NEURONSCAPE SYSTEM SPECIFICATION

VERSION 0.01 DRAFT

Julian Bailey

School of Electronics & Computer Science

University of Southampton, UK

[jab@ecs.soton.ac.uk](mailto:jab@ecs.soton.ac.uk)

17th June 2010

Table of Contents

[1 Definitions & Abbreviations 4](#_Toc274067809)

[3 Neuronscape System Structure 5](#_Toc274067810)

[3.1 Environment Server 6](#_Toc274067811)

[3.2 Neuron Environment Interface 6](#_Toc274067812)

[3.2.1 Sensory Input 6](#_Toc274067813)

[3.2.2 Muscle Output 7](#_Toc274067814)

[3.3 Activity Viewer 7](#_Toc274067815)

[4 Network Communication 8](#_Toc274067816)

[4.1 Packet Structure 8](#_Toc274067817)

[4.2 Packet Types 8](#_Toc274067818)

[4.3 Packet Types Detail 9](#_Toc274067819)

[4.3.1 Acknowledge (ACK) 9](#_Toc274067820)

[4.3.2 Negative Acknowledge (NAK) 9](#_Toc274067821)

[4.3.3 Error (ERR) 9](#_Toc274067822)

[4.3.4 Connection Request 9](#_Toc274067823)

[4.3.5 Disconnection Request 10](#_Toc274067824)

[4.3.6 Force Disconnection 10](#_Toc274067825)

[4.3.7 Client Enumerate 10](#_Toc274067826)

[4.3.8 Req. Add Object 10](#_Toc274067827)

[4.3.9 Ack. Add Object 10](#_Toc274067828)

[4.3.10 Update Object Position 11](#_Toc274067829)

[4.3.11 Bulk Update Objects 11](#_Toc274067830)

[4.3.12 Delete Object 11](#_Toc274067831)

[4.3.13 Tests – Echo Reply 12](#_Toc274067832)

[4.3.14 Tests – Echo 12](#_Toc274067833)

[5 Ack/Nak Codes 13](#_Toc274067834)

[6 Error Codes 13](#_Toc274067835)

# Definitions & Abbreviations

# Neuronscape System Structure

The Neuronscape system is made up of three different parts, each performing different function.



Figure .1: An overall view of the neuronscape system

The overall system view is shown in Figure 2.1, the Environment server (ES) is responsible for tracking all the objects in the environment, handling tasks such as environmental physics. The Neuron Environment interface (NEI) is responsible for muscle and receptor modelling, in other words, it is responsible for interfacing between the neuron network simulation and the virtual physical environment. Finally the activity viewer (AV) allows a person to peek into the virtual world, altering parameters and allowing external interactions.

At a minimum a system requires the Server module and the Neuron Environment interface module therefore these are considered “core components” of the system.

The purpose of dividing these components is to allow processing on separate machines, with many virtual animals in the environment many calculations may have to be performed in order to model, environment physics, reception behaviour and muscle behaviour. By dividing the load less pressure is put on one processing device so processing can be divided easily between machines that are part of a processing cluster.

The next three sections describe the operation of each of the components in greater detail.

## Environment Server

The environment server holds the *master list* of all the objects in the system, an object can be an animal controlled by an NEI or an inanimate object such as food any other environmental “furniture”.

The absolute position of the object is tracked by the server as sets of Cartesian coordinates along with the orientation of the object in radians and the velocity of the object. The magnitude and direction of any muscle force and external force acting on the object is also held since it is used to calculate future position and velocity.

The physics in the environment is modelled according to classical mechanics; as such the environment obeys the conservation of momentum in a closed system and Newton’s laws of motion.

Time progresses in the environment through *ticks*, the period of which can be defined by the user. Each *tick* the forces acting on each object are evaluated and the position and velocity of each object can change. Collisions between objects are also calculated during the *tick*.

## Neuron Environment Interface

The Neuron Environment Interface provides a method by which a network of neurons can interact with the environment, providing sensory input and output via forces generated by muscles. It “owns” an object that exists in the Environment server, which means it is responsible for generating the forces which will move the object and reading the sensory data from the environment, processing it and passing it to the neuronal network simulation.

The NEI holds a local copy of the various objects in the environment, this is important for rendering the vision sensory input.

