# NEURONSCAPE SYSTEM SPECIFICATION

VERSION 0.81 DRAFT WORKING COPY

Julian Bailey, Peter Wilson & John Chad

School of Electronics & Computer Science

University of Southampton, UK

jab@ecs.soton.ac.uk, prw@ecs.soton.ac.uk, jec@soton.ac.uk

2nd Feb 2012

# 1. TABLE OF CONTENTS

## 2. DEFINITIONS & ABBREVIATIONS

TO DO.

# 3. NEURONSCAPE SYSTEM STRUCTURE

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

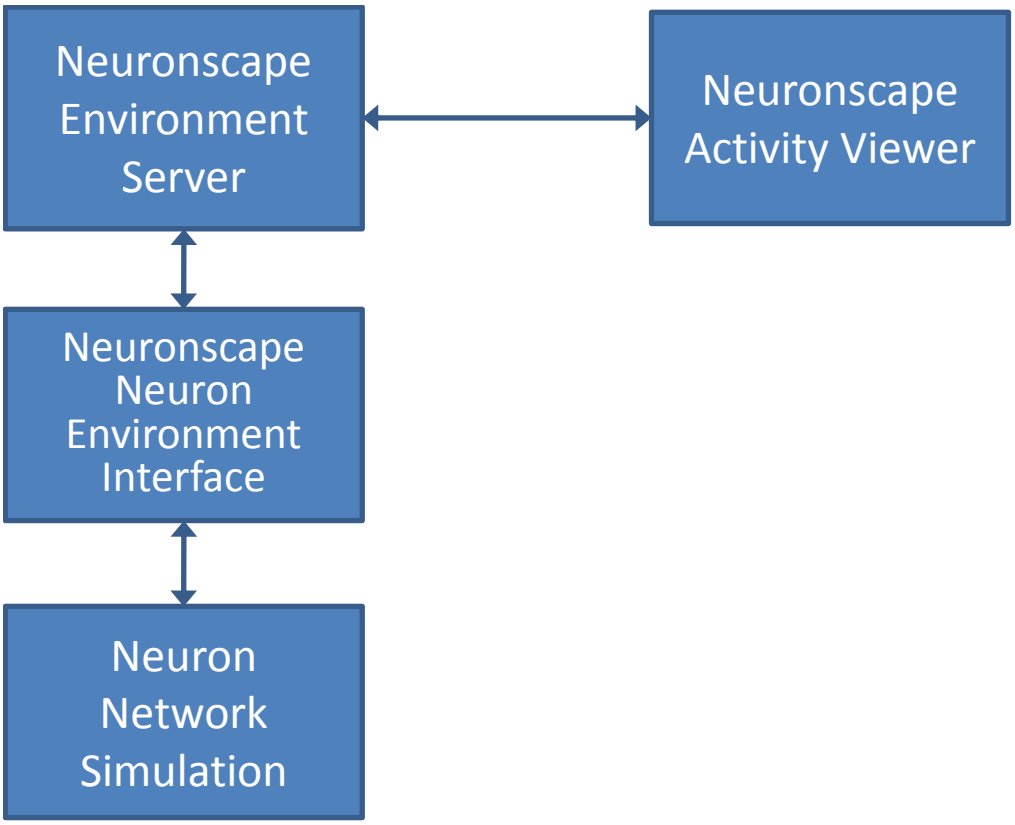


**Figure 3.1: An overall view of the neuronscape system**

The overall system view is shown in Figure 3.1, 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.

# 4. 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 $4^{th}$ order Runge-Kutta explicit integration.

# 5. 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<br>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<br>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.

