Senior Project CIS 4911-U01

Multi-Touch and Mid-Air Framework

Visualizer

Requirements Document

Team Members:

Richard A. Lopez

April 30, 2015

Mentor:

Francisco Ortega

Professor:

Masoud Sadjadi

# Abstract

This document outlines the requirements of creating an API for the use of developers who want to create applications that use any type of input devices, focusing mainly on touch screens and mid-air inputs. Also covered on this document is the plan that we are to follow when developing this API, the functions that are necessary for the system to be functional, the requirements provided by the product owner, the breakdown of the work and the models for the system.

This document will go over the current systems in place that are not meeting the expectations that our API expects to meet. It will describe the purpose of our software system and the work that needs to be accomplished to meet the required functionalities. The project plan will delineate the organization of the project, the team member that will take part in the development of the API and the hardware and software requirements for the system to be implemented.

# Table of Contents

[Abstract 2](#_Toc417901232)

[Table of Contents 3](#_Toc417901233)

[1. Introduction 4](#_Toc417901234)

[1.1. Problem Definition 4](#_Toc417901235)

[1.2. Scope of system 4](#_Toc417901236)

[1.3. Terminology - Definitions, acronyms, and abbreviations 5](#_Toc417901237)

[1.4. Overview of document 5](#_Toc417901238)

[2. Current System 6](#_Toc417901239)

[3. Project Plan 7](#_Toc417901240)

[3.1. Project organization 7](#_Toc417901241)

[3.2. Work breakdown 7](#_Toc417901242)

[3.3. Cost Estimate 8](#_Toc417901243)

[4. Proposed System Requirements 9](#_Toc417901244)

[4.1. Functional Requirements 9](#_Toc417901245)

[4.2. Analysis of System Requirements 9](#_Toc417901246)

[4.2.1. Scenarios 9](#_Toc417901247)

[4.2.2. Use case model 13](#_Toc417901248)

[4.2.3. Static model 13](#_Toc417901249)

[4.2.4. Dynamic model 13](#_Toc417901250)

[5. Glossary 15](#_Toc417901251)

[6. Appendix 16](#_Toc417901252)

[6.1. Appendix A - Complete use cases 16](#_Toc417901253)

[6.2. Appendix B - Use case diagram using UML 31](#_Toc417901254)

[6.3. Appendix C - Static UML diagram 32](#_Toc417901255)

[6.4. Appendix D - Dynamic UML diagram 33](#_Toc417901256)

[6.5. Appendix E - User Interface designs. 39](#_Toc417901267)

[6.6. Appendix F - Diary of meeting and tasks. 39](#_Toc417901268)

[7. References 46](#_Toc417901269)

# Introduction

Input devices have advanced at an outstanding rate in the last few decades. The introduction of the mouse was one of the factors that brought the personal computer out of the technical user niche and into the regular consumer. Now touch input is in almost every cellphone and readily available to the general public. Newer input forms have been developed since such as multi touch screen, 3D mouse, midair tracking, and others. Up until now developers have had to code their respective games and applications for the individual input devices which make it a time consuming task and it limits the utilization of more than one input device in use at a time.

The Touch Air Motion Framework is trying to create a more encompassing input device library such that developers can simply choose their input devices and code their apps without having to deal with the individuality of each device.

## Problem Definition

Today’s market is flooded with different input devices and it has become a common problem for app and game developers to code for the individual input devices aside that to code for their own application or game. Most of the time the plugins and internal software of the input device is available for the developers to search through, learn and be able to apply the necessary characteristics that the developers needs the input devise to express. In the worst cases the information is not open source or there is not enough information around hindering the use of these input devices.

It has become cumbersome for some software developers to decide which input devices they can use out of the list that they really want to use. A framework that contains most of the input devices in the market will make is so that more developers can code for more out of the box applications, making the computer science field a more user friendly environment. A centralized input device library can make it so that new generations of developers and students can code for the low cost input devices and create the next generation of human computer interactions.

## Scope of system

The TAM framework for version 1.0 is expected to provide the functionalities delineated in this document. The functionalities or use cases not portrayed in this document are either expected in a future version or not in the scope of the system.

The system will implement a visualization function in version 1.0 to allow for the testing of input to output mapping, gesture recognition, and raw input testing. The system needs this visualizer in place before the rest of the framework in order to test the input and output recognition. Development of the visualizer requires that an input be directed to the visualizer and a mapping function to be in place (one to one is the default). Since the rest of the framework will be developed in later versions the input and map will be coded within the visualizer to immediately test the functionalities of such visualizer. In this version the input device selected will be a multi-touch screen with the ability to have up to ten different input events at the same time.

In version 1.0 of the TAM visualizer we will expect the user to be able to test if a touch screen is multi touch capable, the user shall be able to draw with multiple fingers on the screen simultaneously, replay the actions taken, map those actions to functions, save the actions and replay different actions from files.

## Terminology - Definitions, acronyms, and abbreviations

* API: Application Program Interface
* HCI: Human Computer Interaction
* TAM: Touch and Mid-Air Framework
* WIMP: Windows, Icons, Menus, Pointer
* UML (Unified Modeling Language)

## Overview of document

This document will go over the all the necessary parts to implement the TAM visualizer version 1.0. In chapter two we will go over the project plan for this version of the system, the organization, developers involved and their designated tasks, and the cost estimate to implement the system. In chapter four of this document we will go over the requirements that the system must follow as expected by the product owner, the analysis of these requirements, and the creation and refinement of use cases. The models that will represent how that system will be implemented and the initial interactions between the functionality. Chapter five will be a glossary of the terms used in this document. Chapter six is the appendix that will contain all of the images and tables that rea supplementary to this document.

# Current System

Currently there is no framework or API that provides an easy to use comprehensive collection of functions and control elements for multiple input devices. Some solutions have been created that allow coding for multiple input devices easily and fast, the problem is that most of this solutions are specific for some game engines or app engines, most are not open source therefore creating a blockade for those the biggest collection of developers which is the open source community. The only active open source solution available today is the Virtual Reality Peripheral Network (VRPN). The VRPN is “a set of classes within a library and a set of servers that are designed to implement a network-transparent interface between the application programs and a set of physical devices used in a virtual reality system.” (Taylor II)

The VRPN suffers from some problems that make it an unqualified candidate for solving the problem at hand. First as the name states this system uses a network in which one computer hosts a Virtual Reality (VR) station that controls the peripheral or input devices. This is problematic since many developers do not want to use a network connection and the speeds of data transfer between devices in a network are very slow when compared to the communications inside an individual computer system. A secondary problem is that the VRPN is codded in C language, this proposes a challenge to game developers that use different languages that provide better graphics and faster processing such as UNITY which is a C++ and C# gaming engine. It is easier to port a C++ framework to other languages such as JAVA and C than to go the other way around not to mention that you still have the versatility of many of the C low level performance.

# Project Plan

In this section we will go over the project plan, the organization of the tasks that will be needed to be completed for the TAM framework to be developed. The focus this semester is the visualization system, this system is an essential part of the TAM framework, allowing application developers to test the input devices in real time, record the inputs and replay them later on. We will also go over the people involved in this project as well as their roles, we will also go over the requirements to develop this system be them software or hardware. Finally we will go over the milestones, deliverables and tasks that will need to be completed this semester.

## Project organization

This section outlines the management and organization aspect of this project. The implementation of this project is broken down into various requirements depending on the features that this product will exhibit. Each group member was assigned a task along with the necessary deadlines needed to complete the project.

This semester the only team member is Richard A Lopez. He will be responsible for the development of the visualization interface that allows the developer to test touch event in a multi touch screen, replay the events, save them for future use and implementing mapping functions.

.

## Work breakdown

The software development methodology used for this semester will be Agile Software Development. This will be divided into five Scrums which is an Agile method that focuses on the organizational aspects of the software engineering process. It is currently one of the most popular Agile methods. (Sen. Proj. Class).

