|  |
| --- |
| Martin Rule, Lane Cotgrove, James Bayliss |
| Motion Project |
| Feature 1.3 Create connection to server |

|  |
| --- |
| Martin Rule  2/7/2012 |

Table of Contents

[1. Overview 2](#_Toc337424134)

[2. Feature team 2](#_Toc337424135)

[3. Sequence diagram 3](#_Toc337424136)

[4. Overall object model 4](#_Toc337424137)

[5. Method prologues 5](#_Toc337424138)

[6. Testing 9](#_Toc337424139)

[7. Design inspection 10](#_Toc337424140)

[8. References 11](#_Toc337424141)

## 1. Overview

This feature will allow us to connect the Kinect client to both local and remote servers. To do this the program will gather information about the server from the user using the programs GUI. This information will be passed to the network model and then used to attempt a connection to the relevant server. After the connection is created it is then tested and the user is informed of the result.

## 2. Feature team

For the design of this feature we are using the following team members.

Martin Rule – Project Manager, Developer  
Lane Cotgrove – Lead developer  
James Bayliss – Developer, Tester

## 3. Sequence diagram



## 4. Overall object model



## 5. Method prologues

//---------------------------------------------------  
// Name: connect()   
// Author:   
// Inputs: STRING IPaddress  
// INT port   
// Outputs: BOOL success  
//   
// Desc: This function allows a network model to setup a   
// connection to a server  
//---------------------------------------------------  
  
//---------------------------------------------------  
// Name: disconnect()  
// Author:   
// Inputs: NULL   
// Outputs: BOOL success  
//   
// Desc: This function allows a network model to disconnect // from it's previous connection  
//---------------------------------------------------  
  
//---------------------------------------------------  
// Name: testConnection()   
// Author:   
// Inputs: NULL   
// Outputs: BOOL success  
//   
// Desc: This function tests a previously set up connection // for a response  
//---------------------------------------------------

//---------------------------------------------------  
// Name: isConnected()   
// Author:   
// Inputs: NULL   
// Outputs: BOOL success  
//   
// Desc: This function tests a previously set up connection // for a response  
//---------------------------------------------------  
  
//---------------------------------------------------  
// Name: isIPAddress()   
// Author:   
// Inputs: STRING address   
// Outputs: BOOL success  
//   
// Desc: This function tests to see if a string is in the   
// correct format to be an IP address  
//---------------------------------------------------  
  
//---------------------------------------------------  
// Name: isNumber()   
// Author:   
// Inputs: CHAR c   
// Outputs: BOOL success  
//   
// Desc: This function tests to see if the value stored in a // character is a number  
//---------------------------------------------------  
  
//---------------------------------------------------  
// Name: isAlpha()  
// Author:   
// Inputs: CHAR c   
// Outputs: BOOL success  
//   
// Desc: This function tests to see if the value stored in a // character is a letter  
//---------------------------------------------------

//---------------------------------------------------  
// Name: numberState()   
// Author: Todd Cochrane  
// Inputs: CHAR current  
// INT count  
// INT state  
// INT nextState   
// Outputs: NULL  
//   
// Desc: Refer to Todds notes  
//---------------------------------------------------  
  
//---------------------------------------------------  
// Name: httpGetTest()   
// Author: Todd Cochrane  
// Inputs: NULL   
// Outputs: NULL  
//   
// Desc: Refer to Todds notes  
//---------------------------------------------------  
  
//---------------------------------------------------  
// Name: HTTPget()  
// Author: Todd Cochrane  
// Inputs: STRING strURL   
// Outputs: NULL  
//   
// Desc: Refer to Todds notes  
//---------------------------------------------------  
  
//---------------------------------------------------  
// Name: HttpDomain()   
// Author: Todd Cochrane  
// Inputs: WSTRING strURL   
// Outputs: WSTRING  
//   
// Desc: Refer to Todds notes  
//---------------------------------------------------

//---------------------------------------------------  
// Name: HttpPath()   
// Author: Todd Cochrane  
// Inputs: WSTRING strURL   
// Outputs: WSTRING  
//   
// Desc: Refer to Todds notes  
//---------------------------------------------------  
  
//---------------------------------------------------  
// Name: validURL()   
// Author: Todd Cochrane  
// Inputs: WSTRING strURL   
// Outputs: BOOL  
//   
// Desc: Checks to see if string is valid URL  
//---------------------------------------------------

## 6. Testing

For this feature we will use unit tests to determine the accuracy of data being produced. Without an end client we are not able to define if the information is correct at this point. Instead we will check to make sure the information is within valid ranges and fully populated.

For testing this feature we will develop tests to ensure a connection has been made from the Kinect Client. The application is able to send data via two ways, one using a socket connection, the other by accessing a webserver. This meant we had to test the connection in multiple ways, by entering an IP address & port, and also by entering a web address. For these tests to be implemented the assert.h header file had to be inserted into the NetworkModel.cpp file which is where the connections are established.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Pass | Fail | Comments |
| NetworkModel.cpp – Confirming that the connection request is an IP address.  (Line 58) |  | 🗶 | These tests were designed to ensure that the Kinect Client will successfully connect to a server either by a socket connection or via a http request. The Socket Server tests have been put into a failed state as the Socket Server is now redundant and not being used. The Socket Server will be removed from the application. |
| NetworkModel.cpp – Confirming a socket connection has been created.  (Line 60) |  | 🗶 |
| NetworkModel.cpp – to ensure the address of the connection has been stored  (Line 73) | ✓ |  |
| NetworkModel.cpp – to confirm an Http connection has been made  (Line 76) | ✓ |  |
| NetworkModel.cpp – to confirm that a connection has been made  (Line 93) | ✓ |  |

## 7. Design inspection

Design inspection was performed by Martin Rule, Lane Cotgrove and James Bayliss.  
  
Advisor inspection was performed by Andrew Eales.

## 8. References

## 

Palmer, S. (2009). *An introduction to Feature Driven Development*. Retrieved from  
 <http://agile.dzone.com/articles/introduction-feature-driven>

Ambler, S. W. (2009).Feature Driven Development. Retrieved from  
 <http://www.agilemodeling.com/essays/fdd.htm>

Dawson, C. W. (2009). *Projects in Computing and Information Systems, A Student’s Guide*. Harlow, England: Pearson Education Limited.