Acceptance Test Document

Project Clementine Group

Table of Contents

1. Infrastructure Acceptance Test .............................................................................................................. 6
   1.1. Programmer Interface .................................................................................................................. 6
      1.1.1. Interface Test Standards ................................................................................................... 6
      1.1.2. Test the application in C++ ............................................................................................. 6
         1.1.2.1. Procedure ............................................................................................................... 6
         1.1.2.2. Pass Condition ....................................................................................................... 6
      1.1.3. Test Static Library on Linux ............................................................................................ 6
         1.1.3.1. Procedure ............................................................................................................... 6
         1.1.3.2. Pass Condition ....................................................................................................... 6
      1.1.4. Test Dynamic Library on Windows ................................................................................. 6
         1.1.4.1. Procedure ............................................................................................................... 6
         1.1.4.2. Pass Condition ....................................................................................................... 7
   1.2. Threading ..................................................................................................................................... 7
      1.2.1. Library In A Different Thread ......................................................................................... 7
         1.2.1.1. Procedure ............................................................................................................... 7
         1.2.1.2. Pass Condition ....................................................................................................... 7
      1.2.2. MultiThreaded Library ..................................................................................................... 7
         1.2.2.1. Procedure ............................................................................................................... 7
         1.2.2.2. Pass Condition ....................................................................................................... 7
      1.2.3. Threading Priority ............................................................................................................ 7
         1.2.3.1. Procedure ............................................................................................................... 7
         1.2.3.2. Pass Condition ....................................................................................................... 8
      1.2.4. Number of Threads .......................................................................................................... 8
         1.2.4.1. Procedure ............................................................................................................... 8
         1.2.4.2. Pass Condition ....................................................................................................... 8
      1.2.5. Compile Time Customization .......................................................................................... 8
         1.2.5.1. Procedure ............................................................................................................... 8
         1.2.5.2. Pass Condition ....................................................................................................... 8
      1.2.6. PreCompilation Directives ............................................................................................... 8
         1.2.6.1. Procedure ............................................................................................................... 8
         1.2.6.2. Pass Condition ....................................................................................................... 8
   2. Data Acceptance Test ............................................................................................................................ 9
      2.1. Data Importing ............................................................................................................................ 9
         2.1.1. Import ............................................................................................................................... 9
            2.1.1.1. Procedure ............................................................................................................... 9
            2.1.1.2. Pass Condition ....................................................................................................... 9
            2.1.1.3. IR Dot Intensity .................................................................................................... 10
               2.1.1.3.1. Procedure .................................................................................................. 10
               2.1.1.3.2. Pass Condition .......................................................................................... 10
2.1. Input History .......................................................................................................................... 10
  2.1.1. Procedure .......................................................................................................................... 10
  2.1.2. Pass Condition .................................................................................................................... 10
2.1.3. History Access Time ........................................................................................................... 10
  2.1.3.1. Procedure ....................................................................................................................... 11
  2.1.3.2. Pass Condition ............................................................................................................... 11
2.2. Data Exporting ........................................................................................................................ 11
  2.2.1. Procedure .......................................................................................................................... 11
  2.2.2. Positional Data ................................................................................................................... 11
    2.2.2.1. Procedure .................................................................................................................... 11
    2.2.2.2. Pass Condition .............................................................................................................. 11
  2.2.3. Velocity Data ..................................................................................................................... 12
    2.2.3.1. Procedure .................................................................................................................... 12
    2.2.3.2. Pass Condition .............................................................................................................. 12
  2.2.4. Acceleration Data ............................................................................................................. 12
    2.2.4.1. Procedure .................................................................................................................... 12
    2.2.4.2. Pass Condition .............................................................................................................. 12
  2.2.5. Gestures ............................................................................................................................. 12
    2.2.5.1. Procedure .................................................................................................................... 12
    2.2.5.2. Pass Condition .............................................................................................................. 13
  2.2.6. Export Rate ....................................................................................................................... 13
    2.2.6.1. Procedure .................................................................................................................... 13
    2.2.6.2. Pass Condition .............................................................................................................. 13
2.3. Data Representation .................................................................................................................. 13
  2.3.1. Reference Frames .............................................................................................................. 13
    2.3.1.1. First Frame .................................................................................................................... 13