The Milestones for this semester are as follows:

Sprint 1, starts February 2nd and ends February 13th

Sprint 2, starts February 16th and ends February 27th

Sprint 3, starts March 2nd and ends March 20th

Sprint 4, starts March 23rd and ends April 3rd

Sprint 5, starts April 5th and ends April 17th

Each sprint represents a complete software development cycle of planning, developing, implementing and testing.

The Deliverables for this semester are:

Sprint review 1, February 16th

Sprint review 2, March 2nd

Sprint review 3, March 23rd

Sprint review 4, April 5th

Sprint review 5, April 20th

Final deliverable, April 30th

Final presentation, May 1st

A detailed schedule can be seen in Appendix A

## Cost Estimate

This project will be developed by only one team member with the counseling of a mentor. The time allotted for the project is ten weeks split into five sprints of two weeks each, this would mean that there will be around twenty hours of work for every week it comes out to 40 hours per sprint for a total of 200 hours of work. The cost for the system is therefore 200 hours of man power.

It is necessary for the development process to add more time for any unexpected problems, in this case this amount will be 50 hours allowing for every sprint to have ten hours extra(five a week) to allow for any problems or delays. The final documentation and project preparations are allowed within two more weeks, therefore adding another 40 total hours, therefore at the end the cost of this project will come out to 290 hours to complete version 1.0 of the Touch and Mid-Air framework.

# Proposed System Requirements

The system requirements are the necessary functionalities that the software system must provide for the designated user. These functionalities were obtained by talking with the product owner and going over the different scenarios that can happen whenever the user for the system interacts with it. These requirements are then refined and supplementary requirements are created to allow the coverage of as many different scenarios as possible.

## Functional Requirements

TAMUC01 The system shall subscribe to a windows service to receive touch events.

TAMUC02 The system shall allow a user to touch the screen with one or more fingers (Max 10) and draw on the screen.

TAMUC03 The system shall define the states of a touch event.

TAMUC04 The system shall use the X and Y coordinates and map them to a function.

TAMUC05 The system shall record input.

TAMUC06 The system shall replay the touch action in the order the action happened.

TAMUC07 The system shall allow for input to be saved to a file.

TAMUC08 The system shall read recorded input files and replay them.

TAMUC09 The system shall allow various sizes of touch visualization.

TAMUC10 The system shall allow the user to clear the screen and replay data at will.

TAMUC11 The system shall display all the information from the program to a debug monitor.

The nonfunctional requirements such as usability, reliability, performance and supportability are detailed for every system requirement in Appendix A.

## Analysis of System Requirements

In this section we will go over the system requirements that were collected from the product owner. We will look at the scenarios that a user might face and examine those to later for a use case model, object /class diagrams, and finally sequence diagrams.

### Scenarios

From the functional requirements collected we can derive scenarios that a user in our case app/game developer will encounter while using the TAM framework.

Scenario1-Touch

User: App Developer

1. The user starts the TAM Visualizer
2. The user touches the draw area
3. 2D drawing represents touch
4. The user touches the draw are with up to ten fingers
5. The user moves finger(s) across the draw area
6. 2D drawing displayed everywhere a finger touched

Scenario2-Touch Events

User: App Developer

1. The user starts the TAM Visualizer
2. The user touches the draw area
3. Touch event triggered
4. The user touches the draw are with up to ten fingers
5. Multi touch event triggered
6. The user moves finger(s) across the draw area
7. Moving event triggered
8. The user lift finger(s)
9. Lifted event triggered

Scenario3-X and Y Mapping

User: App Developer

1. The user starts the TAM Visualizer
2. The user touches the draw area
3. X and Y position are recorded and mapped to mapping function
4. The user touches the draw are with up to ten fingers
5. X and Y positions are recorded and mapped to mapping function
6. The user moves finger(s) across the draw area
7. X and Y positions are recorded and mapped to mapping function

Scenario4-Recording Input

User: App Developer

1. The user starts the TAM Visualizer
2. The user touches the draw area
3. Input recorded X, Y, ID, Time stamp
4. The user touches the draw are with up to ten fingers
5. Inputs recorded X, Y, ID, Time stamp
6. The user moves finger(s) across the draw area
7. Inputs recorded X, Y, ID, Time stamp

Scenario5-Replay

User: App Developer

1. The user starts the TAM Visualizer
2. The user touches the draw area
3. The user touches the draw are with up to ten fingers
4. The user moves finger(s) across the draw area
5. The user select playback
6. The events that the user did are replayed in the order they happened

Scenario6-Save

User: App Developer

1. The user starts the TAM Visualizer
2. The user touches the draw area
3. Input recorded X, Y, ID, Time stamp
4. The user touches the draw are with up to ten fingers
5. Inputs recorded X, Y, ID, Time stamp
6. The user moves finger(s) across the draw area
7. Inputs recorded X, Y, ID, Time stamp
8. The user selects save
9. All inputs are saved to a file

Scenario7-Replay from File

User: App Developer

1. The user starts the TAM Visualizer
2. The user touches the draw area
3. Input recorded X, Y, ID, Time stamp
4. The user touches the draw are with up to ten fingers
5. Inputs recorded X, Y, ID, Time stamp
6. The user moves finger(s) across the draw area
7. Inputs recorded X, Y, ID, Time stamp
8. The user selects open
9. The events that the user did on that file are replayed in the order they happened

Scenario8-Brush Size

User: App Developer

1. The user starts the TAM Visualizer
2. The user selects size
3. The user touches the screen
4. The size of the 2D drawing is the one chosen

Scenario9-Clear Screen

User: App Developer

1. The user starts the TAM Visualizer
2. The user touches the draw area
3. The user touches the draw are with up to ten fingers
4. The user moves finger(s) across the draw area
5. The user selects clear
6. The screen is cleared

### Use case model

Refer to Appendix B

The use case model for the TAM Visualizer contains the following elements:

User: In our system there are two users, one is the intended user for the application which is the app/game developer. The other user is the Operating System (in our case Windows), this is an actor that only reacts to events from the system and never triggers any events.

Use cases: The use cases that we can observe in the use case model are the result of the use cases defined in Appendix A.

Interactions: The interactions on this use case model are as follows:

Developer: associated to use cases Mapping, Clear screen, Draw, Playback, and Record Input. Non direct associations to manipulation, output and resize of output.

Operating System: Connection and Debug are associations, and subscription, interpretation and states are indirect associations.

### Static model

Refer to Appendix C

The Static model chosen for TAM is displayed as two class diagrams. The reason that two class diagrams are shown is to give a better reference on where the Visualization application fits within the bigger TAM framework. The second class diagram is the representation of the current TAM Visualizer. As you can observe we have the DrawArea and the MainWindow class, the DrawArea is where the majority of the work is being done, whenever the MainWindow receives an input event it calls the methods of the draw area to complete all of the functions that were described in the use cases.

For each class we have the class name in the first row, there are attributes listed in the second row, and functions listed on the third row.

### Dynamic model

Refer to Appendix D

The Dynamic models for TAM consist of sequence diagrams. These diagrams represent a timeline of the actions taken by the user and the system as well as the responses of the system for those actions. The diagram is read from left to right and from top to bottom. Time flows from the top down therefore every action or object at the same height exist at the same time.

For every use case there is a sequence diagram that follows the same timeline of actions and reactions, users and messages.

# Glossary

**Actor:** External entity that interacts with the system

**Class diagram:** describes the system in terms of classes, attributes (data items, instance variables), operations (member functions, methods) and their relationships. Class diagrams are used to represent object models during development conditions for a specified period of time

**Design pattern:** Provides a schema for refining the subsystems or components of a software system, or relations between them.

**Frequency:** Rating assigned to the piece of functionality that represents the level of invocations a particular piece of functionality will be subjected to.

**Functional requirements:** An area of functionality the system must have.

**Interface:** An abstraction that an entity provides of itself to the outside. Separates the methods of external communication from internal operation, and allows it to be internally modified without affecting the way outside entities interact with it, as well as provide multiple abstractions of itself. **Invariants:** is a predicate that is always true for all the instances of a class.

