*Florida International University*

*School of Computing and Information Sciences*

Software Engineering Focus

Feature Document

User Story ID #719

**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 CircClassifier class to C++

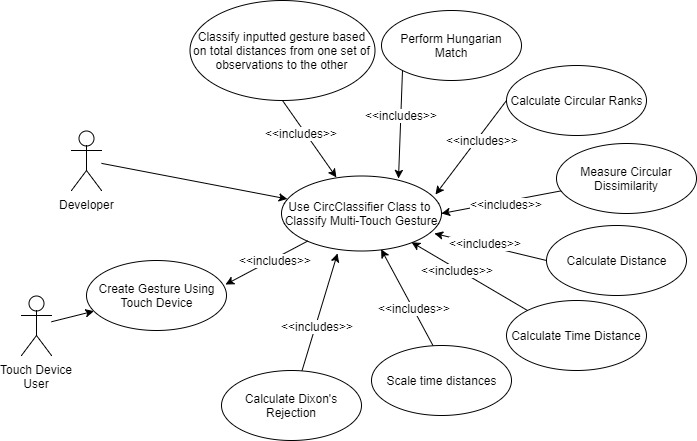
* Description: **As a** developer **I would like** to translate the CircClassifier class from the MTGRLibrary to C++ **so that** it can be used to classify multi-touch circular gestures.

**Acceptance Criteria**

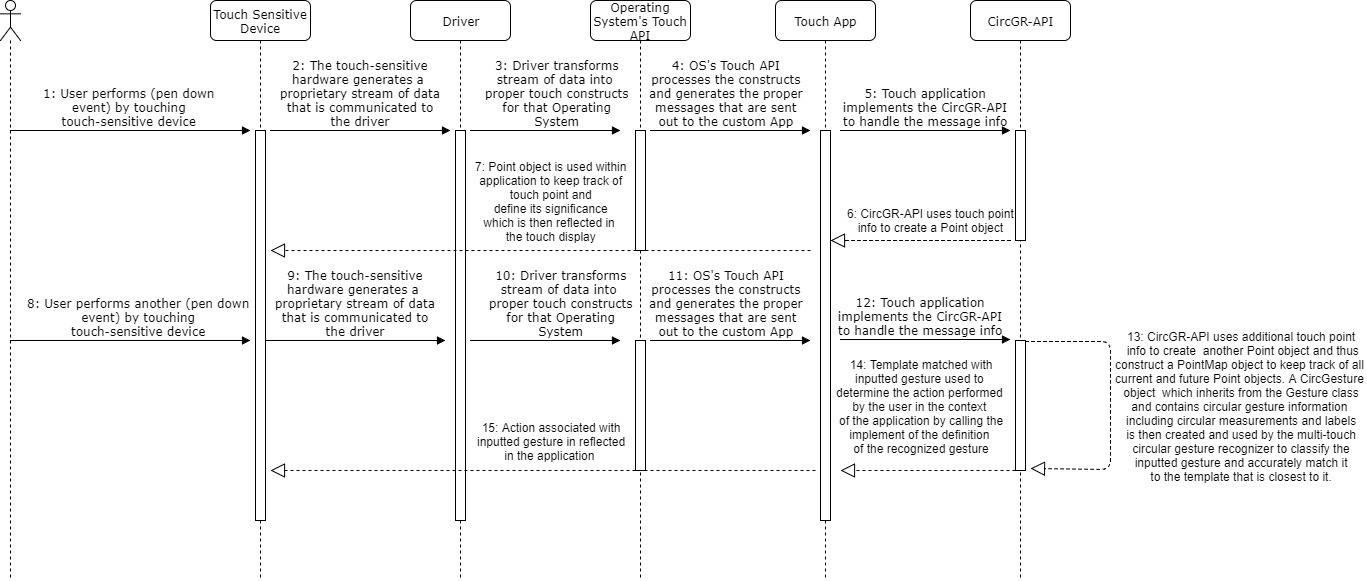
* Must have C++ API best design practices implemented
* Must have the required attributes and member functions that provide the exact translation of the class written in C# to C++ code

