ISU VRAC TACTILE VEST

Team May14-23 : Ben Andry, Garrett Phelps, Ryan Haack, Jacob Cramer, and Cyle Dawson
Client/Advisor: Dr. Stephen Gilbert

Last Edited on **11/14/13**	**Garrett Phelps**	**Version 1.3**

Version 0.1 | Ben Andry | 09/10/2013
-Initial Document Creation
Version 0.2 | Garrett Phelps, Ryan Haack | 09/19/2013
-Rough draft of Problem statement
Version 0.3 | Garrett Phelps | 09/24/2013
-Rough draft of UI description + concept sketch
Version 0.4 | Ryan Haack | 09/27/2013
-Added list of deliverables and requirements.
Version 0.5 | Garrett Phelps | 10/07/2013
-Added risks and mitigation section
Version 0.6.1 | Ryan Haack | 10/09/2013
-Added Market/Product study, Operating Environment, Assumptions and limitations**,** Definition of terms, Table of Contents, Resources
-Updated User Requirements, added organization and formatting to project plan
Version 0.6.2 | Garrett Phelps | 10/09/2013
-Updated Deliverables, User Interface Description, Resources, Definitions of terms, organization to project plan
Version 0.7 | Ryan Haack, Jacob Cramer, Ben Andry | 10/10/2013
-Added Milestones, Reference to WBS, System Block Diagram, System Description
Version 1.0 | Ryan Haack, Jacob Cramer, Ben Andry | 10/11/2013
-Added Appendix A and checked for errors.
Version 1.1| Ryan Haack | 10/23/2013
-Moved Table of Contents, resized pictures in concept sketch, added to 1.3.2, 1.4, 3.3.3 from Gilbert's comments
Version 1.2| Ryan Haack | 11/7/2013
-Updates to 1.3 and Work Plan
Version 1.3| Garrett Phelps | 11/14/2013
-Updates to the concept sketch and the UI description
Version 1.4| Garrett Phelps | 04/18/2014
-Updates to UI and adding OSG information. Replaced instances of tactor with tactor

Table of Contents

1) Project Breakdown

1.1 Problem Statement

The VRAC is currently receiving funding from the United States Army for a virtual reality training simulator. The MIRAGE a mixed-reality research lab fitted with IR sensors and a fully functional game engine. Applicants can use this simulator to experience combat simulations at a much lower cost and setup time than setting up an environment with paid actors.

This project is to develop an API for communication to off-the-shelf vibrating tactors on tactile vests that applicants would wear within the MIRAGE. These tactors vibrate whenever an applicant is "shot" within the simulation. The tactors could also be used as communication for navigation. Our main task is to set up an API and UI configuration so the tactors can be placed anywhere on the vest, which will then be able to identify the location of the tactor on the body. The API will be used to send signals to these tactors from a central command *i.e buzz shoulder tactor1*. The API will also need the ability to send multiple signals with predefined patterns and must be generic enough so that it can be easily called from any piece of software as a type of plug-in.

1.2 Operating Environment

The primary operating environment for our program will be Linux but should be compatible with Windows. We will create a serial connection to the tactors and write our API using C++. The user interface will be coded using the Qt application to help us achieve our cross compatibility goal. Our API will be the communication between the MIRAGE game engine and the tactile vest.   Our API will be fully compatible with any application in need of serial connection to a device.

1.3 Assumptions and Limitations

1.3.1 Assumptions

- VRAC has set up a git repo
- tactors, transmitter, and military vest will be provided
- Simultaneous signals can be sent with better hardware
- tactors must be configured before use

1.3.2 Limitations

- Vibration strength of tactors
- Hardware restrictions for simultaneous signals

- Operating range of tactor approximately ¼ mile
- Battery life of tactor approximately 48 hours
- Export control regulations limit us to showing source code to only US citizens
- Final project deadline is May 2014

1.4 Market/Product Study

There have been previous user studies of a tactile vest. There was a prototype of a vest with a solenoid on the chest. It was not practical because of dangerous amount of battery charge the user had to strap to his/her back and the extensive delay to recharge the solenoid mechanism. Another idea were arduinos but they aren't intuitive or cost effective.