**Model:** An abstraction of a system aimed at simplifying the reasoning about the system by omitting irrelevant details.

**Nonfunctional requirements:** a constraint on the system

**Pattern:** Addresses a recurring design problem that arises in specific design situations, and presents a solution to it

**Persistent data:** Data that outlives a single execution of the system.

**Reliability:** Ability of a system or component to perform its required functions under state

**Risk:** An area of uncertainty that can lead to a deviation in the project plan, including failure of the project.

**Sequence diagram:** An interaction diagram that emphasizes the time-ordering of messages.

**Specification:** Describes the requested behavior of the software system.

**UML (Unified Modeling Language)** is a standard language for writing software blueprints. The UML may be used to visualize, specify, construct, and document the artifacts of a software-intensive system. (Booch et al. 1999)

**Usability:** Ease with which a user can learn to operate, prepare inputs for, and interpret outputs of a system or component.

**Use case model:** Represents the functionality of the system in terms of a sequence of interactions between an actor and the system.

**Use case:** General sequence of events that describe all possible actions between actor and the system for a given piece of functionality.

# Appendix

## Appendix A - Complete use cases

**Use Case ID:** TAMUC01 – Subscription

**Use Case Level:** High-Level

**Details:**

* **Actors:** Operating System, Developer
* **Pre-conditions:**
  1. The system is capable of multi touch input (Touch Screen)
* **Description:**
  1. Use case begins when the application is started.
  2. The system creates a listener service that subscribes to the touch events in the draw area.
  3. The use case ends when the system receives the subscription ticket.
* **Post-conditions:**
  1. The system is ready to start receiving touch events in the draw area.
* **Alternative Courses of Action:**

n/a

* **Exceptions:**

The system cannot connect to the touch event service.

* **Related Use Cases:**
  1. TAMUC02
  2. TAMUC03
  3. TAMUC04
  4. TAMUC05
  5. TAMUC07

**---------------------------------------------------------------------------------------------------------------------**

**Decision Support**

**Frequency:** On average, 1-10’000 touch events per second.

**Criticality:** High. The Framework cannot be tested if the touch input cannot be received.

**Risk:** Medium. The subscription service depends on the Windows operating system.

**---------------------------------------------------------------------------------------------------------------------**

**Constraints:**

* Usability: No Human User
* Reliability: The system shall recognize 10 inputs maximum and up to 10’000 different touch events per second.
* Performance: The input position should be within ±0.5 mm of the touch screen position.
* Supportability: The screens with multi-touch capabilities cannot exceed 10 fingers at the same time.
* Security: n/a

---------------------------------------------------------------------------------------------------------------------

**---------------------------------------------------------------------------------------------------------------------**

**Use Case ID:** TAMUC02 – Draw Action

**Use Case Level:** High-Level

**Details:**

* **Actors:** Developer
* **Pre-conditions:**
  1. The Application has been launched
  2. The screen is cleared or its started as a new blank canvas
* **Description:**
  1. Use case begins when the user touches the screen with 1-10 fingers inside the “drawing area”
  2. The system shall obtain the X and Y coordinates and use the mapping function selected (TAMUC04-Mapping)
  3. The system shall display a point the size which has been selected (TAMUC09– Resize) at the position designated by the Mapping function.
  4. The user continues the drawing without lifting the finger(s).
  5. The System continues to trace all of the points while simultaneously saving the points (TAMUC07-Record)
  6. The use case ends whenever the user lifts their finger(s) from the screen.
* **Post-conditions:**
  1. The screen allows for another draw gesture.
* **Alternative Courses of Action:**
  1. The user after lifting finger(s) touches the draw area again and continues drawing.
* **Exceptions:**

The user asks another person for them to draw at the same time as him/her; making it more than 10 inputs (the system will only draw original 10).

* **Related Use Cases:**
  1. TAMUC05
  2. TAMUC07
  3. TAMUC09

**---------------------------------------------------------------------------------------------------------------------**

**Decision Support**

**Frequency:** On average, 1-10’000 touch events per second.

**Criticality:** High. The Framework cannot be tested if 2D representation is not available.

**Risk:** Low. The implementation can only fail by not drawing an input.

**---------------------------------------------------------------------------------------------------------------------**

**Constraints:**

* Usability: No previous Training Time
* Reliability: The system shall recognize 10 inputs maximum and up to 10’000 different touch events per second.
* Performance: The displayed points should appear in less than ¼ of a second
* Supportability: The screens with multi-touch capabilities cannot exceed 10 fingers at the same time.
* Security: n/a

---------------------------------------------------------------------------------------------------------------------

**---------------------------------------------------------------------------------------------------------------------**

**Use Case ID:** TAMUC03 – States

**Use Case Level:** High-Level

**Details:**

* **Actors:** Developer
* **Pre-conditions:**
  1. The Application has been launched
  2. The screen is cleared or its started as a new blank canvas
* **Description:**
  1. Use case begins when the user touches the screen with 1-10 fingers inside the “drawing area”
  2. The Operating system flags the touch even and sends the touch signal and information to the application.
  3. The application receives the touch event “Touching” and labels it at such.
  4. The user moves their finger(s) across the draw area.
  5. The Operating system flags the touch event and sends the touch signal information to the application.
  6. The application receives the touch event “Moving” and labels it as such.
  7. The user lifts their finger(s) from the draw area.
  8. The Operating system flags the touch event and sends the touch signal information to the application.
  9. The application receives the touch event “Lifted” and labels it at such.
  10. The use case ends after the event “Lifted” is received.
* **Post-conditions:**
  1. The screen allows for another draw gesture.
* **Alternative Courses of Action:**
  1. The user after lifting finger(s) touches the draw area again and continues drawing.
* **Exceptions:**

The user asks another person for them to draw at the same time as him/her; making it more than 10 inputs (the system will only draw original 10).

* **Related Use Cases:**
  1. TAMUC01
  2. TAMUC02
  3. TAMUC04
  4. TAMUC05
  5. TAMUC07

**---------------------------------------------------------------------------------------------------------------------**

**Decision Support**

**Frequency:** On average, 1-10’000 touch events per second.

**Criticality:** High. The Framework cannot be tested if the touch events cannot be differentiated.

**Risk:** Low. The application does not fail if a touch event is not received.

**---------------------------------------------------------------------------------------------------------------------**

**Constraints:**

* Usability: No previous Training Time
* Reliability: The system shall recognize 10 inputs maximum and up to 10’000 different touch events per second.
* Performance: The touch events should be sent and received in less than 1/10000 of a second.
* Supportability: The screens with multi-touch capabilities cannot exceed 10 fingers at the same time.
* Security: n/a

---------------------------------------------------------------------------------------------------------------------

**---------------------------------------------------------------------------------------------------------------------**

**Use Case ID:** TAMUC04 – Mapping

**Use Case Level:** High-Level

**Details:**

* **Actors:** Developer
* **Pre-conditions:**
  1. The Application has been launched
  2. The screen is cleared or its started as a new blank canvas
  3. TAMUC01 is in step 2 of the description.
* **Description:**
  1. Use case begins when the user selects the mapping menu.
  2. The system displays a selection of all the available mapping functions.
  3. The user selects the desired mapping function.
  4. The system then keeps the selected mapping in memory whenever an input is obtained.
  5. The system shall compute the output coordinates based on the selected mapping function (use 1 to 1 if no function selected).
  6. The System traces all of the points while simultaneously saving the points (TAMUC07-Record)
  7. The use case ends when the last input is mapped to its output, displayed and saved.
* **Post-conditions:**
  1. The screen allows for another draw gesture.
  2. The latest selected mapping function remains active
* **Alternative Courses of Action:**
  1. The user after lifting finger(s) touches the draw area again and continues drawing.
  2. The user selects a different mapping function.
* **Exceptions:**

N/A

* **Related Use Cases:**
  1. TAMUC02

**---------------------------------------------------------------------------------------------------------------------**

