|  |
| --- |
| Martin Rule, Lane Cotgrove, James Bayliss |
| Motion Project |
| Feature 2.3 Transform Movement Data from Stored Data |

|  |
| --- |
| Martin Rule & Lane Cotgrove  9/5/2012 |

Table of Contents

[1. Overview 2](#_Toc337416991)

[2. Feature team 2](#_Toc337416992)

[3. Sequence diagram 3](#_Toc337416993)

[4. Refined object model 4](#_Toc337416994)

[4.1 Overall object model 4](#_Toc337416995)

[5. Class and method prologues 5](#_Toc337416996)

[5.1 Class prologues 5](#_Toc337416997)

[5.2 Method prologues 5](#_Toc337416998)

[6. Testing 7](#_Toc337416999)

[7. Design inspection 8](#_Toc337417000)

[8. References 9](#_Toc337417001)

## 1. Overview

This feature is an internal function of the MData object. After a new segment of movement data is stored in an MData object it will be able to be transformed back into the Skeleton format ready to be drawn and used within the Avateering example.



## 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. During the Update process from the AvateeringXNA object, the MData object will request the transformedData string from the NetworkModel object.

2. The NetworkModel object returns the transformedData string that it has acquired from the Server.

3. The MData object then begins to take the string; split it into tokens based on the comma delimiter, and storing each index of the string array into the correct location within a Skeleton object located within MData.

## 4. Refined object model

## 4.1 Overall object model



## 5. Class and method prologues

## 5.1 Class prologues

//---------------------------------------------------

// @Name: MData

// @Author: Lane - PeePeeSpeed

// @Inputs: string networkString

// @Outputs: NULL

//

// @Desc: Class constructor for MData.

// Takes a string for the data

// to be transformed into a Skeleton.

//---------------------------------------------------

5.2 Method prologues

//---------------------------------------------------

// @Name: getTransMData

// @Author: Lane - PeePeeSpeed

// @Inputs: NULL

// @Outputs: string transformedData

//

// @Desc: This GET method returns the

// unedited transformedData string.

//---------------------------------------------------

//---------------------------------------------------

// @Name: setTransMData

// @Author: Lane - PeePeeSpeed

// @Inputs: string transformedData

// @Outputs: NULL

//

// @Desc: This SET method sets the MData

// transformedData string to an

// inputted string.

//---------------------------------------------------

//---------------------------------------------------

// @Name: transformMDataToSkeleton

// @Author: Lane - PeePeeSpeed

// @Inputs: NULL

// @Outputs: NULL

//

// @Desc: Takes the string received from

// the server and splits it into

// tokens based on the comma delimiter.

// Then sets each index in the splitString

// array into the correct value

// within a Microsoft.Kinect.Skeleton.

//---------------------------------------------------

//---------------------------------------------------

// @Name: writeToFile

// @Author: Lane - PeePeeSpeed

// @Inputs: NULL

// @Outputs: NULL

//

// @Desc: Writes a string of data to the

// AvateeringData.txt file. Used

// for testing only.

//---------------------------------------------------

## 6. Testing

For this feature, we will test the skeleton data to see if the transformed data is the same as the skeleton data after the transformation and the assignment of the data values back into the correct locations within the Skeleton object. To test the integrity of the Skeleton object, we will test various objects hash codes within the Skeleton’s data types to ensure they are different objects. We will also check to see if the data that was passed from the NetworkModel is the same as the data from the Kinect Client using the same techniques used in the previous Feature design documents.

## 7. Design inspection

Design inspection was performed by Martin Rule, Lane Cotgrove and James Bayliss.  
  
Advisor inspection was performed by Andrew Eales on the 27th September 2012.

## 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.