Revision Sign-off

By signing the following, the team member asserts that he/she has read all changes since the last revision and has, to the best of his/her knowledge, found the information contained herein to be accurate, relevant, and free of typographical error.

Name	Signature	Date
Josh Alvord		
Alex Grosso		
Jose Marquez		
Sneha Popley		
Phillip Stromberg		
Ford Wesner		

Revision History

The following is a history of revisions of this document.

Date	**Reason For Changes**	**Version**
11/03/2009	Initial draft	1.0

Table of Contents

1. Introduction

1.1 Purpose

This document describes the requirements, both functional and non-functional, of the Appropriate Gesture Project version 1.0 developed by Team Better Recognize.

1.2 Document Conventions

The FROG product is divided into four primary modes and as such, the section of this document dealing with functional requirements is also divided in to four modes (with one additional section covering requirements common of all modes). This document includes a *Definition of Terms* section so the reader can quickly reference terms they are unfamiliar with.

1.3 Intended Audience and Reading Suggestions

This specifications document is intended for the customer and sponsor of this project. It ensures that the customer and Team Better Recognize are better synchronized and share the same vision.

1.4 Product Scope

The Appropriate Gesture Project has as its goal the development of a device and platform-independent gesture recognition software. In its initial release, Appropriate Gesture will be packaged with a plug-in for Sun SPOTs. Additional plug-ins will be written later. There will also be a demo program for testing the capabilities of the project.

1.5 References

Name	Website	Description
IEEE SRS Template	http://www.ieee.org	The style guidelines followed by this document.
Wiigee Project	http://www.wiigee.org	An open source gesture recognition project that Appropriate Gesture borrows from.

2. Definition of Terms

JSR-82	Java API for Bluetooth communication. There are several implementations of JSR-82. The BlueCove project is an open source and well-developed option for those wanting to use a variety of different Bluetooth stacks and operating systems.
Hidden Markov Model	A doubly stochastic (as opposed to deterministic) math model being used to represent a gesture. Constructed HMMs are then used in recognition. “A statistical model in which the system being modeled is assumed to be a Markov process with unobserved state.”
K-means Clustering	Method of cluster analysis which aims to partition n observations into k clusters in which each observation is clustered with the nearest mean.
Sun SPOT	Sun Small Programmable Object Technology: The wireless motes developed by Sun Microsystems used as the first plug-in for Appropriate Gesture. Contains a 180MHz 32-bit processor, 512K RAM, and a variety of sensors including a three-axis accelerometer.

3. Overall Description

3.1 Product Perspective

Appropriate Gesture is a self contained gesture recognition system.

3.2 Product Functions

The FROG Project will have four main modes. These are: Training, Recognition, Evaluation, and Demo. The training mode will allow the user to train gestures for later use. The recognition mode will allow the user to load previously trained gestures and adjust the inner-workings of the system to see how it affects accuracy. The evaluate mode allows the user to see accuracy and other read-outs to diagnose issues with their hardware or the FROG system itself. The demo mode is a simple game to demonstrate the capabilities of the system.

3.3 User Classes and Characteristics

This product is being developed for use in an experimental, academic environment. This product is designed for use by those fairly experienced with gesture recognition already. However, the demo is meant to be a fun and lighthearted way for anyone to use the system.

3.4 Operating Environment

The FROG Project was design to operate with the following software installed

- Windows XP or later, Mac OSX, or Linux
- Java Runtime Environment 6.0 or later
- Sun SPOT SDK (for the driver; if using a Sun SPOT as a controller)

3.5 Design and Implementation Constraints

Time Constraint

Limited by academic school year (ending in May 11, 2010)

Mobile Device Limitations

J2ME library on Sun SPOTs

Data communication between mobile device and host may be limited

Hardware Limitations

System developed on older machines. This may cause performance issues

3.6 User Documentation

3.7 Assumptions and Dependencies

stuff

4. External Interface Requirements

4.1 User Interfaces

Jose could you please fill out this section?

4.2 Software Interfaces

The project shall have a plug-in based system for adding support for additional hardware. These plug-ins shall contain all device-specific code and output vector data collected from the device. The plug-ins shall be derived from a common interface in the FROG software.

Since it is written in Java, FROG will have an obvious need to interact with a Java Virtual Machine Standard Edition 6.0 or later. Additional JVMs (such as ME or EE), if required, should be handled by plug-ins.