# 6. 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 6-1) 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 6-1: Generic Packet Format**

| Description | Packet ID | Payload |
|---|---|---|
| Field Length | 2 Bytes | =< 1020 Bytes |
| Value | See Table 6-2 | Dependent on Packet Type |

## Packet Types

Table 6-2: Packet Types

| Packet ID | Name |
|---|---|
| 0 (0x0000) | Acknowledge (ACK) |
| 1 (0x0001) | Error (ERR) |
| 2 (0x0002) | Connection Request |
| 3 (0x0003) | Disconnection Request |
| 4 (0x0004) | Force Disconnection |
| 5 (0x0005) | Client Enumerate |
| 6 (0x0006) | Req. Add Object |
| 7 (0x0007) | Update Object Position (Deprecated) |
| 8 (0x0008) | Bulk Update Objects |
| 9 (0x0009) | Delete Object |
| 10 (0x000A) | Object Forces Packet |
| 11 (0x000B) | Torque Forces |
| 12 (0x000C) | Energy Delta (Deprecated) |
| 13 (0x000D) | Colour Change |
| 14 (0x000E) | Add Inanimate Object |
| 15 (0x000F) | Eat Object |
| 16 (0x0010) | Remove Eaten Object |
| 17 (0x0011) | Kill System |
| 65534 (0xFFFE) | Tests - Echo Reply |
| 65535 (0xFFFF) | Tests - Echo |

## Packet Types Detail

### ACKNOWLEDGE (ACK)

**Table 6-3: Acknowledge Packet Format**

| Description | Packet ID | Data Word 1 | Data Word 2 |
|---|---|---|---|
| Field Length | 2 Bytes | 4 Bytes | 4 Bytes |
| Value | 0x0000 (0) | | |

An acknowledge packet is sent as a positive response to requests, such as connection requests.

Data word 1 is always the packet ID which this acknowledge packet is in response to.

Data Word 2 is Zero or the value that should be passed back to the original sender.

An example is acknowledging to a connection request: Data word 1 would be 2 (indicating it is in response to a connection request) and Data word 2 would contain the system ID assigned by the server to the client.

### ERROR (ERR)

**Table 6-4: Error Packet Format**

| Description | Packet ID | Data Word 1 | Data Word 2 | Data Word 3 |
|---|---|---|---|---|
| Field Length | 2 Bytes | 4 Bytes | 4 Bytes | 4 Bytes |
| Value | 0x0000 (0) | | | |

An error packet is sent as a negative response to requests.

As with Acknowledge packets data word 1 contains the id of the packet that caused the error.

Data word 2 contains a value which corresponds with the error. These are dependent on the packet that caused the error.

Data word 3 can contain extra information

## CONNECTION REQUEST

**Table 6-5: Connection Request Packet Format**

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

The connection request is sent by a client to the server to request participation in the network. The role id is shown in the following table.

| Role | ID |
|---|---|
| Unspecified | 0 |
| Server | 1 |
| Interactor | 2 |
| Neuron Environment Interface(N.E.I.) | 3 |
| Controller | 4 |

The Command Set Major/Minor tell the server what version of the Neuronscape Network Protocol the client is using.

**Responses:**

Acknowledge: Data word 1 = 2, Data word 2 = Assigned client ID.

Error:

Data Word 1 = 2

Data Word 2:

- 0 – Unspecified Error
- 1 – Invalid Role:  An non existant role was specified
- 2 – Server Role: A server cannot connect to a server
- 3 – Command Set Error: The command set version specified is not supported, see the value in data word 3 for the version the server is running  (Little endian, Major:Minor)
- 4 – Server Busy: Server cannot process the request at this time
- 5 – Server Full: Server cannot accept the connection of any more clients

## DISCONNECTION REQUEST

**Table 6-6: Disconnection Request Packet Format**

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

A client wishes to disconnect from the server. The server will use the clients source ip address and port to send an acknowledge with Data word 1 = 3 and Data Word 2 = Clients Assigned ID.

This serves as a check that the client actually sent the disconnection request. If the ID in the acknowledge packet matches the id assigned by the server after connection request then the client can disconnect.

If the client receives an acknowledge for disconnect with the incorrect ID set it should ignore it.