The benefits of using tactors are the cost, safety and efficiency. The tactors can be attached to a vest and easily replaced if one breaks during regular use. One major concern with the tactors is the vibration strength. When trying to simulate being shot, there should probably be more pain or sensation than a simple vibration. We need to find a way to either strengthen the tactors or use multiple tactors in one area to "double" or "triple" the vibration.

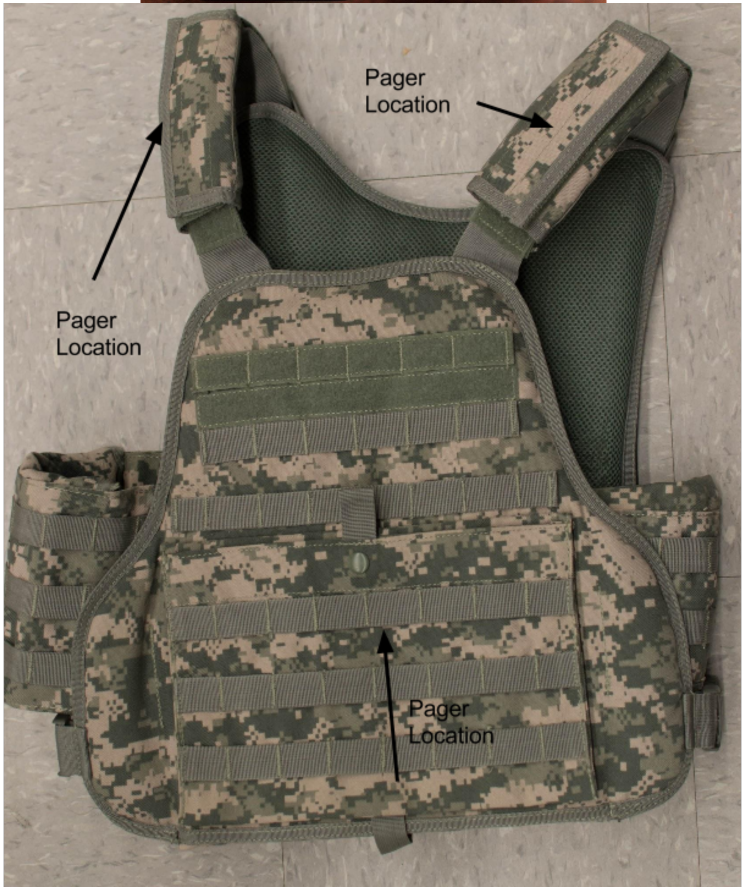
One major concern of this project is the ability to signal two tactors simultaneously. There has been one solution to program an identical tactor ID to all tactors that you want to page at the same time. We would like to find a solution that does not restrict us to programming the same IDs because these tactors can only be programmed within 20 seconds of being removed from the charger. This would also limit us because we would have to predetermine which tactors we want to have the same IDs and then those tactors are the only ones with the ability to be paged simultaneously.

There have been various other studies that we have discussed with our advisor. Vibrators can be used in watches to treat "Freezing" for people diagnosed with Parkinson's. "Freezing" refers to a person who suddenly stops and feels that they are frozen in place. A simple vibration can wake these people up. We also talked about the optimal frequencies for different body parts because different body parts have optimal vibration sensitivities.