**Decision Support**

**Frequency:** On average, 1-10 changes in the mapping function for every input test.

**Criticality:** Medium. Mapping functions can be used or 1 to 1 as crude testing implements.

**Risk:** Low. The default 1 to 1 mapping creates a safe fall.

**---------------------------------------------------------------------------------------------------------------------**

**Constraints:**

* Usability: No previous Training Time
* Reliability: The system shall map the output to two ½ a pixel from the received input.
* Performance: The displayed points should appear in less than ¼ of a second
* Supportability: The screens with multi-touch capabilities cannot exceed 10 fingers at the same time. The mapping function shall not exceed the edges of the drawing area.
* Security: n/a

---------------------------------------------------------------------------------------------------------------------

**---------------------------------------------------------------------------------------------------------------------**

**Use Case ID:** TAMUC05 – Input Recording

**Use Case Level:** High-Level

**Details:**

* **Actors:** Developer
* **Pre-conditions:**
  1. The Application has been launched
  2. The screen is cleared or its started as a new blank canvas
* **Description:**
  1. Use case begins when the user touches the screen.
  2. The system stores the X and Y coordinate, the ID of the finger, and a timestamp of the touch event.
  3. The user continues to touch the screen and move their fingers around the draw area.
  4. The system records each touch even and adds it at the end of a list.
  5. The use case ends when the user lifts all fingers from the draw screen and selects either replay, save, or closes the application.
* **Post-conditions:**
  1. The screen allows for another draw gesture.
  2. The system continues to wait for the user to end the current test.
* **Alternative Courses of Action:**
  1. The user after lifting finger(s) touches the draw area again and continues drawing.
* **Exceptions:**

The maximum number of touch points is reached.

* **Related Use Cases:**
  1. TAMUC02

**---------------------------------------------------------------------------------------------------------------------**

**Decision Support**

**Frequency:** On average, 1-10,000 touch events per second.

**Criticality:** High, It is necessary for the input to be saved so that the replay can happen and the visualizer performs its task.

**Risk:** Medium. The size of the input data may slow down the system while displaying the dots or while playing the recorded data back.

**---------------------------------------------------------------------------------------------------------------------**

**Constraints:**

* Usability: No previous Training Time
* Reliability: The system shall add the input to the list in less than 1/100 of a second.
* Performance: There list should be able to handle 1 Million touch points without noticeable system slowness.
* Supportability: The screens with multi-touch capabilities cannot exceed 10 fingers at the same time.
* Security: n/a

---------------------------------------------------------------------------------------------------------------------

**---------------------------------------------------------------------------------------------------------------------**

**Use Case ID:** TAMUC07 – Saving Input to a file

**Use Case Level:** High-Level

**Details:**

* **Actors:** Developer
* **Pre-conditions:**
  1. The Application has been launched
  2. The screen is cleared or its started as a new blank canvas
  3. The user has drawn on the draw area
* **Description:**
  1. Use case begins when the user selects the “Save” option.
  2. The system prompts the user for a filename and a file type. (CSV is default).
  3. The user selects file path and enters the name of the file (UNTITLED is default).
  4. The user selects the save button.
  5. The system creates the file with the name provided and saves each touch point in the order that the touch action happened.
  6. The use case ends when the system writes the last touch point in the file and closes the file.
* **Post-conditions:**
  1. The screen allows for another draw gesture.
  2. The file is ready to be read from.
* **Alternative Courses of Action:**
  1. The user cancels the save.
* **Exceptions:**

There might not be any touch points (empty file).

* **Related Use Cases:**
  1. TAMUC02
  2. TAMUC05

**---------------------------------------------------------------------------------------------------------------------**

**Decision Support**

**Frequency:** On average 5 saves for every touch test.

**Criticality:** Medium. The ability to save is necessary for replays of older actions, an immediate replay is more important as implemented in TAMUC06 Playback.

**Risk:** Low. If the saving action fails it only cancels the save then the user might save again.

**---------------------------------------------------------------------------------------------------------------------**

**Constraints:**

* Usability: No previous Training Time
* Reliability: The system shall write the input list to the file for every touch event.
* Performance: There file should be able to handle 1 Million touch points without noticeable system slowness.
* Supportability: The screens with multi-touch capabilities cannot exceed 10 fingers at the same time. The file needs to be saved in a Comma Separated Value file to be used by TAMUC08
* Security: n/a

---------------------------------------------------------------------------------------------------------------------

**---------------------------------------------------------------------------------------------------------------------**

**Use Case ID:** TAMUC08 – Playback from a File

**Use Case Level:** High-Level

**Details:**

* **Actors:** Developer
* **Pre-conditions:**
  1. The Application has been launched
  2. The screen is cleared or its started as a new blank canvas
  3. A file with a list of touch points has been created
* **Description:**
  1. Use case begins when the user selects the “Open” option.
  2. The system prompts the user to select the path of the file and the file to be replayed.
  3. The user navigates to the file location, selects the file and selects the “Open” option
  4. The system opens the file, and reads from it while saving the touch points to the replay list.
  5. The system replays the touch points per TAMUC06 Replay.
  6. The use case ends when the last touch point on the file has been replayed.
* **Post-conditions:**
  1. The screen allows for another draw gesture.
  2. The system allows for replay of the file since it has been open once already.
* **Alternative Courses of Action:**
  1. The user might cancel the open action by not selecting a file and selecting the “Cancel Option”
* **Exceptions:**

N/A

* **Related Use Cases:**
  1. TAMUC02
  2. TAMUC06

**---------------------------------------------------------------------------------------------------------------------**

**Decision Support**

**Frequency:** On average, 5 opens per touch test.

**Criticality:** Medium, immediate replay is more necessary than opening files previously saved.

**Risk:** Medium. The size of the input data may slow down the system while displaying the dots or while playing the recorded data back.

**---------------------------------------------------------------------------------------------------------------------**

**Constraints:**

* Usability: No previous Training Time
* Reliability: The system shall add the input to the list in less than 1/100 of a second.
* Performance: There list should be able to handle 1 Million touch points without noticeable system slowness.
* Supportability: The file type has to be a Comma Separated Value type.
* Security: n/a

---------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------

**Use Case ID:** TAMUC06 – Playback

**Use Case Level:** High-Level

**Details:**

* **Actors:** Developer
* **Pre-conditions:**
  1. The Application has been launched
  2. The screen is cleared or its started as a new blank canvas
* **Description:**
  1. Use case begins when the developer touches the screen with 1-10 fingers
  2. The system shall store the coordinate data from the touch event.(TAMUC04)
  3. The developer stops touching the screen.
  4. The developer chooses the “Playback” button
  5. Use case ends when the system replays the touch events from the stored data
* **Post-conditions:**
  1. The next time a developer causes a touch event the last touch event to be replayed gets replaced by the newest touch event.
* **Alternative Courses of Action:**
  1. The developer may choose to clear the screen and create a new touch event without replaying the previous touch event
* **Exceptions:**

None.

* **Related Use Cases:**
  1. TAMUC04

**---------------------------------------------------------------------------------------------------------------------**

**Decision Support**

**Frequency:** On average, 2-5 playbacks every 10 minutes.

**Criticality:** High. The Framework cannot be tested if a playback is not possible.

**Risk:** Low. The system is independent of the playback until it becomes time to replay, therefore it is not necessary to integrate and can be a module loaded when needed.

**---------------------------------------------------------------------------------------------------------------------**

**Constraints:**

* Usability: No previous Training Time
* Reliability: The system shall replay the touch events up to ½ mm precision.
* Performance: The replay should load in no more than 1 second
* Supportability: The screens with multi-touch capabilities cannot exceed 10 fingers at the same time.
* Security: n/a

---------------------------------------------------------------------------------------------------------------------

**Use Case ID:** TAMUC09 – Resize

**Use Case Level:** High-Level

**Details:**

* **Actors:** Developer
* **Pre-conditions:**
  1. The Application has been launched
  2. The screen is cleared or its started as a new blank canvas
* **Description:**
  1. Use case begins when the user selects the “Size” option.
  2. The system displays a selection of all the available sizes that can be applied to the touch visualization.
  3. The user selects the size wanted.
  4. Use case ends when the system applies the selected size brush to any future touch event.
* **Post-conditions:**
  1. The next time a developer causes a touch event the displayed brush output will be the size selected by the user.
* **Alternative Courses of Action:**
  1. The developer may choose to select a different size after the first selection.
* **Exceptions:**

None.