The client should wait up to 5 seconds for the acknowledge to a disconnection request before exiting. If 5 seconds elapses without receiving a disconnect acknowledge from the server then it si safe to assume the server has hung and the client should exit anyway.

## FORCE DISCONNECTION

**Table 6-7: Force Disconnection Request Packet Format**

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

A force disconnect is sent by a server to all attached clients to inform them the server is shutting down. No acknowledge should be sent in response to a force disconnect. All clients should cease participation in the neuronscape network.

## CLIENT ENV.ENUMERATE

**Table 6-8: Client Environment Enumerate Packet Format**

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

A client enumerate packet is sent to the client from the server directly after the acknowledgement to a connection request.

It tells the client what the bounds of the environment are. X, Y & Z are double floating point numbers.

No response should be sent from the client to the server for this packet. Sending a response will result in an error packet being sent by the server to the client.

### REQ. ADD OBJECT

**Table 6-9: Request Add Object Packet Format**

| Description | Packet ID | Initial Energy | X | Y | Z |
|---|---|---|---|---|---|
| Field Length | 2 Bytes | 8 Bytes | 4 Bytes | 4 Bytes | 4 Bytes |
| Value | 0x0006 (6) | | See Description Below | | |

Initial energy is a positive double floating point value.

The fields X, Y, Z specify the desired position the client wishes the object to be added at. These fields are **optional** and can be left out if they are not needed. The server will attempt to place the object at the desired coordinates but it is not required to do so. All values of X, Y and Z must be positive, negative values in any of the fields will cause the server to place the new object wherever it chooses.

**The length of the packet is 10 when X, Y, Z are not included and 34 when they are included, any other value is invalid.**

**Responses:**

Acknowledge: Data Word 1 = 6, Data Word 2 = Assigned object ID

Error:

Data Word 1 = 6

Data Word 2:

- 0 – Unspecified Error
- 1 – Bad Packet Format: Packet is not 10 or 34 bytes long and is invalid
- 2 – Client is not an N.E.I so cannot own an object
- 3 – Client already owns an object (Clients may only own one object at a time)
- 4 – Server Database error: Server failed to add a new object to the object database
- 5 – Server Database Verify Error: Server could not verify the object has been added to the object database. The database may be inconsistent!

## BULK UPDATE OBJECTS

The bulk update object packet is sent to all connected clients each time the server completes a new time step. It is used to inform all the clients of the positions of the objects in the environment and their attributes. Each bulk update packet can contain the attributes for up to 17 objects. If more than 17 objects exist in the environment then several bulk update packets may be sent.

**Table 6-10: Bulk Update Objects Packet Format**

| Description | Packet ID | # Objects | Object #1 Data | Object # … Data | Object #n Data |
|---|---|---|---|---|---|
| Field Length | 2 Bytes | 4 Bytes | 58 Bytes | 58 Bytes | 58 Bytes |
| Value | 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 1000 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 6-11: Object Data Format**

| Description | Object ID | X | Y | Z | Theta | Phi | R | G | B | L | Energy | Flags |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Field Length (bytes) | 4 | 8 | 8 | 8 | 8 | 8 | 1 | 1 | 1 | 1 | 8 | 2 |
| Value | | | | | | | | | | | | |

- ID is the ID of the object whose information is contained in this object data section.
- X, Y, Z are double floating point values specifying the position of the object in the environment.
- Theta, Phi are double floating point values specifying the direction in radians that the object is facing. Theta being the rotation on the X-Y plane and Z being the angle the object is looking up or down.
- R, G, B, L are the values for Red, Green, Blue and Luminosity of the object.
- Energy is the amount of energy the object has.
- Flags specify special attributed the object has, these are shown in the table below.