      2.3.1.1.1. Procedure ................................................................................................................. 13
      2.3.1.1.2. Pass Condition ....................................................................................................... 13
    2.3.1.2. Next Frame .................................................................................................................... 13
      2.3.1.2.1. Procedure ................................................................................................................. 13
      2.3.1.2.2. Pass Condition ....................................................................................................... 14
    2.3.1.3. Next N Frames ............................................................................................................. 14
      2.3.1.3.1. Procedure ................................................................................................................. 14
      2.3.1.3.2. Pass Condition ....................................................................................................... 14
    2.3.1.4. Previous Frame ............................................................................................................. 14
      2.3.1.4.1. Procedure ................................................................................................................. 14
      2.3.1.4.2. Pass Condition ....................................................................................................... 14
    2.3.1.5. Previous N Frames ....................................................................................................... 14
      2.3.1.5.1. Procedure ................................................................................................................. 14
      2.3.1.5.2. Pass Condition ....................................................................................................... 14
    2.3.1.6. Get Frame ...................................................................................................................... 14
      2.3.1.6.1. Procedure ................................................................................................................. 14
      2.3.1.6.2. Pass Condition ....................................................................................................... 15
  2.3.2. Quaternion To Roll Pitch Yaw ........................................................................................... 15
    2.3.2.1. Procedure .................................................................................................................... 15
    2.3.2.2. Pass Condition .............................................................................................................. 15
2.3.3. Roll Pitch Yaw To Quaternion ................................................................. 15
   2.3.3.1. Procedure ......................................................................................... 15
   2.3.3.2. Pass Condition ................................................................................ 15
2.4. Data Transformation ............................................................................. 15
   2.4.1. Standard Data Sets ............................................................................. 15
   2.4.2. Test Linear Regression ........................................................................ 16
       2.4.2.1. Procedure ..................................................................................... 16
       2.4.2.2. Pass Condition ............................................................................. 16
   2.4.3. Test Circular Regression ..................................................................... 16
       2.4.3.1. Procedure ..................................................................................... 16
       2.4.3.2. Pass Condition ............................................................................. 16
   2.4.4. Test Data Point Collision Resolution .................................................. 16
       2.4.4.1. Procedure ..................................................................................... 16
       2.4.4.2. Pass Condition ............................................................................. 17
   2.4.5. Test Update Distance Tolerance ......................................................... 17
       2.4.5.1. Procedure ..................................................................................... 17
       2.4.5.2. Pass Condition ............................................................................. 17
   2.4.6. Test Data Point Loss ........................................................................... 17
       2.4.6.1. Procedure ..................................................................................... 17
       2.4.6.2. Pass Condition ............................................................................. 17
   2.4.7. Test point removal and waiting queue ............................................... 17
       2.4.7.1. Procedure ..................................................................................... 17
       2.4.7.2. Pass Condition ............................................................................. 18
   2.4.8. Test Dimensional Smoothing ............................................................ 18
       2.4.8.1. Procedure ..................................................................................... 18
       2.4.8.2. Pass Condition ............................................................................. 18
   2.4.9. Test Future Computation Using Raw Data ......................................... 18
       2.4.9.1. Procedure ..................................................................................... 18
       2.4.9.2. Pass Condition ............................................................................. 18
   2.4.10. Test Derivative Computation ........................................................... 19
       2.4.10.1. Procedure ................................................................................... 18
       2.4.10.2. Pass Condition ........................................................................... 19
2.5. Abstract Data Processing ..................................................................... 19
   2.5.1. Test Atomic Gesture .......................................................................... 19
       2.5.1.1. Procedure ..................................................................................... 19
       2.5.1.2. Pass Condition ............................................................................. 19
   2.5.2. Test Complex Gesture ....................................................................... 19
       2.5.2.1. Procedure ..................................................................................... 19
       2.5.2.2. Pass Condition ............................................................................. 19
3. Diagnostic Utilities Tests ........................................................................... 20
   3.1. Data Raw Graph .................................................................................... 20
       3.1.1. Procedure ......................................................................................... 20
       3.1.2. Pass Condition ................................................................................ 20
   3.2. Data Smoothed Graph ............................................................................ 20
       3.2.1. Procedure ......................................................................................... 20
       3.2.2. Pass Condition ................................................................................ 20
3.3. Data Accuracy Graph ............................................................................................................. 21
  3.3.1. Procedure .......................................................................................................................... 21
  3.3.2. Pass Condition .................................................................................................................. 21
4. Interface Functionality Tests ........................................................................................................ 22
  4.1. 3D Interface .......................................................................................................................... 22
    4.1.1. Standards ......................................................................................................................... 22
      4.1.1.1. Viewing Angle ........................................................................................................... 22
      4.1.1.2. IR Dot Size ................................................................................................................ 22
    4.1.2. Tests ................................................................................................................................ 22