2) Approach and Design

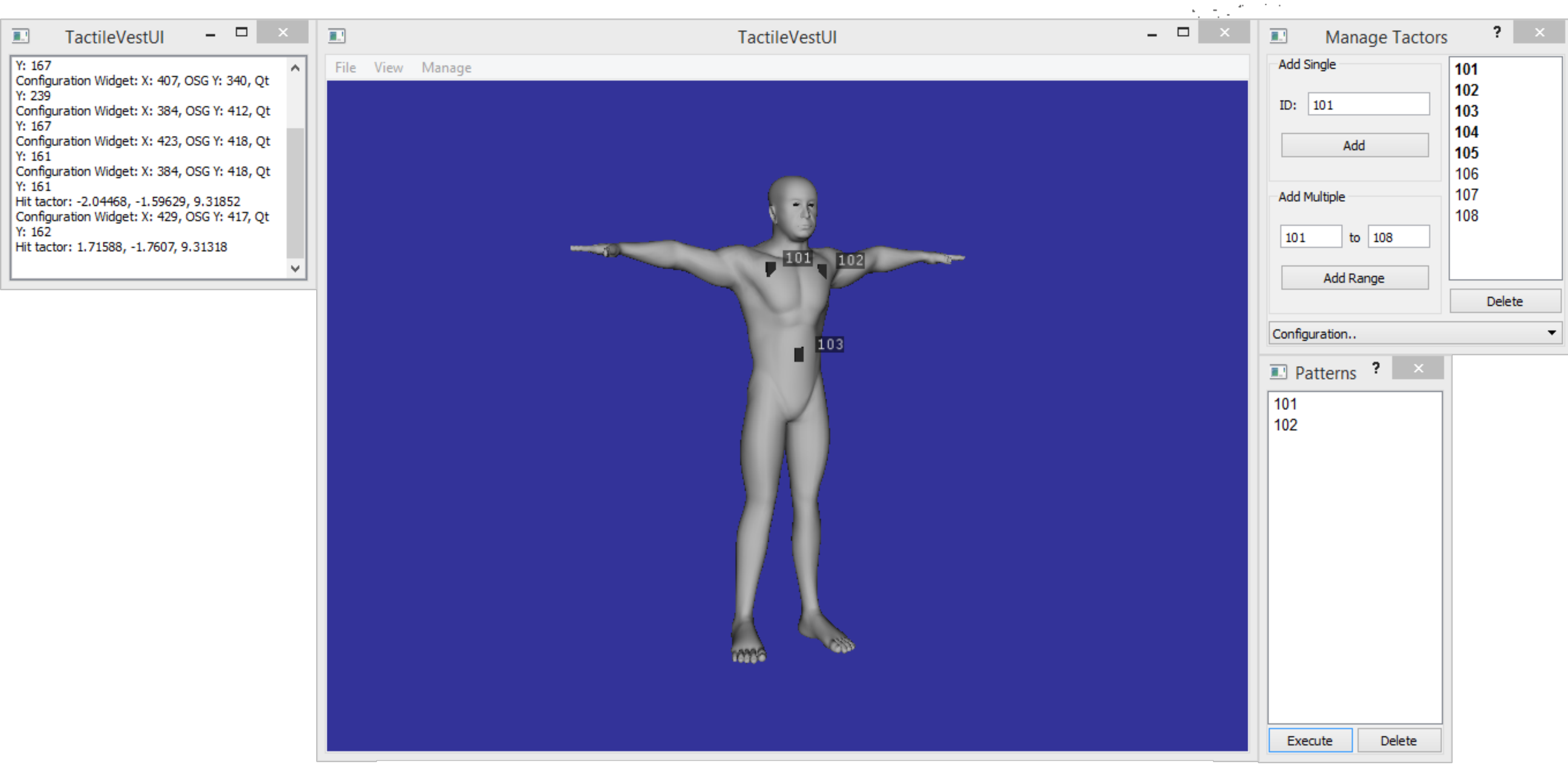
2.1 Concept Sketch

The spots to place the tactors on the vest will be the shoulders, stomach and also on the back. Multiple tactors can be placed in these spots to further increase the intensity of the vibrations if necessary. The vest has velcro for each location for the tactors to stay in place.



2.2 User Interface Description

The user interface will be developed by using the cross platform Qt framework. The main screen a user will see is a human model along with the Log, patterns, and manage tactos window. Users will be allowed to add however many tactors they need and also to assign an ID to the tactors within the manage tactors. Once the tactors are created the user has the option of right clicking anywhere on the model and selecting what tactor to go at that location. Once this is done a tactor object is added to the model along with a label indicating its ID. Our first iteration (Version 1) will have pre-defined location on the body to place tactors, but in the future (Version 2) we hope to be able to allow the user to place the tactors anywhere on the body. We also will have the ability to allow the user to load on the fly configurations that they have previously saved.



```
Y: 167
Configuration Widget: X: 407, OSG Y: 340, Qt
Y: 239
Configuration Widget: X: 384, OSG Y: 412, Qt
Y: 167
Configuration Widget: X: 423, OSG Y: 418, Qt
Y: 161
Configuration Widget: X: 384, OSG Y: 418, Qt
Y: 161
Hit tactor: -2.04468, -1.59629, 9.31852
Configuration Widget: X: 429, OSG Y: 417, Qt
Y: 162
Hit tactor: 1.71588, -1.7607, 9.31318
```