| Flag | Definition | Explanation |
|---|---|---|
| 0x0001 | Fixed | The object is fixed and cannot move. If something collides with it the object it will remain in its original position. |
| 0x0002 | Edible | The object can be eaten |
| 0x0004 | No Collide | Objects will pass through this object instead of colliding with it |
| 0x0008 | Inanimate | The object is inanimate and cannot perform any action |
| 0x0010 | N/A | Unassigned |
| 0x0020 | N/A | Unassigned |
| 0x0040 | N/A | Unassigned |
| 0x0080 | N/A | Unassigned |
| 0x0100 | N/A | Unassigned |
| 0x0200 | N/A | Unassigned |
| 0x0400 | N/A | Unassigned |
| 0x0800 | N/A | Unassigned |
| 0x1000 | N/A | Unassigned |
| 0x2000 | N/A | Unassigned |
| 0x4000 | N/A | Unassigned |
| 0x8000 | N/A | Unassigned |

The flags are assigned so that objects can have several of the attributes defined by these flags assigned at any one time. Flags are tested by checking if the relevant bit is set in the flags field.

**Responses:**

No response shall be sent in reply to a Bulk Update Packet.

## DELETE OBJECT

**Table 6-12: Delete Object Packet Format**

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

A delete object packet can be sent in two situations:

- A client telling a server that it wishes to remove the object it owns from the environment
- A server telling all the other clients an object has been removed from the environment and that they should remove the object with the ID contained in the packet from their internal databases.

**Client -> Server Response:**

A server shall respond in the following way upon reception of a Delete Object Packet.

**Acknowledge:** Data Word 1= 9, Data Word 2 = Deleted Object ID

**Error:**

Data Word 1: 9

Data Word 2:

- 0 – Unspecified Error
- 1 – Client is not an N.E.I. and therefore cannot own an object
- 2 – Client does not own an object or does not own the object specified by Object ID
- 3 – Server failed to remove object from database

**Server -> Client Response:**

No response shall be sent in reply to a delete object command from the server.

## OBJECT FORCES PACKET

**Table 6-13: Object Forces Packet Format**

| Description | Packet ID | Object ID | Force X | Force Y | Force Z |
|---|---|---|---|---|---|
| Field Length | 2 Bytes | 4 Bytes | 8 Bytes | 8 Bytes | 8 Bytes |
| Value | 0x000A (10) | | | | |

The object forces packet is sent from the client to the server to set either the motor force or an external force acting on the object.

An N.E.I. sending this packet will cause the server to set the motor force whist an interactor will cause the external force to be set.

Force X, Y, Z is a 3d Cartesian vector (each component is a double floating point number) representing a force applied at the point the object is currently located.

**Response (Server to Client):**

**Acknowledge:** Data Word 1 = 10, Data Word 2 = 0

**Error:**

Data Word 1 = 10

Data Word 2:

- 0 – Unspecified Error
- 1 – Bad packet: Packet is not 30 bytes long
- 2 – Client is not an N.E.I. or an interactor
- 3 – Object with given object ID does not exist
- 4 – Object does not belong to the N.E.I. who tried to apply the force

## TORQUE PACKET

**Table 6-14: Torque Packet Format**

| Description | Packet ID | Object ID | Torque Theta | Torque Phi |
|---|---|---|---|---|
| Field Length | 2 Bytes | 4 Bytes | 8 Bytes | 8 Bytes |
| Value | 0x000B (11) | | | |

An N.E.I uses this packet to set a rotational force on an object specified by *Object ID.*

Torque Theta is a rotational force on the X-Y plane and is a double floating point number.

Torque Phi is a rotational force in the Z direction and is a double floating point number.

**Response (Server to Client):**

**Acknowledge:** Data Word 1 = 11, Data Word 2 = 0

**Error:**

Data Word 1 = 11

Data Word 2:

- 0 – Unspecified Error
- 1 – Bad packet: Packet is not 22 bytes long
- 2 – Client is not an N.E.I
- 3 – Object with given object ID does not exist
- 4 – Object does not belong to the N.E.I. who tried to apply the force

## COLOUR PACKET

**Table 6-15: Colour Packet Format**

| Description | Packet ID | Object ID | Red | Green | Blue | Brightness |
|---|---|---|---|---|---|---|
| Field Length | 2 Bytes | 4 Bytes | 1 Byte | 1 Byte | 1 Byte | 1 Byte |
| Value | 0x000D (13) | | | | | |

A colour packet is sent from a client to a server to change its colour.