5. System Modes

5.1 Training Mode

In this mode, the user records gestures, names them, and saves them to be used later in other modes.

5.1.1 TRA-01

The system shall limit the number of connected devices to *one* during training mode. This is to ensure that there is no ambiguity as to what device is currently feeding gestures to the training system.

5.1.2 TRA-02

The system will provide the user with the ability to save and load training sessions for reuse. The file the system creates will be platform independent and contain not only the gesture data but the user's name, comments (for describing what significance that particular session has), and what device plug-in was being utilized.

5.1.3 TRA-03

The system shall provide a user with an intuitive method of training gestures into the system. These gestures may be represented by either a word and/or a picture or illustration.

5.1.4 TRA-04

The system shall wait until a button (determined by the plug-in) on the device is being held before collecting *any* incoming accelerometer data and immediately cease upon the button's release.

5.1.5 TRA-05

The system will allow the user to load an existing training session either to add additional gestures to the file or to delete gestures permanently from the file.

5.2 Recognition Mode

In this mode, the user loads a previously saved training session

5.2.1 REC-01

The system shall be able to connect and recognize the gestures of up to four devices simultaneously.

5.2.2 REC-02

The system will allow each user/device connected to load a training session library for evaluation.

5.2.3 REC-03

The system shall provide feedback *to each user* when a gesture is not recognized or, if it is recognized, report to the user what gesture it was.

5.3 Evaluation Mode

Some evaluation mode description goes here

5.3.1 EVA-01

The system will allow only one device to connect at a time.

5.3.2 EVA-02

The system shall provide real-time feedback of accelerometer and other relevant data for the purposes of diagnosing the hardware, the plug-in, and the system itself.

5.3.3 EVA-03

The system shall contain a standard set of gestures, specifically* a square, a circle, and a "Z", which can be used as test gestures for the user to perform to evaluate performance.

5.3.4 EVA-04

The system shall provide the following non-exhaustive* list of data in the console window/pane:

- number of vectors in last training instance
- processing time in milliseconds of each operation
- success or failure of recognition of last training instance
- current memory being used by classifier

5.4 Demo Mode

5.4.1 DEM-01

The system shall allow up to four devices to connect and play.

5.4.2 DEM-02

The system shall contain at least one demo program to better showcase the abilities of the underlying recognition system

5.4.3 DEM-03

The system shall keep track of each user's accuracy for evaluation of the system's (or perhaps the users') performance.

5.4.4 DEM-04

The system shall require that each user have already trained the gestures needed for the demo. The user's currently loaded training session file must contain gestures that share the same names as defined by the demo. A message to this effect must be displayed to the user so that he/she may go back to training mode and correct the situation.

5.5 Requirements for All Modes

5.5.1 GEN-01

The system shall *not* contain support internally for *any* device. Support for any device shall be incorporated into a corresponding plug-in for the device and then may be added or removed from the system via a GUI option menu.

6. Non-functional Requirements

6.1 Performance Requirements

6.1.1 PR-01

Communication between host and device shall be fast enough to support up to four connected devices

6.1.2 PR-02

The speed at which a gesture is performed shall not exceed 10μs per HMM

6.1.3 PR-03

Code written for the Sun SPOT (i.e. for filtering) must not hinder its ability to process accelerometer and radio transmission/reception data. Hinder is defined as causing the threads associated with the above activities to be skipped for more than 1 period.

6.2 Safety Requirements

6.2.1 SR-01

Accelerometer-enabled mobile devices must be used with attention to surroundings. The user must not allow the accelerometer to become airborne, potentially causing injury or damage.

6.3 Software Quality Requirements

6.3.1 SQR-01

The FROG Project will be able to recognize gestures with an 80% or better accuracy.

6.3.2 SQR-02

Plug-ins can be used to extend the versatility of the software.

6.3.3 SQR-03

FROG is a multi-platform framework.