      4.1.2.1. Tilt Recognition ......................................................................................................... 22
        4.1.2.1.1. Procedure ........................................................................................................... 22
        4.1.2.1.2. Pass Condition ................................................................................................. 23
      4.1.2.2. Comparison of Digital Results to Real World Positions ............................................. 23
        4.1.2.2.1. Procedure ........................................................................................................... 23
        4.1.2.2.2. Pass Condition ................................................................................................. 23
  4.2. Spatial Interface ....................................................................................................................... 23
    4.2.1. Standards ........................................................................................................................ 23
      4.2.1.1. Blob Size ................................................................................................................... 23
      4.2.1.2. Colors ...................................................................................................................... 23
      4.2.1.3. Sample Gestures ..................................................................................................... 24
    4.2.2. Tests ................................................................................................................................ 24
      4.2.2.1. Blob Comprehension ............................................................................................... 24
        4.2.2.1.1. Procedure ........................................................................................................... 25
        4.2.2.1.2. Pass Condition ................................................................................................. 25
      4.2.2.2. Initial Gesture Recognition ..................................................................................... 25
        4.2.2.2.1. Procedure ........................................................................................................... 25
        4.2.2.2.2. Pass Condition ................................................................................................. 25
      4.2.2.3. Percentage of Gesture Completed .......................................................................... 25
        4.2.2.3.1. Procedure ........................................................................................................... 25
        4.2.2.3.2. Pass Condition ................................................................................................. 26
      4.2.2.4. Certainty of Gesture Recognition ............................................................................ 26
      4.2.2.5. Gesture Composition ............................................................................................... 26
        4.2.2.5.1. Procedure ........................................................................................................... 26
        4.2.2.5.2. Pass Condition ................................................................................................. 26
      4.2.2.6. Practical Generation of Gestures .............................................................................. 26
        4.2.2.6.1. Procedure ........................................................................................................... 27
        4.2.2.6.2. Pass Condition ................................................................................................. 27
      4.2.2.7. Potential Gesture Recognition ............................................................................... 27
        4.2.2.7.1. Procedure ........................................................................................................... 27
        4.2.2.7.2. Pass Condition ................................................................................................. 28
      4.2.2.8. Gesture Storage ........................................................................................................ 28
        4.2.2.8.1. Procedure ........................................................................................................... 28
        4.2.2.8.2. Pass Condition ................................................................................................. 28
        4.2.2.8.3. Procedure ........................................................................................................... 28
        4.2.2.8.4. Pass Condition ................................................................................................. 29
5. Demo Tests ................................................................................................................................... 30
5.1. 3D Demo .................................................................................................................. 30
  5.1.1. Tests .................................................................................................................. 30
    5.1.1.1. Player Movement ....................................................................................... 30
      5.1.1.1.1. Procedure ...................................................................................... 30
      5.1.1.1.2. Pass Condition ............................................................................. 30
    5.1.1.2. Physics Gravity Test ............................................................................... 30
      5.1.1.2.1. Procedure ...................................................................................... 30
      5.1.1.2.2. Pass Condition ............................................................................. 31
    5.1.1.3. Ball Launcher Test ..................................................................................... 31
      5.1.1.3.1. Procedure ...................................................................................... 31
      5.1.1.3.2. Pass Condition ............................................................................. 31
  5.2. Spatial Demo Tests ............................................................................................... 31
    5.2.1. Basic Application Integration ...................................................................... 31
      5.2.1.1. Procedure ......................................................................................... 32
      5.2.1.2. Pass Condition ................................................................................. 32
    5.2.2. Gesture Recognition ................................................................................... 32
      5.2.2.1. Procedure ......................................................................................... 32
      5.2.2.2. Pass Condition ................................................................................. 32
    5.2.3. Vertex Selection Functionality .................................................................... 32
      5.2.3.1. Procedure ......................................................................................... 32
      5.2.3.2. Pass Condition ................................................................................. 33
    5.2.4. Vertex Merging Functionality ................................................................... 33
      5.2.4.1. Procedure ......................................................................................... 33
      5.2.4.2. Pass Condition ................................................................................. 33
    5.2.5. Face Creation Functionality ..................................................................... 33
      5.2.5.1. Procedure ......................................................................................... 33
      5.2.5.2. Pass Condition ................................................................................. 33
    5.2.6. 3D Transformation Functionality .............................................................. 33
      5.2.6.1. Procedure ......................................................................................... 34
      5.2.6.2. Pass Condition ................................................................................. 34
    5.2.7. 3D Manipulation Functionality ................................................................. 34
      5.2.7.1. Procedure ......................................................................................... 34
      5.2.7.2. Pass Condition ................................................................................. 34
    5.2.8. Menu Navigation Functionality ................................................................. 34
      5.2.8.1. Procedure ......................................................................................... 34
      5.2.8.2. Pass Condition ................................................................................. 35