The NEI component has an internal *tick* system like the server which allows complex muscle and receptor models to be used.

### Sensory Input

Initially sensory input will be offered through sight and touch. For sight the objects in front of the animal will be rendered and a bitmap of what the animal can see will be generated allowing this visual plane to be used to drive light receptor models which in turn will provide input to the neuronal network.

The default size of the viewing area will be 320 x 240 pixels; this can be translated on to a smaller number of light receptors if a 1:1 pixel receptor mapping is not required. Stereoscopic vision is possible by rendering two views of the environment offset by a user defined distance.

Sensory input such as touch so the neuronal network can know if a collision with an object has occurred will also be present as required.

### Muscle Output

Movement is achieved by the action of muscles; the force vector derived from the muscle action is fed into the environment which calculates the resultant motion.

Initially if several muscles are acting at the same time the NEI must calculate the resultant vector and feed it to the environment.

## Activity Viewer

The activity viewer is the way that the user of the system can interact with the environment, seeing the environment through the eyes of an animal or a god type view showing the positions of all the objects. Like the NEI the viewer holds a local copy of the list of objects which is updated by the server when changes occur.

The view is rendered locally on the machine the viewer program is run on so causes no significant load on the server.

The end goal is to allow the user to push and touch objects in the system and watch how they respond or change the physical environment.

# Environment

## Coordinate Systems

Table ‑: Coordinate Systems Used in Neuronscape

|  |  |
| --- | --- |
|  | Coordinate System |
| Positional | 3D Cartesian (x, y, z) |
| Force | Spherical (Magnitude, Theta, Phi) |
| Velocity | Spherical (Magnitude, Theta, Phi) |

## Environmental Physics

# Server Database Structure

## Clients Table

|  |  |  |
| --- | --- | --- |
| Column Name | SQL Data Type | Notes |
| id | SQL\_INT | Primary Key  Auto Increment |
| ipv4\_addr | SQL\_VARCHAR[15] |  |
| port | SQL\_INT |  |
| role | SQL\_INT |  |
| command\_set\_major | SQL\_INT |  |
| command\_set\_minor | SQL\_INT |  |
| status | SQL\_INT |  |

## Objects Table

|  |  |  |
| --- | --- | --- |
| Column Name | SQL Data Type | Notes |
| id | SQL\_INT | Primary Key  Auto Increment |
| client\_id | SQL\_INT |  |
| x | SQL\_DOUBLE |  |
| y | SQL\_DOUBLE |  |
| z | SQL\_DOUBLE |  |
| theta | SQL\_DOUBLE |  |
| phi | SQL\_DOUBLE |  |
| motorforce | SQL\_DOUBLE |  |
| motorforce\_theta | SQL\_DOUBLE |  |
| motorforce\_phi | SQL\_DOUBLE |  |
| xternforce | SQL\_DOUBLE |  |
| xternforce\_theta | SQL\_DOUBLE |  |
| xternforce\_phi | SQL\_DOUBLE |  |
| velocity | SQL\_DOUBLE |  |
| velocity\_theta | SQL\_DOUBLE |  |
| velocity\_phi | SQL\_DOUBLE |  |

# Network Communication

Communication in the neuronscape system is achieved using the TCP/IP protocol, and as such, the communications protocol sits in the application layer in the TCP/IP reference model.

## Packet Structure

The basic packet structure (shown in Table 2‑1) consists of a 2 byte packet ID, a 2 Byte Length and the payload. The value specified by *Length* is the length of only the payload of the packet, this means all packets should be a minimum of 4 bytes long if they do not include a payload. This does not include the overhead of the TCP/IP protocol.

All multi-byte values are transmitted in Little-endian format.