**Use Case**

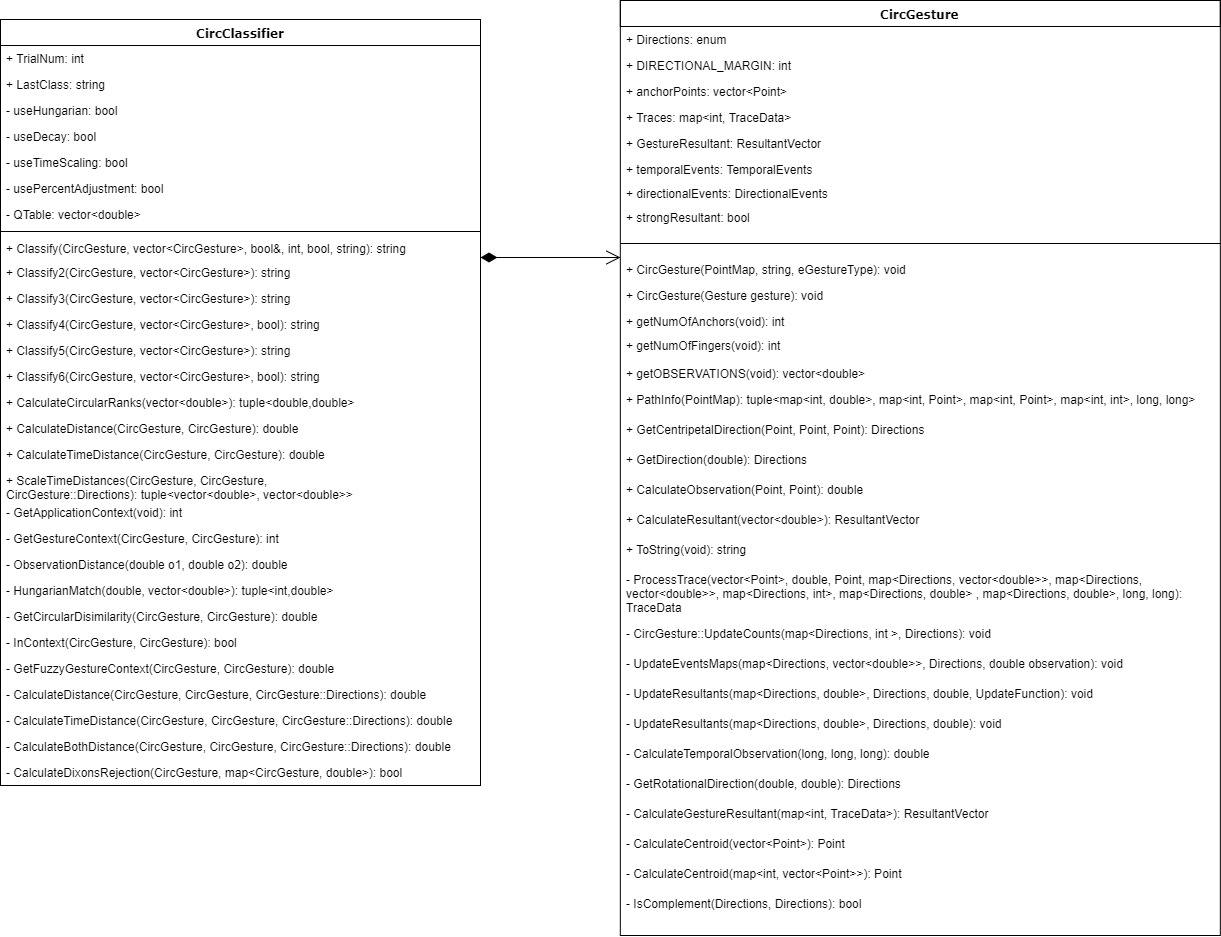
* Name: Accurately Classify Circular Gesture
* Actor: Developer
* Preconditions: Have the CircGR-API library (all cpp and headers files inside namespace ‘GR’) and #include “CircClassifier.h” header within the code
* Description:
  + Developer uses the Point class to store a touch point’s properties
    - * 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 which a map for traces where the key is the touch ID of trace and the value is a list of points for that trace.
  + Developer uses the Gesture class to use this Pointmap to create a gesture which is normalized with respect to scale, translated to origin, and resampled into a fixed number of SAMPLING\_RESOLUTION specified points
    - System instantiates a generic Gesture Object which consist of unprocessed traces, a name, a type, and the template name it is supposed to be a Gesture
  + Developer uses the CircGesture class to inherit from the Gesture class and extend its functionality to deal with circular measurements as gesture representation
    - System creates CircGesture object that inherits from gesture to compute and work with the spatial angles and temporal angles. This will then later be used for pattern-matching and classifying circular gestures.
  + Developer uses the CircClassifier class to analyze circGesture object constructed and determine Gesture template that best matches
    - System uses all measurements and labels to accurately determine the template that best matches the inputted gesture

**Use Case Diagram**

**Sequence Diagram**



**Class Diagram**



**Unit Test**

* Test case ID: Classify\_Inputted\_Circular\_Gesture
* Description/Summary of Test: Test whether a Circular Gesture object that contains circular measurements and labels pertaining to an inputted gesture is properly used by the CircClassifier class to determine what template best matches the inputted gesture.
* Pre-condition: Have the CircGR-API library (all cpp and headers files inside namespace ‘GR’) and #include “CircClassifier.h” header within the code
* Expected Results: Inputted circular gesture is accurately matched to the intended template
* Actual Result: same as expected results
* Status (Fail/Pass): Pass

**Integration Test**

To perform integration testing during this particular user story I first tested the interoperability between the following classes in a particular order:

1. Point class and PointMap class
2. Point class and Geometry class
3. Point class and Gesture class
4. Point class, PointMap class, and Geometry class
5. Point class, PointMap class, Geometry class, and Gesture class
6. Point class, PointMap class, Geometry class, Gesture class, and CircGesture class, and CircClassifier class

**Visual User Guide**