* **Related Use Cases:**
  1. TAMUC02
  2. TAMUC04

**---------------------------------------------------------------------------------------------------------------------**

**Decision Support**

**Frequency:** On average, 5-10 resize events per minute.

**Criticality:** Medium. Resizing is a secondary need to be able to determine the precision of the touch mapping.

**Risk:** Low. The Resizing event changes the size of the drawing brush and no other aspects.

**--------------------------------------------------------------------------------------------------------------------**

**Constraints:**

* Usability: No previous Training Time
* Reliability: The size selected is to be used every time, no failures are acceptable.
* Performance: The change in the size should be available immediately after selecting it.
* Supportability: Brush sizes 5 - 100.
* Security: n/a

**---------------------------------------------------------------------------------------------------------------------**

**---------------------------------------------------------------------------------------------------------------------**

**Use Case ID:** TAMUC010 – Clear Screen

**Use Case Level:** High-Level

**Details:**

* **Actors:** Developer
* **Pre-conditions:**
  1. The Application has been launched
  2. The screen has been drawn to
* **Description:**
  1. Use case begins when the user selects the “Clear” option.
  2. The system creates a new blank canvas the size of the draw area.
  3. Use case ends when the system replaces the old canvas with the new canvas and displays it to the user.
* **Post-conditions:**
  1. The draw area will be clear allowing for new touch events to happen.
* **Alternative Courses of Action:**
  1. The developer may choose to clear the screen more than once consecutively.
* **Exceptions:**

None.

* **Related Use Cases:**
  1. TAMUC02
  2. TAMUC04

**---------------------------------------------------------------------------------------------------------------------**

**Decision Support**

**Frequency:** On average, 1 – 30 clear events per minute.

**Criticality:** High. For the touch replays and the touch events to be studied there is a need to have a clear canvas.

**Risk:** Low. The new blank canvas can be created in a secondary process to that of the touch actions.

**---------------------------------------------------------------------------------------------------------------------**

**Constraints:**

* Usability: No previous Training Time
* Reliability: For every 1 million clear events the screen should clear 999,999 times.
* Performance: The screen shall appear blank in less than 1/1000 of a second
* Supportability: The new canvas should support screens of all sizes.
* Security: n/a

**---------------------------------------------------------------------------------------------------------------------**

**---------------------------------------------------------------------------------------------------------------------**

**Use Case ID:** TAMUC011 – Debug

**Use Case Level:** High-Level

**Details:**

* **Actors:** Developer
* **Pre-conditions:**
  1. The Application has been launched
* **Description:**
  1. Use case begins when the application first launches.
  2. The system opens a window displaying a debug terminal.
  3. As the actor uses the program any message resulting form that will be displayed in this debug monitor.
  4. Use case ends when the user closes the application.
* **Post-conditions:**
  1. Closing the debug monitor causes the program to close.
* **Alternative Courses of Action:**
  1. The developer may choose to close the program but not the debug monitor.
* **Exceptions:**

None.

* **Related Use Cases:**

n/a

**---------------------------------------------------------------------------------------------------------------------**

**Decision Support**

**Frequency:** On average, 1 – 10,000 messages per minute.

**Criticality:** Medium. The debug monitor is a secondary instance to the main application.

**Risk:** Low. The debug has no connection to the application the messages only go one way from the application to the debug.

**---------------------------------------------------------------------------------------------------------------------**

**Constraints:**

* Usability: No previous Training Time
* Reliability: For every 1 million messages 1 missed message and 999,999 written.
* Performance: The information being displayed shall do so in less than 1/1000 of a second
* Supportability: The debug monitor is a Windows application and shall only work in Windows Operating Systems.

