**< WEEKLY REPORT FOR WEEK 3 >**

Name: Oh SuJin

Project: Multimodal Sensor Interfacing, Acquisition and Visualization

**I) Project Work Summary**

**Finished:**

* Data acquisition from leap motion; what types of data are available in what languages and which one to choose for iFarm.
* Study available finger/hand exercise and try to record using existing leap motion data recording application to determine whether it's possible to detect and recognize those exercises.
* Communication between Leap motion and Arduino: WebSocket vs Serial communication

**Ongoing:**

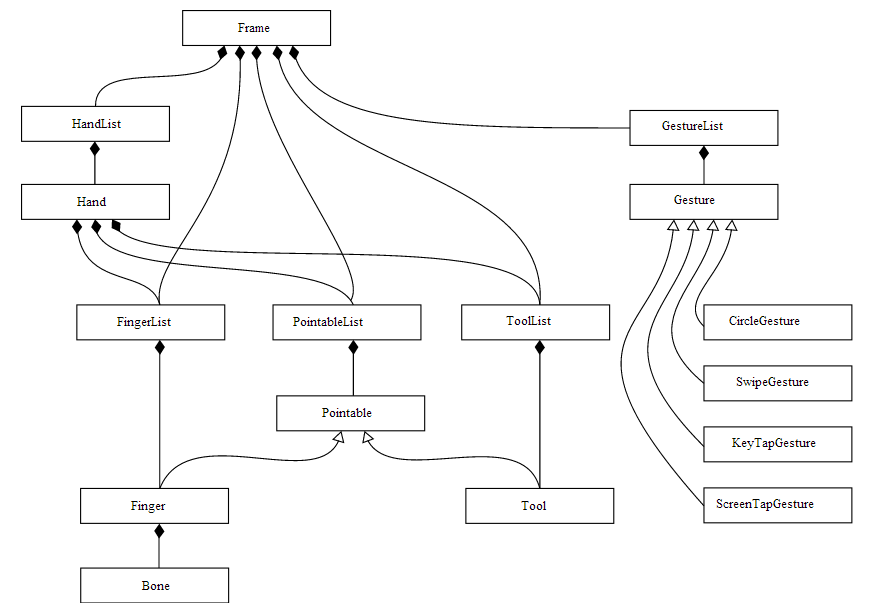
* How to store such data, in

**II) Tasks Assigned**

**III) Detailed Activities / Accomplishments**

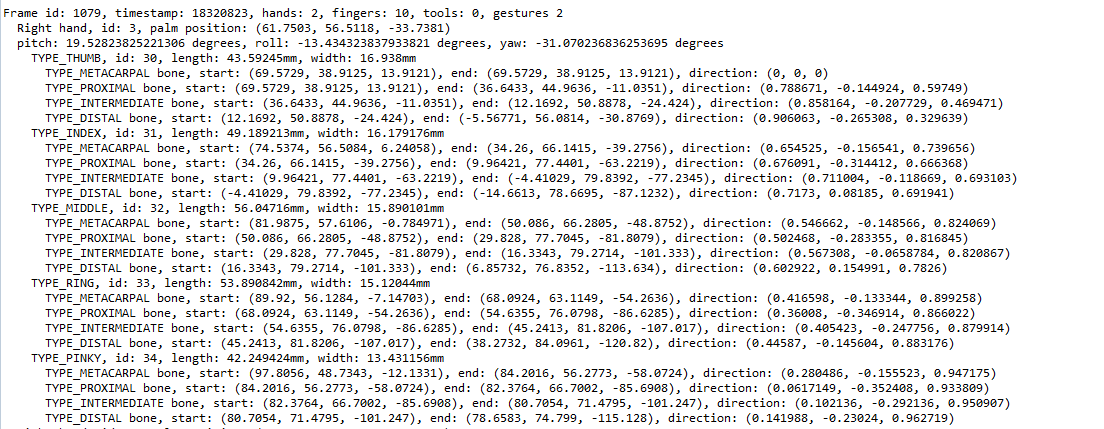
**DAY 1:**

* Understand basic application flow in Leap Motion. Study how each class and method are related to each other to get bigger picture of how Leap motion application works.