Glossary ...................................................................................................................... 36
1. Infrastructure Acceptance Test

1.1. Programmer Interface

1.1.1. Interface Test Standards

Tests to test the programmer interface on both Linux and Windows. Both of these tests require a program which creates an instance of the subscriber class of the 3D Interface. Have the instance output "Success" to std::cout inside the Initialize() function.

1.1.2. Test the application in C++

Priority: 1 REQ: _language

1.1.2.1. Procedure

Write the above application in C++ following standards to compile under G++. Compile the application on a linux machine. Run the executable.

1.1.2.2. Pass Condition

The executable should output "Success".

1.1.3. Test Static Library on Linux

Priority: 1 REQ: _linux,_operating_system

1.1.3.1. Procedure

Compile the test program using G++ on Linux, compiling with the static linux version (*.so)library. Move the executable to a directory up from the .so file.

1.1.3.2. Pass Condition

The program outputs "Success" to the console.

1.1.4. Test Dynamic Library on Windows

Priority: 2 REQ: _windows,_operating_system

1.1.4.1. Procedure

Compile the test program using Visual Studios™ on Windows, compiling with the dynamic linked Windows version (*.dll) of the library.
1.1.4.2. Pass Condition

The program outputs "Success" to the console.

1.2. Threading

Tests to test the threading infrastructure of the library. For this scope, the thread manager is a public class you can use to access information about the threads being used in the library.

1.2.1. Library In A Different Thread

Priority: 2 REQ: _threading

1.2.1.1. Procedure

Create a subscriber instance with blank methods. Inside the Update() method, write a statement to retrieve the ID of the thread.

1.2.1.2. Pass Condition

The ID of the thread retrieved in the method above should be different than the ID of the thread being used to process the test.

1.2.2. MultiThreaded Library

Priority: 2 REQ: _multi_threading_in_the_library

1.2.2.1. Procedure

Obtain the number of running threads in the library from the library's thread manager when requesting the regression model from a data point (see the linear regression test for data point creation).

1.2.2.2. Pass Condition

The number of threads must be greater than 1.

1.2.3. Threading Priority

Priority: 3 REQ: _thread_priority

1.2.3.1. Procedure

Create a subscriber exactly as you did in the library threading test, but instead of writing a statement to retrieve the ID of the threads, make the statement retrieve the priority of the threads.
1.2.3.2. Pass Condition

The priority of the thread from the subscriber should be lower than that of the thread being used to process the test.

1.2.4. Number of Threads

Priority: 3 REQ: _number_of_threads

1.2.4.1. Procedure

Perform the multithreaded test on a machine with a 2 core processor.

1.2.4.2. Pass Condition

The number of threads must be equal to 2.

1.2.5. Compile Time Customization

Priority: 2 REQ: _compile_time_customization

1.2.5.1. Procedure

Perform the multithreaded test, but before requesting the regression model, tell the thread manager to use 5 threads. Obtain the number of running threads from the thread manager.

1.2.5.2. Pass Condition

The number of running threads should be 5.

1.2.6. PreCompilation Directives

Priority: 3 REQ: _pre_compilation_directives

1.2.6.1. Procedure

Set the threading preprocessor variable to FALSE. Compile the project. Perform the thread customization test.