## Appendix B -Use case diagram using UML

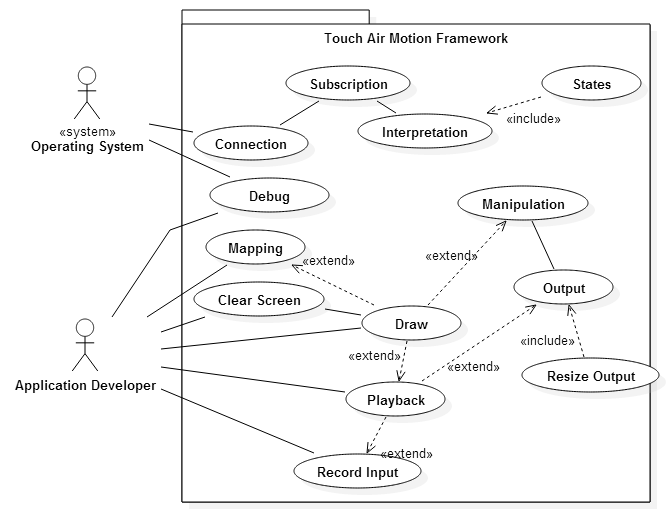


Fig. 6.2.1

## Appendix C - Static UML diagram

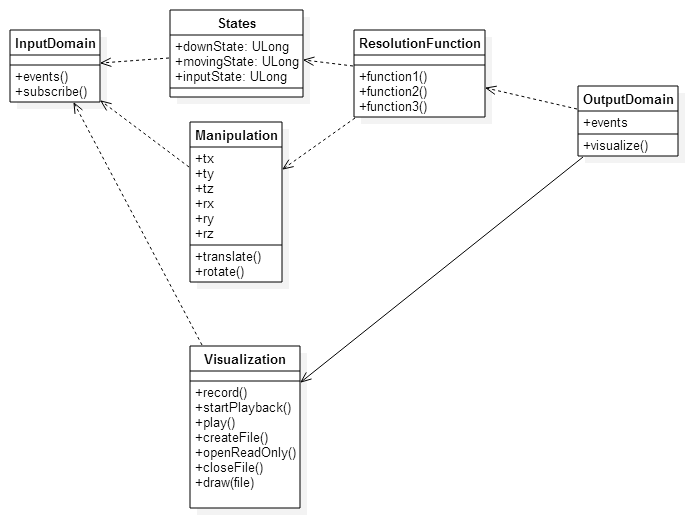


Fig. 6.3.1

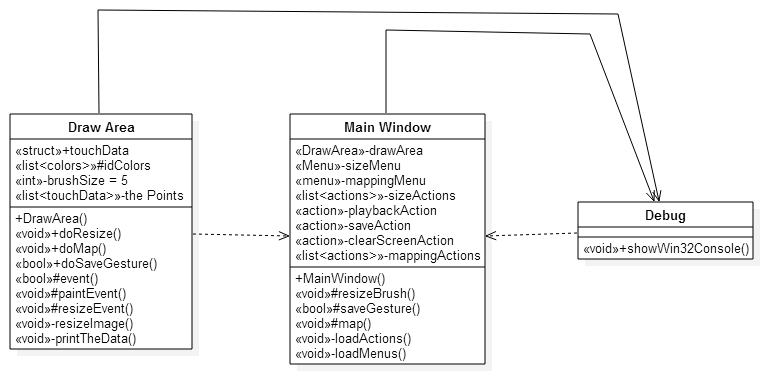


Fig. 6.3.2

## Appendix D - Dynamic UML diagram

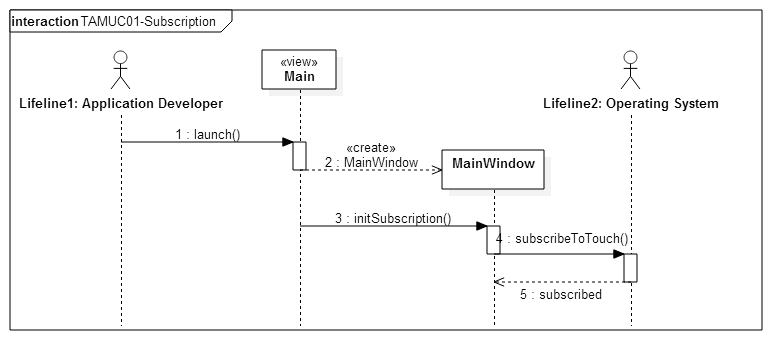


Fig.6.4.1

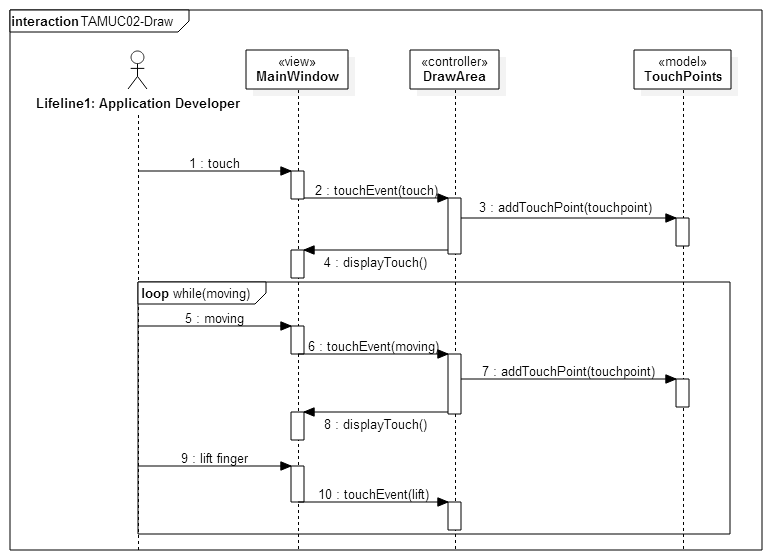


Fig. 6.4.2

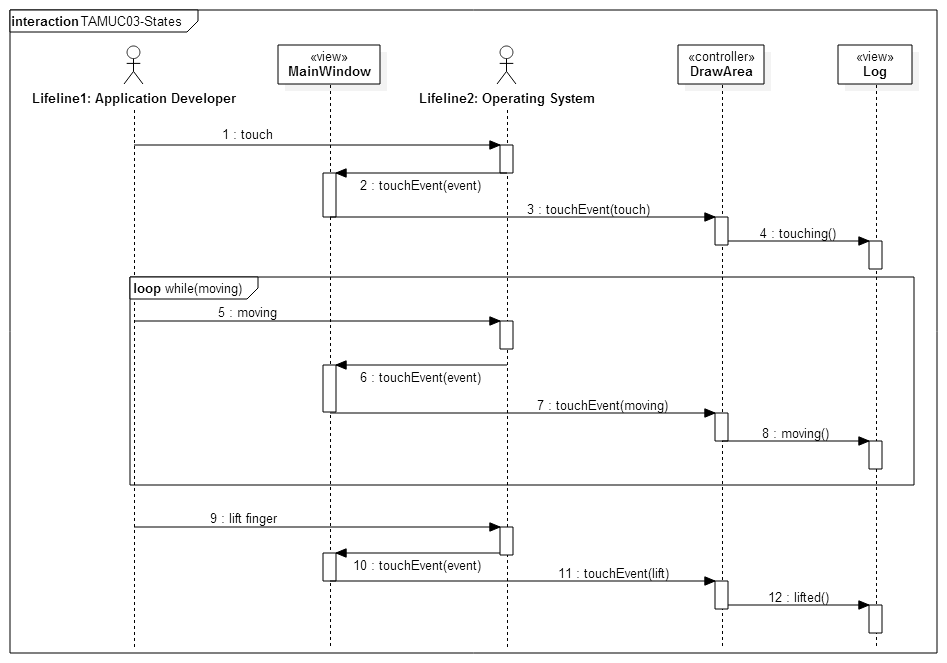
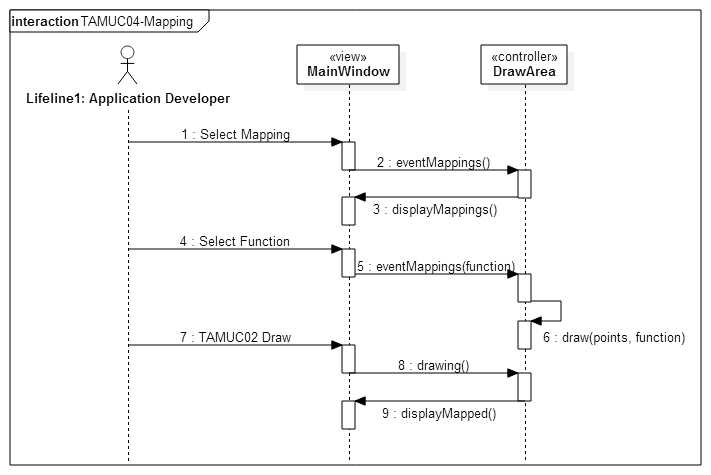
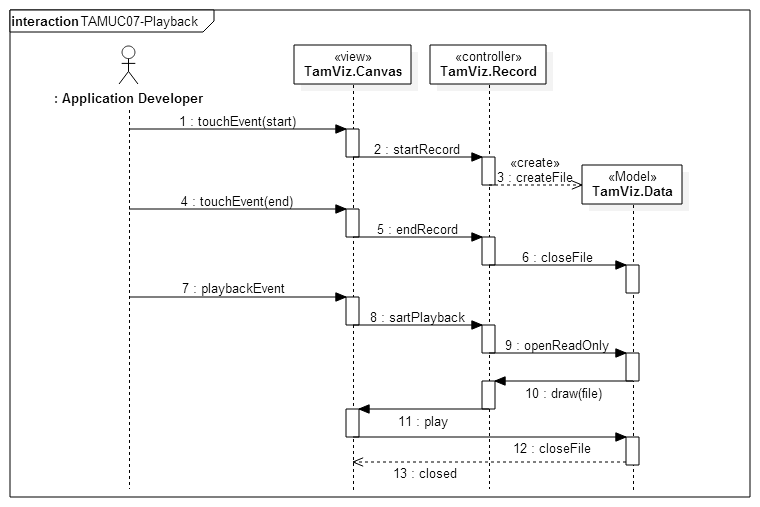
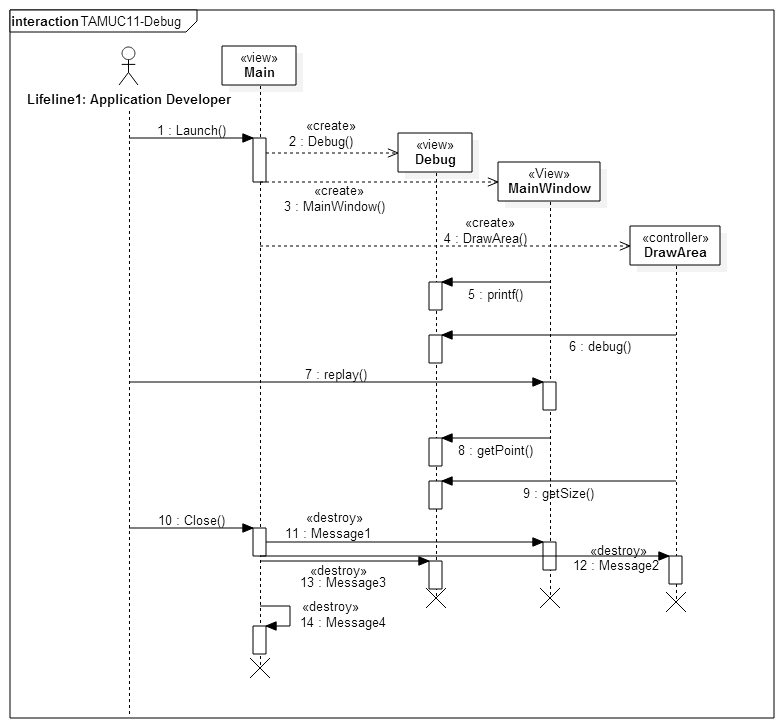