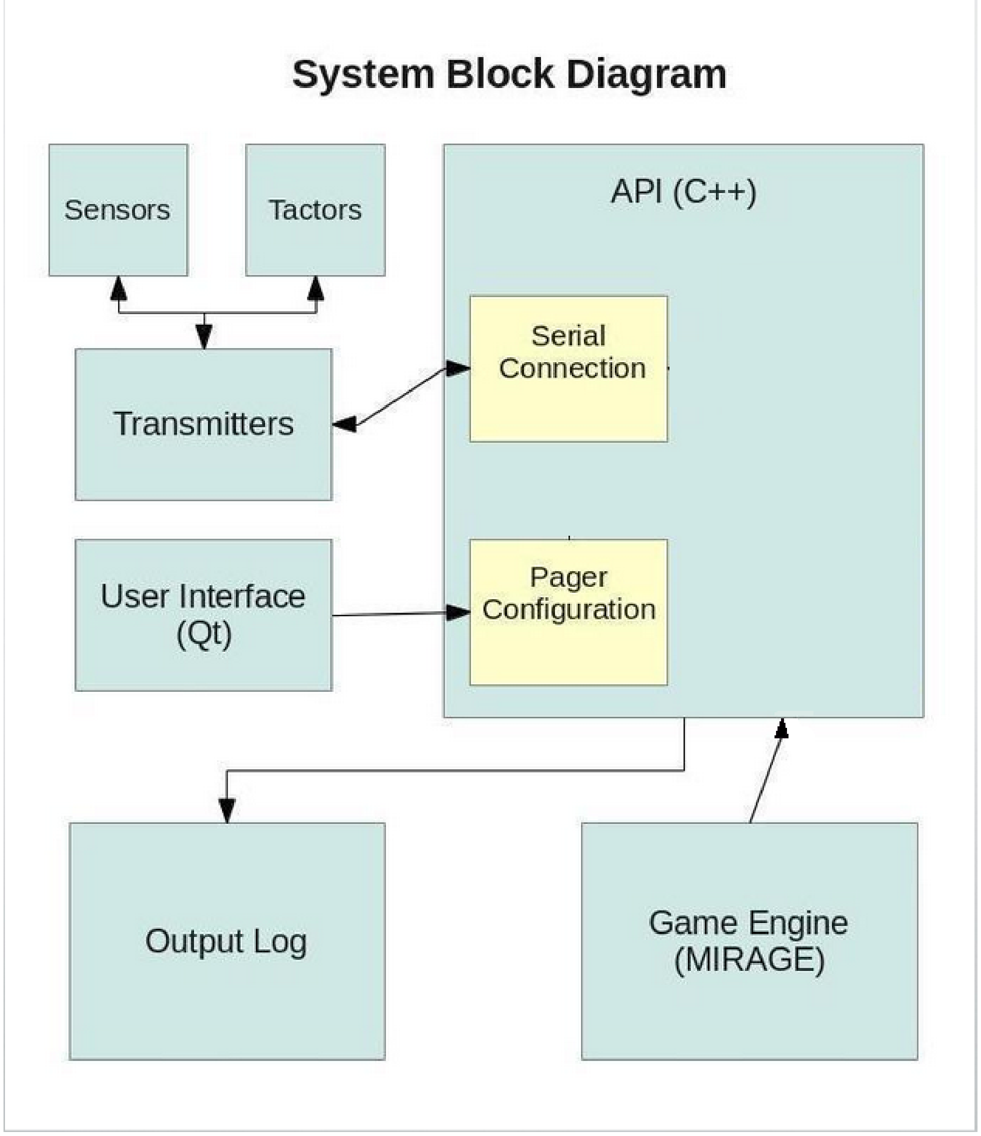
ID: 101

101 to 108

101
102
103
104
105
106
107
108

101
102

2.3 System Block Diagram



2.4 System Description

2.4.1 Sensors - Obtains data from the subject and sends back to transmitter
2.4.2 Tactors - Responds to data from transmitter (ex. tactor)
2.4.3 Transmitters - Sends and receives data through serial connection
2.4.4 API - Specifies interaction of the game engine, user interface, and transmitters
2.4.5 Serial Connection - Takes the data from the interpreter, converts the data into a serial format and sends it to the transmitter
2.4.6 Interpreter - Integrates data from the game engine into the tactor configuration and sends it to serial connection
2.4.7 tactor Configuration - Stores the layout of the tactors as received from the user

interface

2.4.8 Game Engine Communication - Receives data from the game engine and sends it to the interpreter

2.4.9 Game Engine - the MIRAGE environment

2.4.10 Output Log - A log of the events occurred and command given during the use of the API

3) Design Requirements

3.1 Functional Requirements

Version 1 (December 2013)

V1.1 User will be able to choose from predetermined locations for the tactors on the vest/body.