Table ‑: Generic Packet Format

|  |  |  |  |
| --- | --- | --- | --- |
| Description | Packet Length | Packet ID | Payload |
| Field Length | 2 Bytes | 2 Bytes | =< 1020 Bytes |
| Value | =< 1022 | See Table 2‑2 | Dependent on Packet Type |

## Packet Types

Table ‑: Packet Types

|  |  |  |  |
| --- | --- | --- | --- |
| Packet ID | Name | Section | Page |
| 0 (0x0000) | Acknowledge (ACK) | 4.3.1 |  |
| 1 (0x0001) | Error (ERR) | 4.3.2 |  |
| 2 (0x0002) | Connection Request | 4.3.3 |  |
| 3 (0x0003) | Disconnection Request | 4.3.4 |  |
| 4 (0x0004) | Force Disconnection | 4.3.5 |  |
| 5 (0x0005) | Client Enumerate | 4.3.6 |  |
| 6 (0x0006) | Req. Add Object | 4.3.7 |  |
| 7 (0x0007) | Update Object Position | 4.3.8 |  |
| 8 (0x0008) | Bulk Update Objects | 4.3.9 |  |
| 9 (0x0009) | Delete Object | 4.3.10 |  |
| 10 (0x000A) | Object Forces Packet | 4.3.11 |  |
| 11 (0x000B) | Client State Update |  |  |
| 65534 (0xFFFE) | Tests - Echo Reply | 4.3.12 |  |
| 65535 (0xFFFF) | Tests - Echo | 4.3.13 |  |

## Packet Types Detail

### Acknowledge (ACK)

Table ‑: Acknowledge Packet Format

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Packet Length | Packet ID | Data Word 1 | Data Word 2 |
| Field Length | 2 Bytes | 2 Bytes | 2 Bytes | 2 Bytes |
| Value | 0x0002 (2) | 0x0000 (0) |  |  |

### Error (ERR)

Table ‑: Acknowledge Packet Format

|  |  |  |  |
| --- | --- | --- | --- |
| Description | Packet Length | Packet ID | Data |
| Field Length | 2 Bytes | 2 Bytes | 2 Bytes |
| Value | 0x0004 (4) | 0x0001 (1) | See Table for Error Codes |

### Connection Request

Table ‑: Connection Request Packet Format

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Description | Packet Length | Packet ID | Role | Command Set Major | Command Set Minor |
| Field Length | 2 Bytes | 2 Bytes | 1 Byte | 2 Bytes | 2 Bytes |
| Value | 0x0007 (7) | 0x0002 (2) | See Table | MSW Command Set Version | LSW of Command Set Version |

### Disconnection Request

Table ‑: Disconnection Request Packet Format

|  |  |  |
| --- | --- | --- |
| **Description** | **Packet Length** | **Packet ID** |
| **Field Length** | **2 Bytes** | **2 Bytes** |
| **Value** | **0x0002(2)** | **0x0003 (3)** |

### Force Disconnection

Table ‑: Force Disconnection Request Packet Format

|  |  |  |
| --- | --- | --- |
| **Description** | **Packet Length** | **Packet ID** |
| **Field Length** | **2 Bytes** | **2 Bytes** |
| **Value** | **0x0002 (2)** | **0x0004 (4)** |

### Client Env. Enumerate

Table ‑: Client Enumerate Packet Format

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Description** | **Packet Length** | **Packet ID** | **Env. Size X** | **Env. Size Y** | **Env. Size Z** |
| **Field Length** | **2 Bytes** | **2 Bytes** | **8 Bytes** | **8 Bytes** | **8 Bytes** |
| **Value** | **0x001A (26)** | **0x0005 (5)** |  |  |  |

### Req. Add Object

Table ‑: Client Enumerate Packet Format

|  |  |  |
| --- | --- | --- |
| **Description** | **Packet Length** | **Packet ID** |
| **Field Length** | **2 Bytes** | **2 Bytes** |
| **Value** | **0x0002 (2)** | **0x0006 (6)** |

### Update Object Position

Table ‑: Client Enumerate Packet Format

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Description** | **Packet Length** | **Packet ID** | **Object ID** | **X** | **Y** | **Z** | **Theta** | **Phi** |
| **Field Length** | **2 Bytes** | **2 Bytes** | **2 Bytes** | **4 bytes** | **4 bytes** | **4 Bytes** | **4 Bytes** | **4 Bytes** |
| **Value** | **0x0018 (24)** | **0x0007 (7)** |  |  |  |  |  |  |

### Bulk Update Objects

Table ‑: Client Enumerate Packet Format

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Description** | **Packet Length** | **Packet ID** | **# Objects** | **Object #1 Data** | **Object # … Data** | **Object #n Data** |
| **Field Length** | **2 Bytes** | **2 Bytes** | **2 Bytes** | **22 Bytes** | **22 Bytes** | **22 Bytes** |
| **Value** | **0x001A (26) to**  **0x03F8 (1016)** | **0x0008 (8)** | **See #1 Below** | **See #2 Below** | | |

1. The # Objects Field Specifies how many of the objects this packet updates. Valid values for this field are between 1 and 46. This is to ensure the packet length stays below the 1024 byte limit.
2. The format of the Object data is similar to that of the Update Object Position Packet except it does not require the header.