Fig. 6.4.3

Fig. 6.4.4

Fig. 6.4.7

Fig. 6.4.11



## Appendix E - User Interface designs.

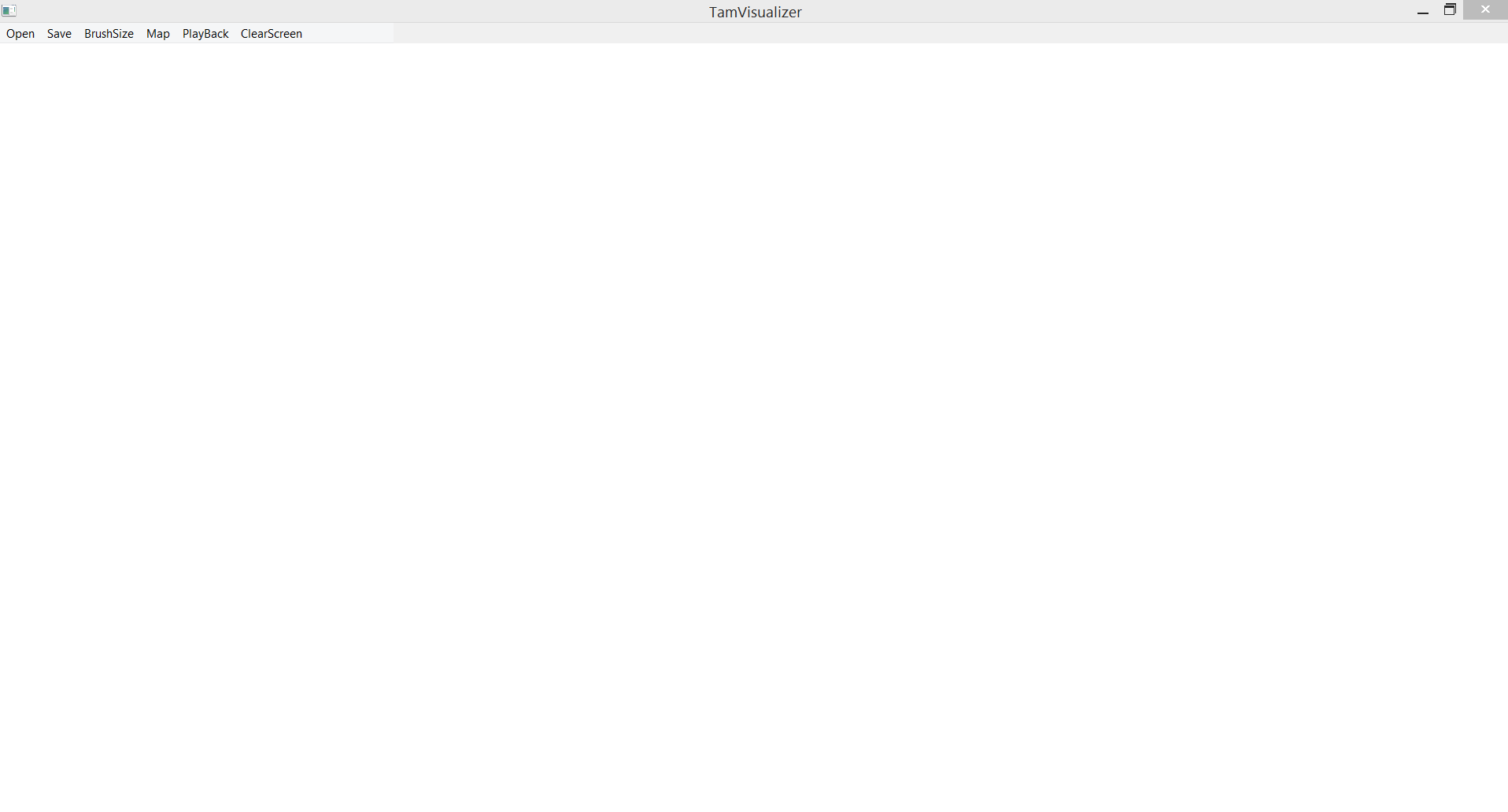


Fig. 6.5.1

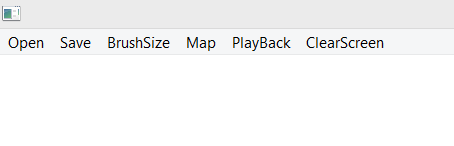


Fig 6.5.2

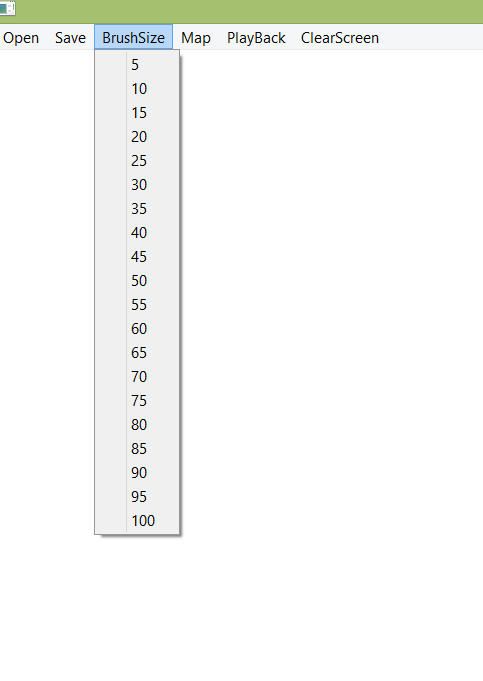


Fig. 6.5.3

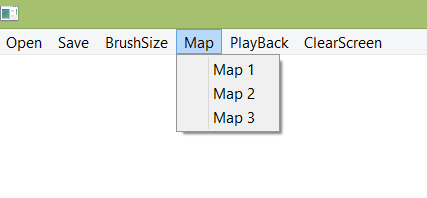


Fig. 6.5.4

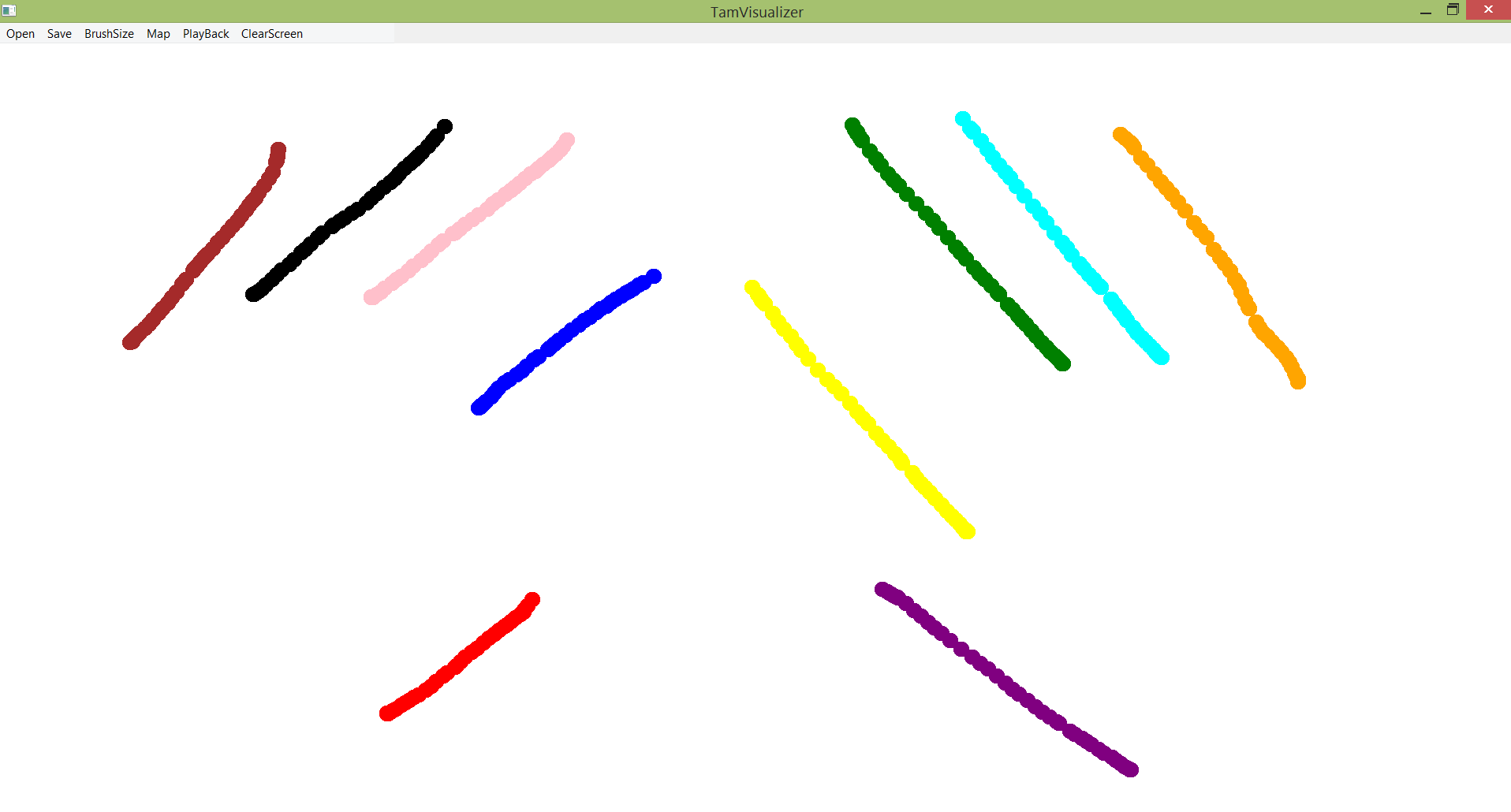


Fig. 6.5.5

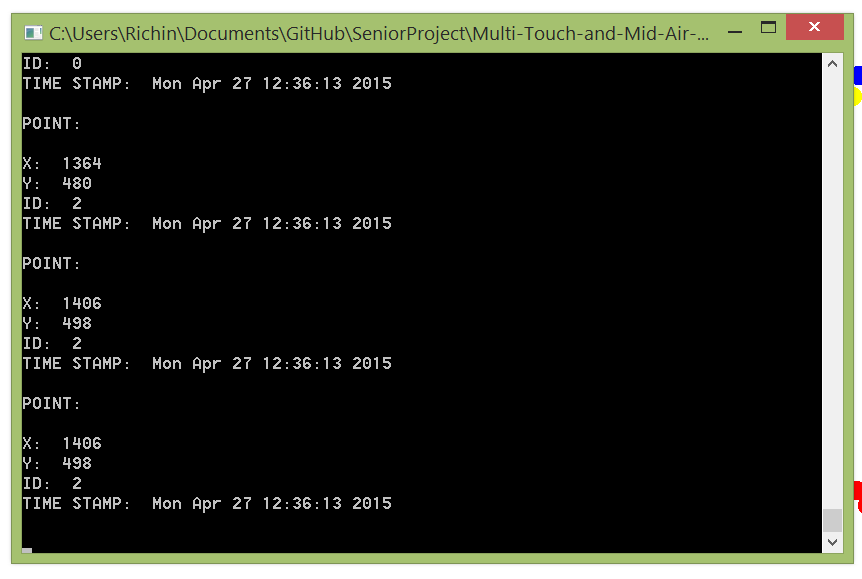


Fig. 6.5.6

## Appendix F - Diary of meeting and tasks.

Meeting 1:

Date: 01/19/15

Participants: Anthony M Amador

Francisco Ortega

Richard A Lopez

Topics discussed:

* More in-depth explanation of the system by Francisco
* Detailed explanation on Francisco expectations on Anthony and Richard
* Background information obtained from Anthony and Richard
* Set up next meeting Tuesday January 20th

Meeting 2:

Date: 01/20/15

Participants: Anthony M Amador

Francisco Ortega

Richard A Lopez

Topics discussed:

* Assigned Scrum roles: Richard: Scrum Master

Anthony: Team Member

* Discussed possible use cases
* Set up next meeting Wednesday January 21st

Meeting 3:

Date: 01/21/15

Participants: Anthony M Amador

Francisco Ortega

Richard A Lopez

Topics discussed:

* Took down all the ideas that Francisco wanted for the system
* Molded the ideas into the format of user stories ”who, what, why”
* Collected all the user stories and placed the ones that seemed unfeasible at the moment
* Set up a face to face meeting to learn C++ for Monday January 26th

Meeting 4:

Date: 01/26/15

Participants: Anthony M Amador

Francisco Ortega

Richard A Lopez

Topics discussed:

* Francisco introduced C++
* Francisco introduced some code for the system
* Anthony and Richard set up reading from C++ book
* Set up next meeting for Wednesday January 28th

Meeting 5:

Date: 01/28/15

Participants: Anthony M Amador

Richard A Lopez

Topics discussed:

* Cleaned up user stories
* Decided on points to be given for all stories
* Decided on stories that could not be developed during this semester
* Split up development of documents and diagrams
* Set up net meeting for Saturday January 31st

Meeting 6:

Date: 01/31/15

Participants: Anthony M Amador

Francisco Ortega

Richard A Lopez

Topics discussed:

* Talked about the task that would be created for each story
* Decided on meeting time for sprint planning for Monday February 2nd
* Split up more work for presentation

Meeting 7:

Date: 02/02/15

Participants: Francisco Ortega

Richard A Lopez

Topics discussed:

* Talked about the task that would be created for each story
* Decided on the task to be implemented for sprint 1
* Talked about learning Qt
* Created the basic structure of the Git repository

Meeting 8:

Date: 02/13/15

Participants: Francisco Ortega

Richard A Lopez

Topics discussed:

* Reviewed the work done in sprint one
* Went over the user stories to refine them
* Sent a few user stories to the backlog
* Created a new user story replay
* Learned more C++ from Francisco
* Talked about the user stories for next sprint, all user stories will be carried over

Meeting 9:

Date: 02/27/15

Participants: Francisco Ortega

Richard A Lopez

Topics discussed:

* Reviewed the work done in sprint two
* Accepted most of the user stories for sprint two
* Choose the user stories for sprint three
* Clarified some questions on Qt and the touch interface
* Learned more C++ from Francisco

Meeting 10:

Date: 03/20/15

Participants: Francisco Ortega

Richard A Lopez

Topics discussed:

* Reviewed the work done in sprint three
* Accepted all user stories from sprint three
* Re-defined the user stories for the semester due to a teammate leaving
* Choose the user stories for sprint four
* Created new user stories to accommodate for previous defects
* Talked about the mapping and recognizer user stories

Meeting 11:

Date: 04/03/15

Participants: Francisco Ortega

Richard A Lopez

Topics discussed:

* Reviewed the work done in sprint four
* Accepted all user stories except saving to file from sprint four
* Choose the user stories for sprint five that encompasses all the needs for a visualizer in version one point zero.

Meeting 12:

Date: 04/17/15

Participants: Francisco Ortega

Richard A Lopez

Topics discussed:

* Reviewed the work done in sprint five
* Accepted the final user story
* Talked about the documents and the poster
* Went over the complete visualizer
* All done Now to work on the documents!!

# References

“Agile tutorial for the Senior Project Class”. Web. 26 Apr. 2015. <https://moodle.cis.fiu.edu/v2.8/pluginfile.php/45767/mod\_resource/content/2/Agile\_Tutorial.pdf>

Taylor II, Russell. "Overview." *VRPN*. Web. 26 Apr. 2015. <http://www.cs.unc.edu/Research/vrpn/>