**Response (Server to Client):**

**Acknowledge:** Data Word 1 = 13, Data Word 2 = 0

**Error:**

Data Word 1 = 13

Data Word 2:

- 0 – Unspecified Error
- 1 – Bad packet: Packet is not 10 bytes long
- 2 – Client is not an N.E.I
- 3 – Object with given object ID does not exist
- 4 – Object does not belong to the N.E.I. who tried to change the colour

## ADD INANIMATE OBJECT PACKET

**Table 6-16: Add Inanimate Packet Format**

| Description | Packet ID | X | Y | Z | Red | Green | Blue | Luminosity | Energy | FLAGS |
|---|---|---|---|---|---|---|---|---|---|---|
| Field Length | 2 Bytes | 8 Bytes | 8 Bytes | 8 Bytes | 1 Byte | 1 Byte | 1 Byte | 1 Byte | 4 Bytes | 2 Bytes |
| Value | 0x000E (14) | | | | | | | | | |

This packet is sent from an interactor to the server to add an inanimate object to the environment.

The fields X, Y, Z (double floating point numbers) specify the desired position the client wishes the object to be added at. The server will attempt to place the object at the desired coordinates but it is not required to do so. All values of X, Y and Z must be positive, negative values in any of the fields will cause the server to place the new object wherever it chooses.

Red, Green, Blue and Luminosity set the colour and luminosity of the object.

Energy is a double floating point number specifying the amount of energy the object contains. This value will be added to an animals object if it is eaten. This value may be positive or negative.

Flags specify special attributed the object has, these are shown in the table below.

| Flag | Definition | Explanation |
|---|---|---|
| 0x0001 | Fixed | The object is fixed and cannot move. If something collides with it the object it will remain in its original position. |
| 0x0002 | Edible | The object can be eaten |
| 0x0004 | No Collide | Objects will pass through this object instead of colliding with it |
| 0x0008 | Inanimate | The object is inanimate and cannot perform any action |
| 0x0010 | N/A | Unassigned |
| 0x0020 | N/A | Unassigned |
| 0x0040 | N/A | Unassigned |
| 0x0080 | N/A | Unassigned |
| 0x0100 | N/A | Unassigned |

| | | |
|---|---|---|
| 0x0200 | N/A | Unassigned |
| 0x0400 | N/A | Unassigned |
| 0x0800 | N/A | Unassigned |
| 0x1000 | N/A | Unassigned |
| 0x2000 | N/A | Unassigned |
| 0x4000 | N/A | Unassigned |
| 0x8000 | N/A | Unassigned |

The flags are assigned so that objects can have several of the attributes defined by these flags assigned at any one time. The inanimate flag will be set automatically by the server.

**Response (Server to Client):**

**Acknowledge:** Data Word 1 = 14, Data Word 2 = Assigned object ID

**Error:**

Data Word 1 = 14

Data Word 2:

- 0 – Unspecified Error
- 1 – Bad packet: Packet is not 36 bytes long
- 2 – Client is not an interactor
- 3 – Inserting object to Server object database failed. Database may now be inconsistent.

## EAT OBJECT PACKET

**Table 6-17: Eat Inanimate Packet Format**

| Description | Packet ID | Object ID |
|---|---|---|
| Field Length | 2 Bytes | 4 Bytes |
| Value | 0x000F (15) | |

This packet is sent from a client to a server to indicate the client wished to try to eat whatever is in front of it.

Object ID indicates the ID of the object that wishes to try to eat.