6.3.4 SQR-04

7. Other Requirements

Appendix A: Use-case Model

Appendix B: User Interface Prototype

Appendix C: Architecture

Appendix D: Algorithms

Forward Algorithm:
The forward algorithm is designed for computing the probability of observing a certain sequence of observations being emitted from an HMM. In other words, it deals with the problem of recognition. Its viability comes from its marked increase in efficiency as compared to that of a brute force approach. A brute force approach would involve traversing every possible state path and combining each path's probability of producing the sequence. The forward algorithm, on the other hand, utilizes the Markov property of an HMM to drastically improve this process. The algorithm relies on calculating partial probabilities at each time step in the observation sequence (that is, consider each observation in the sequence as arriving at a discrete time t corresponding to the place it occurs in the sequence). We define the partial probabilities recursively (using the notation defined in Section 2):

$$\lambda = (S,\ O,\ a,\ b,\ \pi)$$
$$\alpha_{t+1}(j) = b[j][o_{t+1}] * \Sigma_{i=1}^{n} \alpha_t(i) * a[i][j]$$
$$\alpha_1(i) = \pi[i] * b[i][o_1]$$

The variable $\mathbf{o_t}$ represents the observation occurring at time t. And so the algorithm is based on initializing the $\boldsymbol{\alpha_1(i)}$ according to the initial probability multiplied by the emission probability of the first symbol in the observation sequence. Then $\boldsymbol{\alpha_{t+1}(j)}$ is the emission probability for the next observation multiplied by the sum of all the previous partial probabilities multiplied by the probability of transitioning from each state to the new state j. The recursive computation of this value can yield the probability of observing any sequence by simply summing up over the $\boldsymbol{\alpha_T(i)}$ where T is the total number of observations in the sequence (that is, sum over the last set of partial probabilities computed).

Backward Algorithm
The backward algorithm is based on exactly the same premises as the forward algorithm. It is

also utilized for calculating the probability of an observed sequence from an HMM. This algorithm, however, calculates probabilities, as suggested by its name, starting from the last observation and working backward. The algorithm is defined recursively as:

$$\beta_T(i) = 1$$
$$\beta_{t-1}(j) = \Sigma_{i=1}^{n} \beta_t(i) * a[j][i] * b[i][o_t]$$

I will leave out verbose description of the formula as it is exactly derivative of the forward algorithm above, only now using the β function.

Baum-Welch Algorithm
The Baum-Welch algorithm deals instead with the training problem. The Baum-Welch is an expectation maximization algorithm that utilizes the forward and backward algorithms. The algorithm is designed to optimize the parameters of an HMM so as to best model given training sequences. Thus it deals with maximizing the conditional probability of an observation sequence occurring given an HMM (to be optimized). The algorithm is only capable of carrying out local optimization, however, so there is no guarantee of the truly optimal HMM, which would require knowledge of a global optimum. Here let **a** again be our transition matrix, but use notation **$e_i(o_k)$** to represent the emission probability for observation k from state i. Then formally stated the Baum-Welch algorithm is:

```
Algorithm: Baum Welch

Input:
   A set of observed sequences, $O^1, O^2, \ldots$
Initialization:
   Select arbitrary model parameters, $\lambda' = a_{ij}, e_i()$.
         score = $\sum_d P(O^d|\lambda')$.
   Repeat
   {
      $\lambda = \lambda'$, $S = S'$
      For each sequence, $O^d$,
      {
         /* Calculate ``probable paths'' $Q^d = q_1^d, q_2^d, \ldots$ */
         Calculate $\alpha(t, i)$ for $O^d$ using the Forward algorithm.
         Calculate $\beta(t, i)$ for $O^d$ using the Backward algorithm.
         Calculate the contribution of $O^d$ to $A$ using (1).
         Calculate the contribution of $O^d$ to $E$ using (2).
      }
      $a_{ij} = \frac{A_{ij}}{\sum_l A_{il}}$
      $e_i(\sigma) = \frac{E_i(\sigma)}{\sum_\tau E_i(\tau)}$
      score = $\sum_d P(O^d|a_{ij}, e_i())$.
   }
   Until (the change in score is less than some predefined threshold.)
```

Where:

$$A_{ij} = \sum_d \frac{1}{P(O^d)} \sum_t \alpha(t,i)\, a_{ij}\, e_i(O^d_{t+1})\, \beta(t+1,i)$$

$$E_i(\sigma) = \sum_d \frac{1}{P(O^d)} \sum_{\{t|O^d_t=\sigma\}} \alpha(t,i)\, \beta(t,i).$$

O^d is the sequence of d observations of the complete observation sequence. α and β refer just as they did above to the respective forward/backward algorithm partial probabilities. And so with each iteration the HMM model parameters are modified towards a local optimum, training the model to the sequences provided.

Algorithm Definition Source: http://www.cs.cmu.edu/~durand/03-711/2006/Lectures/hmm-bw.pdf