TANGIBLE INTERFACES FOR VR APPLICATIONS

2025 HELHA BIP PROJECT

READING IMU SENSOR DATA

Data Read: Each loop in the code retrieves new acceleration values continuously (x, y, z).

Smoothing: The program places these values into circular buffers, averaging them to reduce noise.

Calculation: Pitch and roll are derived from the filtered (x, y, z) values for further use.

CSV Formatting: The code combines the values to into a single string

BALL MAZE PROJECT IDEA

Creating a 3D maze game playable on VR headset

Use ESP32 with ADXL345 IMU sensor to make a controller

Use Bluetooth to connect the controller with the headset

Enhance our game with additional components

BLUETOOTH COMMUNICATION

INITIALIZATION: THE CODE CREATES A **BLE CUSTOM GAMEPAD** DEVICE, SETS UP A CUSTOM HID DESCRIPTOR AND STARTS BLUETOOTH ADVERTISING.

DATA MAPPING: PITCH AND ROLL VALUES FROM THE ADXL345 ARE **MAPPED** TO HID GAMEPAD AXES

REPORT SENDING: CALLING BLEGAMEPAD.SENDREPORT() FUNCTION TRANSMITS THESE VALUES OVER BLUETOOTH.

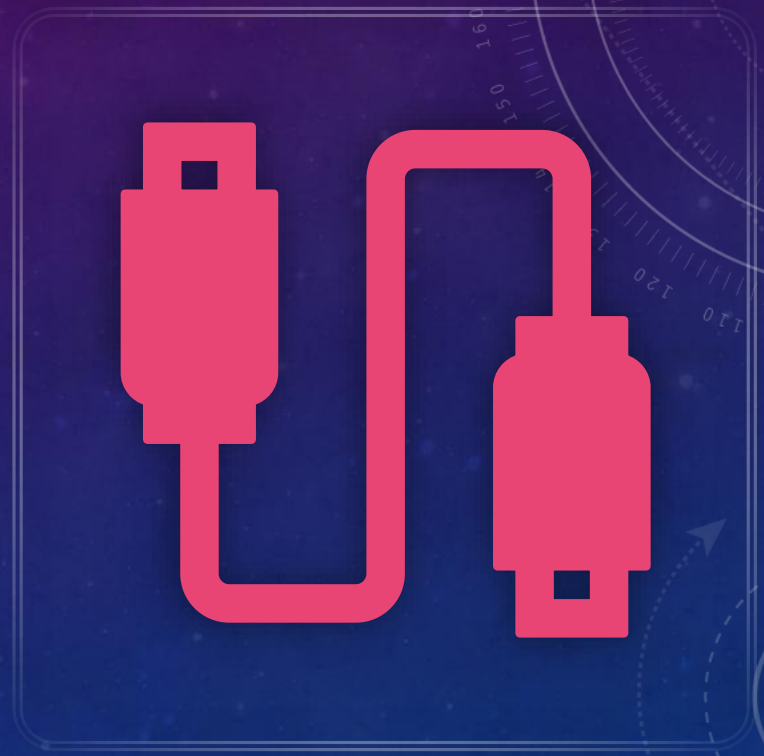
ROS2 USB COMMUNICATION

Serial Read: The Python node opens the USB serial port and reads each new line of data

Parsing: Each line is split into X, Y, Z, roll, and pitch values.

ROS2 Message Creation: The code wraps these values in a ROS 2 AccelStamped message

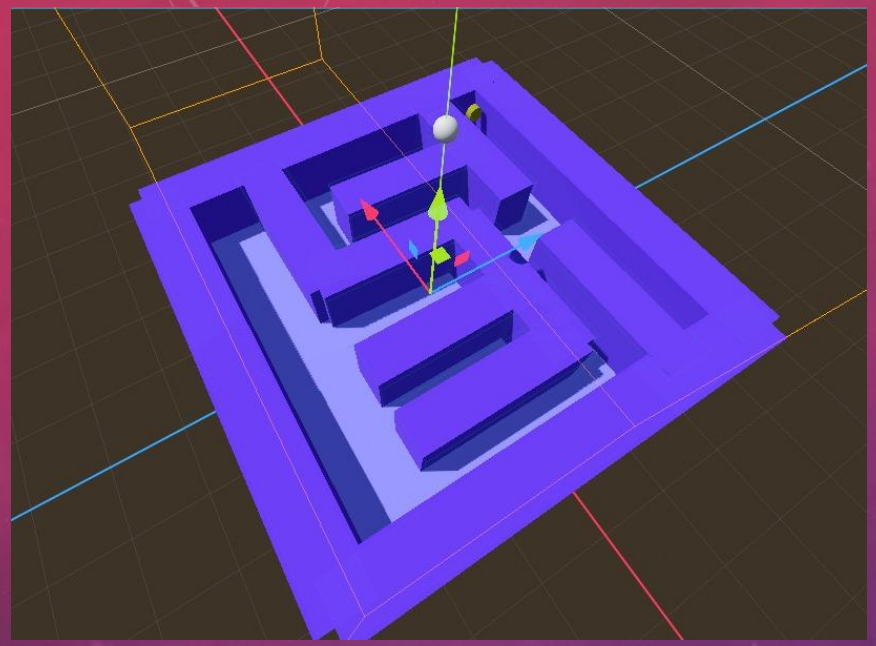
Topic Publishing: It publishes to the topic so other ROS 2 nodes can use that sensor information.

