|  |
| --- |
| Martin Rule, Lane Cotgrove, James Bayliss |
| Motion Project |
| Feature 2.5 Display avatar from raw data |

|  |
| --- |
| Martin Rule & Lane Cotgrove  1/10/2012 |

Table of Contents

[1. Overview 2](#_Toc337417013)

[2. Feature team 2](#_Toc337417014)

[3. Sequence diagram 3](#_Toc337417015)

[4. Refined object model 4](#_Toc337417016)

[4.1 Overall object model 4](#_Toc337417017)

[5. Class and method prologues 5](#_Toc337417018)

[5.1 Method prologues 5](#_Toc337417019)

[6. Testing 6](#_Toc337417020)

[7. Design inspection 7](#_Toc337417021)

[8. References 8](#_Toc337417022)

## 1. Overview

This feature will be used to submit the Skeleton data to the AvateeringXNA object to be handled and rendered. It will use information located the MData object and using Avateering’s API to submit that information. We will do this by locating where the skeleton is normally created when a Kinect device is connected, removing this system and substituting our own system.

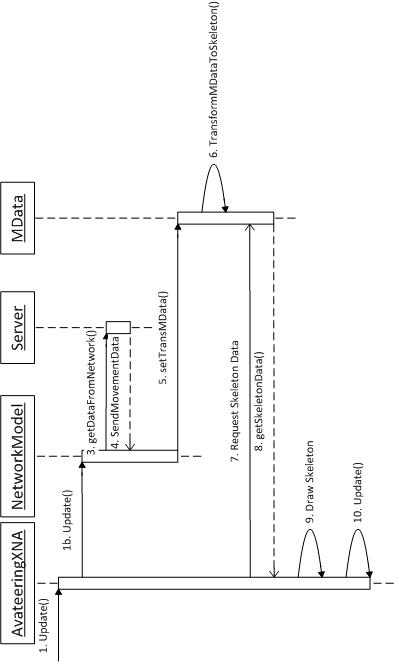


## 2. Feature team

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. At a regular interval, the AvateeringXNA object will Update, it will send a notification to the NetworkModel.

2. The NetworkModel object will then contact the registered server and request the next section of movement data from the queue.

3. The next section of movement data from the queue is then returned to the NetworkModel object.

4. The NetworkModel object then stores this new movement data segment into a local string.

5. The MData object requests the transformedData string from the NetworkModel object.

6. 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.

7. The AvateeringXNA object will then request the Skeleton data recreated within the MData object.

8. The MData object returns the Skeleton data to the AvateeringXNA object.

9. The AvateeringXNA object then stores the Skeleton into a new Skeleton and proceeds to draw a new frame with the new Skeleton data within it.

10. The AvateeringXNA object calls its own Update method to begin the process again.

## 4. Refined object model

## 4.1 Overall object model



## 5. Class and method prologues

## 5.1 Method prologues

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

// @Name: Update

// @Author: Microsoft Corporation

// @Inputs: GameTime

// @Outputs:

//

// @Desc: This function updates the application

// to display the most current data

// available. Our additions involve

// the MData object receiving and

// transforming the string of data

// from the NetworkModel and then

// passing it to a Skeleton to be

// drawn by the Avateering Application.

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

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

// @Name: getSkeletonData

// @Author: Lane - PeePeeSpeed

// @Inputs: NULL

// @Outputs: Microsoft.Kinect.Skeleton skeleton

//

// @Desc: This GET method returns a completed

// Skeleton ready to be drawn.

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

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

// @Name: getSkeletonIndex

// @Author: Lane - PeePeeSpeed

// @Inputs: NULL

// @Outputs: int skeletonIndex

//

// @Desc: This GET method returns the index

// of the Skeleton. (0 - 5).

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

## 6. Testing

For this feature, we will test the system by examining sections of data from the string and comparing them to the same data that will be located within the Skeleton, this will ensure that the data that is being passed into the Skeleton and drawn with the same unmodified data from the server.

## 7. Design inspection

Design inspection was performed by Martin Rule, Lane Cotgrove and James Bayliss.  
  
Advisor inspection was performed on the 4th of October 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.