V1.2 User will be able to send commands with predefined buttons.

V1.3 User can change the intensity of the vibration for each tactor.

V1.4 User can test tactors currently assigned to body to ensure functionality

Version 2 (May 2014)

V2.1 User can place pagers to any appropriate location

V2.2 User can make custom patterns with UI

V2.3 User can make custom tactor layout and save custom tactor layout

V2.4 User can use other tactile attire (belt, wrist)

V2.5 Interface must be able to be “plugged” into any platform or system

3.2 Non-functional Requirements

3.2.1 Detailed documentation, every method declaration and class

3.2.2 Quick response time to signals sent to tactors

3.2.3 Use the fewest possible transmitters to keep price low

3.2.4 UI is simple enough to use without any knowledge of the code

3.3 Deliverables

3.3.1 One functional Tactile Vests that are able to be configured with the tactors

3.3.2 Each vest contains up to 8 tactors place throughout the vest/body

3.3.3 A generic and reusable API for serial communication to the tactors. The API will be used within the game engine in the MIRAGE or by other developers.

3.3.4 Intuitive UI for tactile vest configurations

3.3.4.1Configuring tactors anywhere on body (front/back)

3.3.4.2 Compatibility with multiple vests

3.3.5 User can use other tactile attire (belt, wrist)

4) Work Plan

4.1 Team Member's Tasks

Garrett - User Interface with Qt
Ryan - Qt communication with API, UI
Jacob - API Development
Ben - API Development
Cyle - System Architect, API

4.2 Schedule/Milestones

Over the next several months the VRAC Tactile Vest Project will partake in several milestones which will demonstrate progress and potential. Each milestone has an overall goal and several tasks that help in achieving this goal which will be tasks in the WBS (4.3).†

Milestones:

10/11- Project Plan V.1
10/21 - Simultaneous Signals Tests
10/24 - Design Document V.1
11/4 - Skeleton API, Skeleton User Interface
11/15 - Project Plan V.2
11/18 - Communication between UI and API to send a signal (Functioning prototype)
12/2 - Fully functioning prototype of one Tactile Vest
12/6 - Final Design Document and Project Plan
12/10 - Group Presentation
1/27 - Fully Working API where user can run predetermined patterns
2/10 - Functioning ability to create custom patterns with UI**
3/20 - Open Scene Graph integration
3/25 - User can use other tactile attire (belt, wrist, etc.)
4/30 - Finalize API and UI

† Milestones may be added or changed once 492 begins.

**Pending on hardware restrictions.

4.3 Work Breakdown Structure

See Appendix A.

4.4 Risks and Mitigation

4.4.1 Risk: The transmitter prohibits us from sending multiple signals at once which will fire off all tactors at once

Mitigation: Testing early can help us determine our hardware limitation so we can move on to other hardware options

4.4.2 Risk: The tactors do not provide enough vibrating power

Mitigation: A user study has already been conducted and although it has been pointed out to be a possible issue, we could look into modifying the tactor design so that it will vibrate at a greater intensity.

4.5 Resources

- Long Range Systems website
- Long Range Systems IT consultants
- 8 Long Range Systems service tactors(More available on request)
- 1 Long Range Systems T74USB Interface Transmitter(More available on request)
- MIRAGE and VRAC lab space
- Tailor for possible vest variations
- Clear Rubbermaid Storage Bin with White lid

Definition of Terms

VRAC	Virtual Reality Application Center

MIRAGE	Mixed Reality Adaptive Generalizable Environment
API	Application programming interface
Qt	A cross platform GUI application framework utilizing C++
Tactor	A piece of hardware to generate a tactile response.
Serial Connection	The process of sending data one bit at a time
tactor Layout	The location of tactors on the vest/body
tactor Pattern	The sequence of the tactors' vibration/lights
Open Scene Graph	Open source library for rendering 3D models with Qt support