Table ‑: Client Enumerate Packet Format

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 7**Description** | **Object ID** | **X** | **Y** | **Z** | **Theta** | **Phi** |
| **Field Length** | **2 Bytes** | **4 bytes** | **4 bytes** | **4 Bytes** | **4 Bytes** | **4 Bytes** |
| **Value** |  |  |  |  |  |  |

Each object data field contains the data as shown in the above table.

The obvious advantages of this packet is that there is a reduced overhead of up to 178 bytes (for 46 objects) when compared to using the update Object Position Command.

### Delete Object

Table ‑: Client Enumerate Packet Format

|  |  |  |  |
| --- | --- | --- | --- |
| **Description** | **Packet Length** | **Packet ID** | **Object ID** |
| **Field Length** | **2 Bytes** | **2 Bytes** | **2 Bytes** |
| **Value** | **0x0004 (4)** | **0x0009 (9)** |  |

### Object Forces Packet

Table ‑: Client Enumerate Packet Format

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Description** | **Packet Length** | **Packet ID** | **Object ID** | **Force** | **Force Theta** | **Force Phi** |
| **Field Length** | **2 Bytes** | **2 Bytes** | **2 Bytes** | **8 Bytes** | **8 Bytes** | **8 Bytes** |
| **Value** | **0x001C (28)** | **0x000A (10)** |  |  |  |  |

### Client State Update

Table 5‑14: Client Enumerate Packet Format

|  |  |  |  |
| --- | --- | --- | --- |
| **Description** | **Packet Length** | **Packet ID** | **Status Flags** |
| **Field Length** | **2 Bytes** | **2 Bytes** | **2 Bytes** |
| **Value** | **0x001C (28)** | **0x000B (11)** |  |

### Tests – Echo Reply

Table ‑: Tests – Echo Reply Packet Format

|  |  |  |  |
| --- | --- | --- | --- |
| **Description** | **Packet Length** | **Packet ID** | **Message** |
| **Field Length** | **2 Bytes** | **2 Bytes** | **Up to 1022 Bytes** |
| **Value** | **0x0004 (4) up to 0x03FE (1022)** | **0xFFFE (65534)** | **Any String of Characters** |

### Tests – Echo

Table ‑: Test – Echo Packet Format

|  |  |  |  |
| --- | --- | --- | --- |
| **Description** | **Packet Length** | **Packet ID** | **Message** |
| **Field Length** | **2 Bytes** | **2 Bytes** | **Up to 1022 Bytes** |
| **Value** | **0x0004 (4) up to 0x03FE (1022)** | **0xFFFF (65535)** | **Any String of Characters** |

# Acknowledge & Error Codes

## Ack. Codes

Table ‑: Ack. Codes

|  |  |  |  |
| --- | --- | --- | --- |
| Data Word 1 | Data Word 2 | Name | Description |
| 0x0000 (0) | 0x0000 (0) | General Ack. |  |
| 0x0001 (1) | Disconnected Client ID | Client Disconnected |  |
| 0x0002 (2) | Assigned Client ID | Client Connected |  |
| 0x0003 (3) | 0x0000 (0) | Client Enumerated |  |
| 0x0004 (4) | Assigned Object ID | Object Added |  |
| 0x0005 (5) | Deleted Object ID | Object Deleted |  |
| 0xFFFF (65536) | 0x0000 (0) | Unspecified Ack. |  |

## Error Codes

Table ‑: Error Codes

|  |  |  |
| --- | --- | --- |
| Code | Name | Description |
| 0x0000 (0) | No Error |  |
| 0x0001 (1) | Client Connected |  |
| 0x0002 (2) | Client Not Connected |  |
| 0x0003 (3) | Disconnect Fail DB Error |  |
| 0xFFFF (65536) | Unspecified Error |  |