|  |
| --- |
| Martin Rule, Lane Cotgrove, James Bayliss |
| Motion Project |
| Feature 1.2 Transform movement Data from data in memory |

|  |
| --- |
| Martin Rule  8/2/2012 |

Table of Contents

[1. Overview 2](#_Toc337421987)

[2. Feature team 2](#_Toc337421988)

[3. Sequence diagram 3](#_Toc337421989)

[4. Overall object model 4](#_Toc337421990)

[5. Class and method prologues 5](#_Toc337421991)

[6. Testing 7](#_Toc337421992)

[7. Design inspection 8](#_Toc337421993)

[8. References 8](#_Toc337421994)

## 1. Overview

This feature will transform data collected from the Kinect device into a format suitable for transfer over a network. The expected server architecture currently expects a string containing movement data. This feature will transform each segment of movement data required for each frame collected from the Kinect device into one string.



## 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

  
1. This function requests the a skeleton frame from the Kinect API

2. The skeleton frame requested is then stored in the same MData object

3. Each segment of the skeleton frame is then converted into a string and stored in a string buffer

4. The string buffer then returns the final string which is then stored within the same MData object

## 4. Overall object model



## 5. Class and method prologues

//---------------------------------------------------  
// Name: getMovementData()   
// Author: Lane Cotgrove - PeePeeSpeed  
// Inputs: NULL   
// Outputs: NUI\_SKELETON\_DATA  
//   
// Desc: This function retreives the skeleton data from an   
// MData object  
//---------------------------------------------------  
  
//---------------------------------------------------  
// Name: setMovementData()   
// Author: Lane Cotgrove - PeePeeSpeed  
// Inputs: NUI\_SKELETON\_DATA   
// Outputs: NULL  
//   
// Desc: This function allows you to set an MData object's   
// skeleton data  
//---------------------------------------------------  
  
//---------------------------------------------------  
// Name: transformMovementData()   
// Author: Lane Cotgrove - PeePeeSpeed  
// Inputs: NULL   
// Outputs: NULL  
//   
// Desc: This function transforms an MData object's skeleton // data into a string to be stored in transformedData  
//---------------------------------------------------

//---------------------------------------------------  
// Name: getI()   
// Author: Lane Cotgrove - PeePeeSpeed  
// Inputs: NULL   
// Outputs: INT i  
//   
// Desc: This function returns the skeleton number of the   
// skeleton frame stored within an MData object  
//---------------------------------------------------  
  
//---------------------------------------------------  
// Name: normaliseX   
// Author: Todd Cochrane  
// Inputs: FLOAT x   
// Outputs: FLOAT x  
//   
// Desc: This function normalises the value of x for an   
// external application  
//---------------------------------------------------  
  
//---------------------------------------------------  
// Name: normaliseY   
// Author: Todd Cochrane  
// Inputs: FLOAT y   
// Outputs: FLOAT y  
//   
// Desc: This function normalises the value of y for an   
// external application  
//---------------------------------------------------  
  
//---------------------------------------------------  
// Name: normaliseZ   
// Author: Todd Cochrane  
// Inputs: FLOAT z   
// Outputs: FLOAT z  
//   
// Desc: This function normalises the value of z for an   
// external application  
//---------------------------------------------------

//---------------------------------------------------  
// Name: transformHeadHands()   
// Author: Todd Cochrane  
// Inputs: NULL   
// Outputs: NULL  
//   
// Desc: This function transforms just the point data for the // head and hands into a string, ready to be sent // across the network.  
//---------------------------------------------------

## 6. Testing

For this feature we have taken into account the fact that the data is being transformed from the basic skeleton and bone orientations in to a form of data that we are able to pass over the server to the end client. For this we need to ensure that the transformed data populates the variable accordingly, therefore the tests have been designed to be implemented in the mData.cpp file ensuring that the end result is the correct data we require.

To test this we had to call the assert function within the file mData.cpp, to allow us to call the assert function we had to include the assert.h header file. By implementing this we were able to place the function calls throughout the application where they were necessary.

TEST 1

|  |  |  |  |
| --- | --- | --- | --- |
|  | Pass | Fail | Comments |
| mData.cpp – Confirming transformedData is not Null  (Line 121) | ✓ |  | This test failed due to an error in the assert code. More a failure due to developer than application. |
| mData.cpp – Testing to ensure skeleton is in the correct position  (Line 124) |  | 🗶 |

TEST 2

|  |  |  |  |
| --- | --- | --- | --- |
|  | Pass | Fail | Comments |
| mData.cpp – Confirming transformedData is not Null  (Line 121) | ✓ |  | All tests passed and the application runs correctly. |
| mData.cpp – Testing to ensure skeleton is in the correct position  (Line 124) | ✓ |  |

## 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.