**Response (Server to Client):**

**Acknowledge:** Data Word 1 = 15, Data Word 2 = 0

**Error:**

Data Word 1 = 15

Data Word 2:

- 0 – Unspecified Error
- 1 – Bad packet: Packet is not 6 bytes long
- 2 – Client does not own an object
- 3 – Inserting object to Server object database failed. Database may now be inconsistent.

## REMOVE EATEN OBJ

**Table 6-18: Remove Eaten Packet Format**

| Description | Packet ID | Object ID |
|---|---|---|
| Field Length | 2 Bytes | 4 Bytes |
| Value | 0x000F (16) | |

This packet is sent from server to all clients to tell them that an object has been eaten in the environment.

They should check the ID from the packet. They should remove the relevant object from their local databases.

If the client is an N.E.I. then it should check that the object is not the one that belongs to the N.E.I. If it is then the N.E.I.'s object has been eaten and the N.E.I should exit.

**Response: No Response should be sent if this packet is received.**

## KILL SYSTEM

**Table 6-19: Kill System**

| Description | Packet ID | Passphrase |
|---|---|---|
| Field Length | 2 Bytes | Up to 32 Bytes |
| Value | 0x0011 (17) | Null terminated string ('\0'), padded to 32 bytes |

This command is sent to the server from a Control Client.

This causes the server to broadcast this packet to all clients and then terminate. Client can choose not to terminate but it is recommended.

The passphrase must match the one set in the server otherwise an error will be returned.

**Response (Server to Client):**

**Acknowledge:** Data Word 1 = 16, Data Word 2 = 0

**Error:**

Data Word 1 = 16

Data Word 2:

- 0 – Unspecified Error
- 1 – Bad packet: Packet is not 34 bytes long
- 2 – Client is not a control client
- 3 – Bad Passphrase
- 4 – Remote Kill is disabled

**Response (Clients to Server): No Response shall be given, server will not respond**

## TESTS – ECHO REPLY

**Table 6-20: Tests – Echo Reply Packet Format**

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

This packet is sent in reply to a tests echo packet and contains the data that was sent in the original packet. It can be sent by any client to the server or the server to a client.

**Response: No Response shall be given**

## TESTS – ECHO

**Table 6-21: Test – Echo Packet Format**

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

This packet is used to test network communications between a client and a server or server and client. The packet contains a string which shall be returned in a Tests- Echo Reply packet.

**Response: A Tests Echo Reply packet containing the original message**

# Acknowledge & Error Codes

## ACK. CODES

**Table 6-22: Ack. Codes**

| In Response To | Data Word 1 | Data Word 2 |
|---|---|---|
| Connection Request | 0x0002 | Assigned Client ID |
| Disconnection Request | 0x0003 | Clients ID |
| Request Add Object | 0x0006 | Added Object ID |
| Delete Object | 0x0009 | Deleted Object ID |
| Object Force | 0x000A | 0 |
| Object Torque | 0x000B | 0 |
| Colour | 0x000D | 0 |
| Add Inanimate | 0x000E | Added Object ID |
| Eat Object Request | 0x000F | 0 |
| Kill System | 0x0011 | 0 |

**Table 6-23: Error Codes**

| Code | Name | Description |
|---|---|---|
| 0x0000 (0) | No Error | |
| 0x0001 (1) | Client Connected | |
| 0x0002 (2) | Client Not Connected | |
| 0x0003 (3) | Disconnect Fail DB Error | |
| 0x000C (12) | Server Platform Unsupported | |
| 0x000E (14) | CRC Error/Data Not Okay | |
| 0xFFFF (65536) | Unspecified Error | |

# 7. 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. There will be variants on this defined in Summer 2011.

# 8. 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

- Neural network/NEI/Environment/Interactor Initial Demonstrator
- Document Environment
- Train Rob Mills in detail on Neuronscape

## October 2011

- Neural network/NEI/Environment/Interactor Initial Demonstrator
- 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

## July 2012

- Demonstrator #2 : Exploration of Biological System Impact

- Evolution
- Neuron Size
- Swarm/Herd Behaviour
- Success and Failure comparison