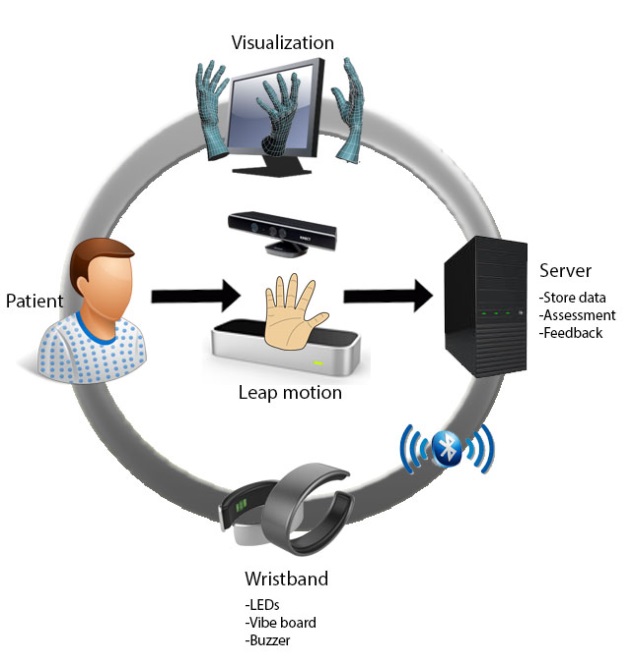
**DAY 2:**

* Managed to attain all the necessary tracking information from Leap Motion.



**DAY 3:**

* Design a ppt slide for iFarm presentation. I came up with a simple diagram which helps not only the therapist but also the other group members to grasp better the big picture of iFarm application.



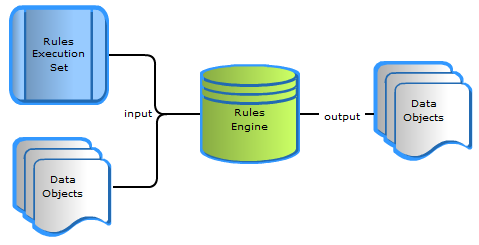
* I intend to combine face tracking as well as voice recognition feature for hand rehab application. In order to do so, I study samples for face tracking and voice recognition as well as some basic visual C# examples to gain better understanding of how MS Visual Studio 2012 works and provides numerous features for C# coding.

**DAY 4 &5:**

* I start my research on Rule Engines to design an assessment unit for iFarm.
* A rule engine may be viewed as a sophisticated if/then statement interpreter. The **if** portions of rules contain conditions such as account.getMoney() < 0. The **then**portions of rules contain actions such a sendWarning(account).

if (account.getMoney() < 0)  
sendWarning(account);

The inputs to a rule engine are a **rule execution set** and some **data objects**. The outputs from a rule engine are determined by the inputs and may include the original input data objects with possible modifications, new data objects, and side effects such as sendMail ('Thank you for shopping').



Using Rules Engine adds several advantages to our project such as

* **Greater flexibility:** Keeping our rules into a Knowledge base let us adapt easily our decisions when they are changing.
* **Easier to grasp:** Rules are easier to understand than procedural code like Java and C# so they can be effectively used to bridge the gap between therapist and us developers.
* **Reduced complexity:**Rules can handle increasing complexity because they use a consistent representation of hand rehab exercise rules.
* **Reusability:** Traditional procedural code often imposes unnecessary variations of base rules which are therefore more difficult to reuse in other contexts.
* [**Drools**](http://www.jboss.org/drools)is an Object-Oriented Rule Engine for Java. Drools is an augmented implementation of Forgy's Rete algorithm tailored for the Java language. Adapting Rete to an object-oriented interface allows for more natural expression of business rules with regards to business objects. More importantly, Drools provides for declarative logic programming and is flexible enough to match the semantics of problem domain. The various parts that compose a rule may be extended to lend domain-specific sense to the rule.

**<How is JBoss Drools is made up>**

Drools is split into two main parts: **Authoring**and **Runtime**.

* The **Authoring**process involves the creation Rules files (.DRL) which contain the rules which are fed into a parser. The parser checks for correctly syntax of the rules and produces an intermediate structure that "describes" the rules. This is then passed to the Package Builder which produces Packagesand undertakes any code generation and compilation that is necessary for the creation of the Package.
* On the other hand, the RuleBase is a runtime component made up of one or more Packages. A RuleBase can instantiate one or more WorkingMemories at any time.  
  The Working Memory consists of a number of sub components, including Working Memory Event Support, Truth Maintenance System, Agenda and Agenda Event Support.
* The **Working memory** is a key point of the Drools engine: it's here that Facts are inserted.
* Facts are plain Java classes which rely on the Java Bean pattern( your Java beans from your application). Facts are asserted into the Working Memory where they may then be modified or retracted.
* When facts are asserted into the working memory, it will result in one or more rules being concurrently true and scheduled for execution by the Agenda - we start with a fact, it propagates and we end in a conclusion. This method of execution for a Production Rule Systems is called **Forward Chaining -**it's depicted in this picture:

