*Florida International University*

*School of Computing and Information Sciences*

Software Engineering Focus

Feature Document

User Story ID #712

**Name:** Francisco Lozada

**Team Member(s):** Lukas Borges, Filip Klepsa, Nicolette Celli, Francisco Lozada, Cristian Cabrera

**Project:** AR-VR-VE for Computer Science (Circular Gesture Recognition API)

**Product Owner(s)**: Francisco Ortega

**Mentor(s)**: Francisco Ortega

**Instructor**: Francisco Ortega, Masoud Sadjadi

**User Story Name:** Translate the PointMap Class to C++

* Description: **As a** developer **I would like** to translate the PointMap class from the MTGRLibrary to C++ **so that** it can be used within the Gesture class for the API

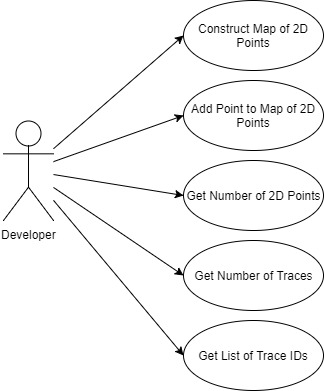
**Acceptance Criteria**

* Must have C++ API best design practices implemented
* Must have the required attributes and member functions that provide the exact translation of the class written in C# to C++ code

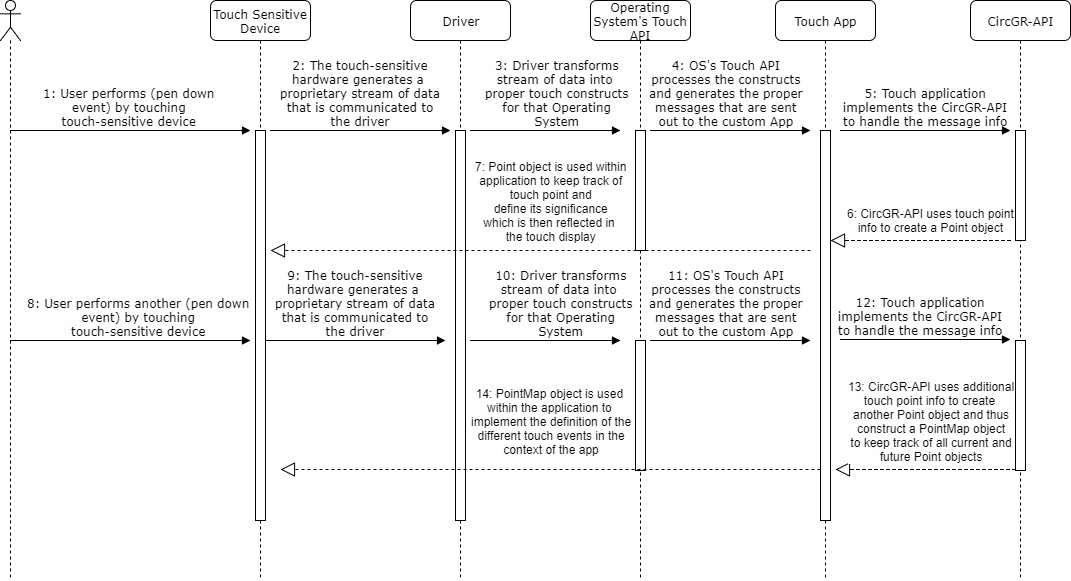
**Use Case**

* Name: Construct Map of 2D Points
* Actor: Developer
* Preconditions: Have the CircGR-API library (Point.cpp and PointMap.cpp file in this case) and #include “PointMap.h” header within the code
* Description:
  + Developer uses the Point class to store a touch point’s properties
    - * System instantiates a Point object whenever it is called upon and provided the right parameters: X, Y, StrokeID, and Timestamp
  + Developer uses the PointMap class to store a list of Point objects inside a map whose key is the Stroke ID that is common among the Point objects in the list.
    - System instantiates PointMap object which is basically a map that keeps track of a set of strokes.
* Name: Add Point to Map of 2D Points
* Actor: Developer
* Preconditions: Have the CircGR-API library (Point.cpp and PointMap.cpp file in this case) and #include “PointMap.h” header within the code
* Description:
  + Developer uses PointMap class to store a list of Point objects inside a map whose key is the Stroke ID that is common among the Point objects in the list.
    - System instantiates PointMap object which is a map that keeps track of a set of strokes.
  + Developer creates a new Point object
    - System adds the new Point object based on the Stroke ID to the correct list of Point objects inside the PointMap
* Name: Get Number of 2D Points
* Actor: Developer
* Preconditions: Have the CircGR-API library (Point.cpp and PointMap.cpp file in this case) and #include “PointMap.h” header within the code
* Description:
  + Developer uses PointMap class to store a list of Point objects
    - System instantiates PointMap object which keeps track of a set of strokes.
  + Developer uses getNumberOfPoints function
    - System returns the total number of Point objects within a PointMap
* Name: Get Number of Traces
* Actor: Developer
* Preconditions: Have the CircGR-API library (Point.cpp and PointMap.cpp file in this case) and #include “PointMap.h” header within the code
* Description:
  + Developer uses PointMap class to store a list of Point objects
    - System instantiates PointMap object which keeps track of a set of strokes.
  + Developer uses getNumOfTraces function
    - System returns the total number of distinct traces. In other words, it returns the number of different list of Point objects with the same Stroke ID.
* Name: Get List of Trace IDs
* Actor: Developer
* Preconditions: Have the CircGR-API library (Point.cpp and PointMap.cpp file in this case) and #include “PointMap.h” header within the code
* Description:
  + Developer uses PointMap class to store a list of Point objects
    - System instantiates PointMap object which keeps track of a set of strokes.
  + Developer uses getTraceIDs function
    - System returns a list of the different IDs for the different collection of Point Objects with the same Stroke ID

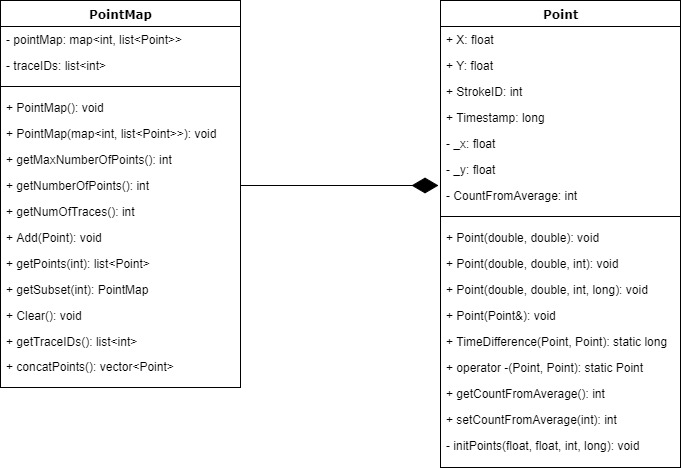
**Use Case Diagram**



**Sequence Diagram**



**Class Diagram**



**Unit Test**

* Test case ID: Construct\_Point\_Map
* Description/Summary of Test: Tests whether a map containing Stroke IDs as keys and list of Point objects as values is properly constructed
* Pre-condition: Have the CircGR-API library (PointMap.cpp file in this case) and #include “PointMap.h” header within the code as well as the proper stubs for the Point class
* Expected Results: Map whose keys are integers and the values are lists of Point objects is created
* Actual Result: same as expected results
* Status (Fail/Pass): Pass
* Test case ID: Add\_Point\_to\_Map
* Description/Summary of Test: Tests whether a Point object is added to the correct value of a PointMap object
* Pre-condition: Have the CircGR-API library (PointMap.cpp file in this case) and #include “PointMap.h” header within the code as well as the proper stubs for the Point class
* Expected Results: Point object is added to the corresponding list of Point objects that have the same Stroke ID
* Actual Result: same as expected results
* Status (Fail/Pass): Pass
* Test case ID: Get\_Number\_of\_Points
* Description/Summary of Test: Tests whether all Point objects generated are correctly counted and kept track of
* Pre-condition: Have the CircGR-API library (PointMap.cpp file in this case) and #include “PointMap.h” header within the code as well as the proper stubs for the Point class
* Expected Results: The number of Point objects within a list of Point objects inside a Point Map is returned
* Actual Result: same as expected results
* Status (Fail/Pass): Pass
* Test case ID: Get\_Number\_of\_Traces
* Description/Summary of Test: Tests whether the number of distinct traces (or list of distinct Point objects) is correctly calculated
* Pre-condition: Have the CircGR-API library (PointMap.cpp file in this case) and #include “PointMap.h” header within the code as well as the proper stubs for the Point class
* Expected Results: The number of distinct touch traces is returned
* Actual Result: same as expected results
* Status (Fail/Pass): Pass
* Test case ID: Get\_List\_of\_Trace\_IDs
* Description/Summary of Test: Tests whether a list of the different trace IDs is correctly generated without any mistakes and duplications
* Pre-condition: Have the CircGR-API library (PointMap.cpp file in this case) and #include “PointMap.h” header within the code as well as the proper stubs for the Point class
* Expected Results: The list of IDs for the different touch traces is returned
* Actual Result: same as expected results
* Status (Fail/Pass): Pass

**Integration Test**

To perform integration testing during this particular user story I performed the same tests as for the unit tests but instead of using Stubs for the Point class to separate both the Point class and PointMap class from each other I used the actual class for the integration tests. This allowed me to quickly perform unit tests at the beginning and then later on verify whether the PointMap class was indeed calling and using the information from the Point class correctly.

**Visual User Guide**

