NEURONSCAPE SYSTEM SPECIFICATION

VERSION 0.03 DRAFT WORKING COPY

Julian Bailey, Peter Wilson & John Chad

School of Electronics & Computer Science

University of Southampton, UK

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

17th June 2011

Table of Contents

[1 Definitions & Abbreviations 4](#_Toc297340906)

[2 Neuronscape System Structure 5](#_Toc297340907)

[3 Environment Server 6](#_Toc297340908)

[3.1 Neuron Environment Interface 6](#_Toc297340909)

[3.1.1 Sensory Input 6](#_Toc297340910)

[3.1.2 Muscle Output 7](#_Toc297340911)

[3.2 Interactor 7](#_Toc297340912)

[4 Environment 7](#_Toc297340913)

[4.1 Coordinate Systems 7](#_Toc297340914)

[4.2 Environmental Physics 7](#_Toc297340915)

[5 Server Database Structure 8](#_Toc297340916)

[5.1 Clients Table 8](#_Toc297340917)

[5.2 Objects Table 9](#_Toc297340918)

[6 Network Communication 10](#_Toc297340919)

[6.1 Packet Structure 10](#_Toc297340920)

[6.2 Packet Types 10](#_Toc297340921)

[6.3 Packet Types Detail 11](#_Toc297340922)

[6.3.1 Acknowledge (ACK) 11](#_Toc297340923)

[6.3.2 Error (ERR) 11](#_Toc297340924)

[6.3.3 Connection Request 11](#_Toc297340925)

[6.3.4 Disconnection Request 11](#_Toc297340926)

[6.3.5 Force Disconnection 12](#_Toc297340927)

[6.3.6 Client Env. Enumerate 12](#_Toc297340928)

[6.3.7 Req. Add Object 12](#_Toc297340929)

[6.3.8 Update Object Position 12](#_Toc297340930)

[6.3.9 Bulk Update Objects 13](#_Toc297340931)

[6.3.10 Delete Object 13](#_Toc297340932)

[6.3.11 Object Forces Packet 14](#_Toc297340933)

[6.3.12 Client State Update 14](#_Toc297340934)

[6.3.13 Tests – Echo Reply 14](#_Toc297340935)

[6.3.14 Tests – Echo 14](#_Toc297340936)

[6.4 Acknowledge & Error Codes 15](#_Toc297340937)

[6.4.1 Ack. Codes 15](#_Toc297340938)

[6.4.2 Error Codes 15](#_Toc297340939)

[7 BIMPA Animal Specification V1.0 16](#_Toc297340940)

[8 Neuronscape Timeline (June 2011 -> June 2012) 17](#_Toc297340941)

[8.1 June 2011 17](#_Toc297340942)

[8.2 July 2011 17](#_Toc297340943)

[8.3 August 2011 17](#_Toc297340944)

[8.4 September 2011 17](#_Toc297340945)

[8.5 October 2011 17](#_Toc297340946)

[8.6 November 2011 17](#_Toc297340947)

[8.7 December 2011-Feb 2012 17](#_Toc297340948)

[8.8 Feb 2012->June 2012 17](#_Toc297340949)

# Definitions & Abbreviations

TO DO.

# Neuronscape System Structure

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



Figure 2.1: 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.

## Interactor

The interactor 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 Interactor 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 Interactor 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

All systems use a 3D Cartesian coordinate system for storage. Coordinates can be converted between Cartesian and spherical easily using built-in classes.

## Environmental Physics

Newtonian physics equations are implemented, and include static friction, kinetic friction and quadratic drag. The physics system is implemented using a 1ms time-step and 4th order Runge-Kutta explicit integration.

# Server Database Structure

The database structure in the server SQL database is outline below for each of the tables.

## 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 |  |

The clients table stores information about associated clients. The id is assigned by the server when the clients connects and along with the IP address (IPv4 currently) and UDP port identifies the client uniquely. The role field specifies the role the client fulfils; this can be Server, NEI or Interactor role.

The fields Command Set Major/Minor identify the highest protocol version that the clients can communicate in. This is intended to be read in the format of “MAJOR:MINOR” version. It will allow a mixture of clients using different protocol versions to use the system. This may or may not be needed but has been included for future proofing.

The status field indicates if the client is currently reachable; the idea here is that if a client becomes disconnected due to network connectivity problems it should be marked as unreachable. This should allow the client to reconnect in the future once network problems are resolved.

## 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\_x | SQL\_DOUBLE |  |
| motorforce\_Y | SQL\_DOUBLE |  |
| motorforce\_Z | SQL\_DOUBLE |  |
| Xternforce\_X | SQL\_DOUBLE |  |
| xternforce\_Y | SQL\_DOUBLE |  |
| xternforce\_Z | SQL\_DOUBLE |  |
| Velocity\_X | SQL\_DOUBLE |  |
| velocity\_Y | SQL\_DOUBLE |  |
| Velocity\_Z | SQL\_DOUBLE |  |

The objects table stores information about the state of the environmental objects. The ID field is a value which identifies each object uniquely and is auto assigned by the server when a new object is created. The client\_id field bind an object to a connected client. If this field is zero then the object is inanimate.

Position and orientation is provided by the Cartesian coordinates in the X, Y, Z fields and orientation is provided by theta and phi, where theta is the left/right turning orientation and phi represents the ability to look up or down. This is specified in meters from the zero point and radians.

Motorforce X, Y and Z represent the “vector thrust” locomotive force acting on the object. This field is updated by values in the NEI. This is specified in Newton’s.

Xternforce X, Y and Z represent the vector of an external force applied by the user who in interacting with the environment through the Interactor program. This is specified in Newton’s.

Velocity X, Y and Z are the vector components of the current velocity of the object. These are specified in meters per second.

# 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 5‑1: 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 5‑2: 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 5‑3: 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 5‑4: 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 5‑5: 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 5‑6: Disconnection Request Packet Format

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

### Force Disconnection

Table 5‑7: 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 5‑8: 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 5‑9: Client Enumerate Packet Format

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

### Update Object Position

Table 5‑10: 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 5‑11: 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 5‑12: 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 5‑13: 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 5‑14: 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 5‑15: 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 5‑16: 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 6‑1: 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 7‑1: 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 |  |

# BIMPA Animal Specification V1.0

|  |  |
| --- | --- |
| Weight | 1kg |
| Shape | Sphere |
| Diameter | 10cm |
| Vision Plane | 320x240 pixels greyscale |
| Vision Orientation | Front |
| Vision Type | Monocular |
| Movement | Front, Spin Left (random) |
| Neuron Count | 100 |
| Health Status | 100% indicator |
| Health reduction | Time (respiration) – 1%/hour |
| Movement default | Continuous forward if Health > 10% |

Need some more precision on the default animal parameters in some fluid medium?

# Neuronscape Timeline (June 2011 -> June 2012)

## June 2011

* Basic OpenGL Infrastructure for Texture Rendering and Object Placement and Movement
* Basic Animal Implementation in Test Environment

## July 2011

* Animal 1.0 to be implemented in Environment
* Manual (external) Force Interface to be added
* Rob Mills to start on Project

## August 2011

* Vision plane – animal perspective to be implemented
* Vision plane – “God” view to be implemented

## September 2011

* Document Environment
* Train Rob Mills in detail on Neuronscape

## October 2011

* Document Environment
* Train Rob Mills in detail on Neuronscape

## November 2011

* Spinnaker Integration to Neuronscape

## December 2011-Feb 2012

* Demonstration Example Development
* Paper on Neuronscape with basic animal implementation

## Feb 2012->June 2012

* Demonstration Example Development
* Animal Enhancements – higher number of Neurons