. Select: elena_bumperbot_pkg 3. Select: keyboard_teleop (if available) - OR use Sarah's/Marcus's keyboard_teleop (compatible!)

Step 7: Control & Observe Safety Features

Elena's package has safety limits!

Test Velocity Limiting: 1. Try to drive forward (press W) 2. Check Log Viewer 3. You might see: [WARN] [safety_monitor]: Velocity limited to max: 0.22 m/s 4. Robot moves but at safe speed (not full speed)

Test Timeout Detection: 1. Press W to move 2. Release W and wait 2 seconds 3. Log shows: [WARN] [safety_monitor]: cmd_vel timeout detected [INFO] [velocity_controller]: Emergency stop triggered 4. Robot automatically stops! (Safety feature)

Fill Out:

Feature	Elena's Result
Build Time	_____ seconds (longest?)
Lifecycle Setup	Easy / Medium / Hard
Keyboard Control	W/A/S/D working
Robot Movement	Visible in 3D
Safety Features	Velocity limiting / Timeout
CPU Usage	_____ %
Enterprise Feel	_____ / 10

Elena's Special Features You Noticed: - Safety limits active? __ - Timeout detection working? __ - More controlled movement? ___

8. Side-by-Side Comparison

Movement Quality

Package	Smoothness	Responsiveness	Control Feel
Sarah's	_____/10	_____/10	_____/10
Marcus's	_____/10	_____/10	_____/10

Elena's	_____/10	_____/10	_____/10

Visual Experience

Package	3D Rendering	FPS Stable?	Lag/Jitter?
Sarah's	Good/Bad	Yes/No	Yes/No
Marcus's	Good/Bad	Yes/No	Yes/No
Elena's	Good/Bad	Yes/No	Yes/No

Ease of Use

Package	Build Easy?	Run Easy?	Control Easy?	Overall
Sarah's	Yes/No	Yes/No	Yes/No	_____/10
Marcus's	Yes/No	Yes/No	Yes/No	_____/10
Elena's	Yes/No	No (lifecycle)	Yes/No	_____/10

Your Favorite?

Most Fun to Drive:___

Why?_____

Best for Learning:___

Best for Real Robot:___

9. Advanced Testing

Test A: Drive Patterns

Figure-8 Pattern: 1. Press W (forward) 2. Alternate pressing A and D (turn left/right) 3. Create figure-8 shape 4. Watch in 3D view!

Record: - Easiest package to control: __ - Smoothest figure-8: __

Test B: Sensor Fusion (All Packages)

Add IMU and EKF:

1. Run Run (Ctrl+R)
2. Launch: imu_reader
3. Launch: ekf_localization (or sensor_fusion)

Watch in Topic Inspector: - /odom (raw) - jumpy values - /odom/filtered (EKF) - smooth values - /imu/data - orientation data

Drive the robot and compare: - Raw odometry drift: High / Medium / Low - Filtered odometry accuracy: Better / Same / Worse

Test C: Multi-Robot (Marcus's Package)

If you want to control TWO robots:

1. Launch Marcus's package with namespace:
2. Edit launch arguments: namespace:=robot1
3. Launch again: namespace:=robot2
4. Control each independently!

Topics become: - /robot1/cmd_vel - /robot2/cmd_vel

Fun Challenge: Control both robots simultaneously!

10. Troubleshooting

Problem: Robot Not Moving in 3D

Check: 1. Is simulation playing? (button pressed) 2. Is MuJoCo subscribed to /cmd_vel? 3. Are keyboard commands publishing? (check Topic Inspector)

Solution: - Click Play button in MuJoCo Viewer - Verify nodes are running in Node Status - Check /cmd_vel topic has data

Problem: Can't Control with Keyboard

Check: 1. Is terminal window focused? (click on it!) 2. Is keyboard_teleop node running? 3. Are keys showing in logs?

Solution: - Click on the terminal/console window - Verify keyboard_teleop in Node Status (should be green) - Check Log Viewer for keypress messages

Problem: Robot Moving But Not Visible

Check: 1. Is URDF loaded in MuJoCo Viewer? 2. Is camera zoomed out too far? 3. Is robot model valid?

Solution: - Import URDF: Robot Import URDF - Double-click viewport to reset camera - Zoom in with mouse wheel

Problem: Lag or Jitter

Check: 1. FPS counter in MuJoCo Viewer 2. CPU usage in Performance Profiler 3. Too many nodes running?

Solution: - Stop unnecessary nodes - Close other applications - Lower simulation quality in settings

Problem: Elena's Lifecycle Not Working

Check: 1. Is node in ACTIVE state? 2. Did you Configure then Activate? 3. Check Lifecycle Manager tab

Solution: - Open Lifecycle Manager tab - Select velocity_controller - Click Configure, then Activate - Check for errors in Log Viewer

11. Success Checklist

For Each Package

- [ ] Built successfully in GUI
- [ ] URDF loaded in MuJoCo 3D viewer
- [ ] Simulation running at 60 FPS
- [ ] Keyboard teleop node running
- [ ] W key moves robot forward (visible!)
- [ ] A key turns robot left (visible!)
- [ ] D key turns robot right (visible!)
- [ ] S key moves robot backward (visible!)
- [ ] SPACE stops robot
- [ ] Topics visible in Topic Inspector
- [ ] /cmd_vel showing velocity commands

- [ ] /odom showing position changes
- [ ] TF tree showing transforms
- [ ] Performance metrics displayed
- [ ] Comparison table filled out

Overall Experience

- [ ] All three packages tested visually
- [ ] Drove robot in square pattern
- [ ] Drove robot in circle pattern
- [ ] Tested sensor fusion (EKF)
- [ ] Monitored live data in GUI
- [ ] Compared performance metrics
- [ ] Identified favorite package
- [ ] Had fun!

12. Certificate of Visual Testing

RoboShire Visual Testing Certificate

I, ___, have successfully controlled and visualized all three AI agent robots using RoboShire's 3D viewer and keyboard teleoperation on ____ (date).

Visual Testing Completed: - [x] Sarah's Robot (Fun Rating: _ /10) - [x] Marcus's Robot (Fun Rating: /10) - [x] Elena's Robot (Fun Rating: __ /10)

Skills Demonstrated: - [x] Building packages via GUI - [x] Loading robots in 3D viewer - [x] Keyboard teleoperation (W/A/S/D) - [x] Watching real-time movement - [x] Monitoring live sensor data - [x] Comparing package performance

Favorite Pattern Driven:___

Most Responsive Package:___

Best Visual Experience:___

Signature:_____

Summary

You just learned how to: - Build ROS2 packages using RoboShire GUI - Load robots in 3D using MuJoCo viewer - Control robots with keyboard (W/A/S/D keys) - Watch them move in real-time at 60 FPS - Monitor live data (velocity, position, sensors) - Compare three AI packages visually - Test safety features (Elena's package) - Have fun!

No terminal commands. No complex setup. Just visual, interactive robotics!

Ready to control your robots? Launch RoboShire and start testing!

Version 1.0 | October 24, 2025 | RoboShire Team