Neural Circuit Documentation for Hexapod

Overview

This directory contains C# scripts that model the neural circuit (spiking neurons) of a hexapod robot. The system is organized into several components that process external commands, generate rhythmic neural activity, and translate this activity into leg movements.

Components

1. `HexapodState.cs`

Holds the state variables for the hexapod, including neural and motor variables.

- **Neural State:**
  - `float[] CPGs = new float[10];`
    Central Pattern Generator (CPG) state variables, encoding oscillatory neural activity.
- **Motor State:**
  - `Q1`, `Q2`, `Q3`: Joint angles for each of the 6 legs.
  - `E`, `LP`, `L2P`, `L3P`: Additional state variables for each leg (position, error, etc.).
- **Directional and Movement State:**
  - `DIR1`, `DIR2`, `DIR3`, `DIR4`: Directional signals derived from stimuli.
  - `FW`, `BW`, `TL`, `TR`, `L`, `R`, `MOV`: Forward, backward, turn left, turn right, left, right, and movement signals.

---

2. `Stimuli.cs`

Processes external commands (inputs) and updates the internal state of the hexapod.

- **Inputs:**
  - `go`, `bk`, `spinL`, `spinR`, `left`, `right`: External control signals (e.g., from user or higher-level controller).
  - `dt`: Time step.
- **Processing:**
  - Uses a Naka-Rushton function to normalize and process the inputs.
  - Updates the state variables in `HexapodState` (e.g., `FW`, `BW`, `TL`, `TR`, `L`, `R`, `MOV`, `DIR1`, `DIR2`, `DIR3`, `DIR4`).
- **Outputs:**
  - Updated fields in `HexapodState`.

---

3. `CPG.cs`

Implements the Central Pattern Generator, which produces rhythmic neural activity for locomotion.

- **Inputs:**
  - `float[] CPGs`: The current state of the CPGs (from `HexapodState`).

  - `dt`: Time step.
- **Processing:**
  - Updates the CPG state variables using a set of coupled differential equations.
  - The equations use parameters (e.g., `Ao`, `Bo`, `Co`, `Do`, etc.) to control the oscillatory dynamics.
- **Outputs:**
  - Updated `CPGs` array, encoding the neural oscillations for locomotion.

---

4. `Locomotion.cs`

Translates neural activity (from the CPG) into joint commands for the hexapod's legs.

- **Inputs:**
  - References to joint angles and state variables (`Q1`, `Q2`, `Q3`, `E`, `LP`, `L2P`, `L3P`).
  - `T`: Target position or timing signal.
  - `CPGXY`, `CPGZ`: Neural signals from the CPG.
  - `dt`: Time step.
- **Processing:**
  - Runs an internal loop to update joint angles and positions based on the CPG signals and current state.
- **Outputs:**
  - Updated joint angles and state variables (by reference).

---

5. `HexapodSimulation.cs`

Demonstrates how the above components interact in a simulation loop.

- **Inputs:**
  - Initial state setup.
  - External commands to `Stimuli.Update` (e.g., `go: 1` for forward movement).
- **Processing:**
  - Updates stimuli, CPG, and locomotion in each simulation step.
  - Applies CPG outputs to locomotion for each leg.
- **Outputs:**
  - Logs and updated state variables, which could be used to drive a simulated or real robot.

Inputs and Outputs Summary

Inputs

- External control signals: `go`, `bk`, `spinL`, `spinR`, `left`, `right`
- Time step: `dt`
- Current state variables (from `HexapodState`)

Outputs

- Updated neural state: `CPGs`
- Updated movement state: `Q1`, `Q2`, `Q3` (joint angles), `E`, `LP`, `L2P`, `L3P`

- Directional and movement signals: `DIR1`, `DIR2`, `DIR3`, `DIR4`, `FW`, `BW`, `TL`, `TR`, `L`, `R`, `MOV`

Data Flow Diagram

```
[External Inputs]
    ↓
Stimuli.Update
    ↓
[HexapodState] ↔ CPG.Update
    ↓
Locomotion.Update
    ↓
[Joint Angles & Positions]
```

For further details, see the inline comments and code in each file.