Initial Concept

Our initial idea was to use the MPU6050 sensor to map mouse X and Y movement to screen coordinates, then create physical buttons to allow the caterpillar to jump and move. The Caterpillar Quest game controls a caterpillar that moves toward the player's cursor when they press the left mouse button, with the right mouse button used to jump.

Development Process

We started by having the Arduino perform calculations on velocity to create a 2D vector, then move the mouse's relative position by that vector on the host machine using a Python script with PyAutoGUI. Due to this approach, we ran into issues with the mouse movement being *very* choppy.

We attempted several fixes: changing the Arduino's baud rate, smoothing the vectors in the Python script, and offloading calculations to the host machine so all the Arduino does is send comma-separated accel X, Y, Z values and button flags. None of these worked.

We thought to try a different Python library to move the mouse—switching from PyAutoGUI to pynput. This worked, luckily. After fixing this and testing it, we realized that since the MPU was measuring forces instead of measuring distance or position, controlling it would probably never be intuitive in this velocity-based way.

We decided to pivot to a joystick simulation style. Instead of tracking cumulative movement, the cursor's position maps directly to the sensor's tilt angle—holding the sensor level keeps the cursor at screen center, and tilting moves it proportionally away from center. This worked much better and surprisingly was very intuitive when paired with our game.

Hardware

Components: Arduino Uno R3, MPU6050 sensor, 3 pennies (for buttons), glove, wire and solder.

Wiring: The MPU6050 connects via I2C (SDA to A4, SCL to A5, plus power and ground). The penny buttons connect to pins 2 and 3 with internal pullup resistors enabled.

For the hardware, we initially imagined strapping the Arduino to the palm of the user's hand and making small buttons to strap onto the fingertips, but we were unable to find an elastic band so instead built it into a glove. This worked well.

We decided to print a finger receptacle that holds a momentary button with holes for wiring. However, group member Jack was too impatient for hot glue to arrive, so he super glued all 2

buttons in 2 attempts—both resulting in the super glue getting into the button casing and insulating the contacts, effectively breaking the buttons.

We pivoted to using pennies. We drilled holes in the pennies: 1 hole to solder wire into and 3 holes to sew the penny into the glove. While this wasn't as elegant as 3D printed finger buttons, it worked reasonably well. We created 3 penny buttons: 2 connecting to data pins on fingers and 1 ground on the thumb. Touching a finger penny to the thumb penny completes the circuit.

Software

The Arduino code is minimal—it just reads the MPU6050 acceleration values and button states, then transmits CSV data at 230400 baud: accelX,accelY,accelZ,btn1,btn2. All processing happens on the host computer.

The Python script handles serial communication, calibration, and mouse control using pynput. Incorporating the buttons was easy as we implemented the button registration in Arduino and sent them as 2 new boolean flags.

The processing pipeline: auto-detect Arduino, calibrate for 500 samples (~2-3 seconds) to learn neutral position, calculate tilt relative to calibration, apply deadzone filtering to prevent drift, scale tilt to pixel offset, clamp to maximum range, apply smoothing, and update cursor position.

Usage

1. Connect Arduino via USB
2. Run python mc_v2.py
3. Keep controller level during calibration
4. Tilt hand to move cursor, touch finger pennies to thumb for clicks

Conclusion

Through iterative development, we learned that the choice of joystick-style vs. velocity-based was more important than attempting technical optimizations. The final implementation provides intuitive control that pairs well with Caterpillar Quest. The hardware development taught us that patience with proper adhesives is worth the wait, and that creative solutions like penny buttons can work better than over-engineered ones.