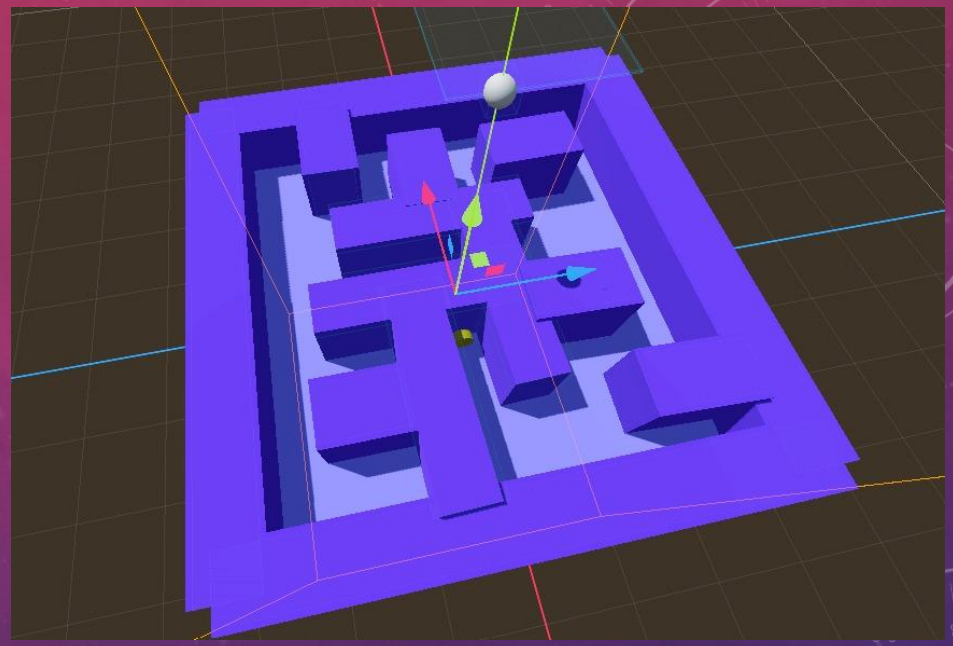


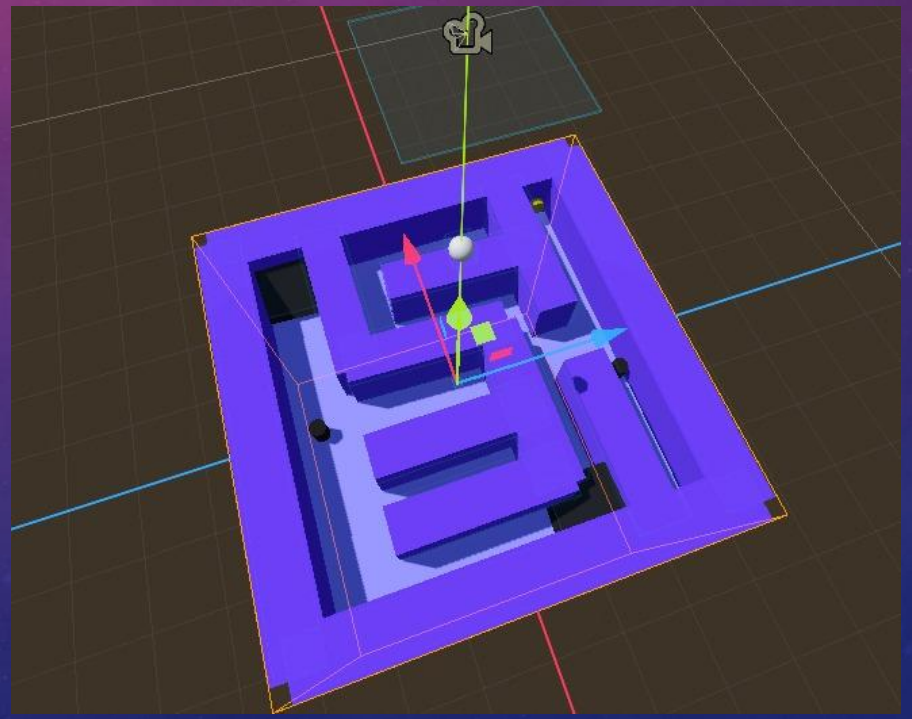
GODOT PART

- **VR Initialization**: In _ready(), the script enables the OpenXR interface, letting the game run in VR mode.
- **Controller Input:** Each frame reads pitch and roll from the BLE gamepad axes
- **Tilting the Maze:** The script scales and smooths those axis values, then applies them to the maze floor’s rotation, causing the ball to roll.
- **Level Progression:** Functions switch scenes when the ball falls or collects a coin, providing multiple stages in the VR maze game.

MAZES







THANK YOU!

RAFAEL LUIS

LLUÍS FRANCESC COLLELL ERRA

VIVIAN PINA

BARTOSZ PIOTROWSKI