1.2.6.2. Pass Condition

The number of running threads should be 1.
2. Data Acceptance Test

2.1. Data Importing

Tests to test the data import layer of the library

2.1.1. Import

Priority: 1
REQ: data_importing, a_href_g_input_frame_input_frames_a, web_cam_data, data_streamline

2.1.1.1. Procedure

Create a subscriber to the Spatial Interface. Set dot size to 9px. Set dot size tolerance to 2px. Set dot color to 0x00FF00. Set maximum number of dots to 10. Convert the below image to an image stream, and stream it into the engine as webcam input. Retrieve the input frame created.

2.1.1.2. Pass Condition

The input frame should have 6 points, each with an X and a Y position.

2.1.1.3. IR Dot Intensity

Priority: 1
REQ: data_streamline
This tests the intensity of the IR emitters attached to the player.

### 2.1.1.3.1. Procedure

1. Place the IR emitter in the area that the wii remote is viewing.

2. In a dark area, for every possible position in the wii remotes viewing angle, move the emitter and record the intensity of the dot.

3. Record the dot intensity at each location onto a chart that represents the viewable area.

4. Repeat step 2 with the exact same movements, but with different light sources in the room in addition to the emitter. These can include lamp light, fluorescent light, digital projector light, halogen light, plasma/LCD screen light, and sunlight.

5. Record the dot intensity and the number of dots seen onto a chart.

6. Using the results from step 3, compare the data from step 5 to make sure that the most intense dot seen is the correct one.

### 2.1.1.3.2. Pass Condition

This test will pass if every location recorded after step four has one clearly visible dot that can be differentiated from all of the background noise.

### 2.1.2. Input History

**Priority:** 1  
**REQ:** _data_history,_data_history_configuration

#### 2.1.2.1. Procedure

Create a mock input device class that provides incremental data every update event. ie, (1,1) then (2,2) continually for each update event. Set the mock input device's rate of input to be .25 seconds. Create a subscriber requesting data from this input device. Set its back history to 1 second. After 3 seconds, request all of this history for the data point passed in to the subscriber.

#### 2.1.2.2. Pass Condition

The history should have only the following values: 9

### 2.1.3. History Access Time

**Priority:** 1  
**REQ:** _data_history_access_time
### 2.1.3.1. Procedure

Create 2 data points. One with a history of length 1000000, and one with a history of length 10. Populate them fully with random data (do not time this). Time how long the following operations take for each data point.

- access history at half back (index 5 for the second data point and index 50000 for the first)
- add one history value to the history stack

### 2.1.3.2. Pass Condition

The time for both tests on both data points should be the same within 0.001 seconds.

### 2.2. Data Exporting

**Priority:** 1

**Requirement Matrix:** _data_exporting,_multi_tiered_exporting,_multiple_unique_devices

#### 2.2.1. Procedure

Feed in the following input data:

Object 1: \((1,2), (2,3), (3,3), (4,2), (4,2)\)

Object 2: \((3,4), (1,6), (2,8), (3,10), (4,10)\)

#### 2.2.2. Positional Data

**Priority:** 1

**Requirement Matrix:** _single_point_exporting,_positional_data,_list_of_positional_data

#### 2.2.2.1. Procedure

Use the procedure from Data Exporting to populate the history data for a data point. After each entry, retrieve the position of the [g.data_point]

#### 2.2.2.2. Pass Condition

It should return values in this order:

Object 1: \((1,2), (2,3), (3,3), (4,2), (4,2)\)

Object 2: \((3,4), (1,6), (2,8), (3,10), (4,10)\)
2.2.3. Velocity Data

Priority: 1

Requirement Matrix: _single_point_exporting,_velocity_data,_list_of_velocities_data

2.2.3.1. Procedure

Use the procedure from Data Exporting to populate the history data for a data point. After each entry, retrieve the velocity of the [g.data_point]

2.2.3.2. Pass Condition

It should return values in this order:

Object 1: (1, 1), (1, 0), (1, -1), (0, 0)
Object 2: (-2, 2), (1, 2), (1, 2), (1, 0)

2.2.4. Acceleration Data

Priority: 1

Requirement Matrix: _single_point_exporting,_acceleration_data,_list_of_accelerations_data

2.2.4.1. Procedure

Use the procedure from Data Exporting to populate the history data for a data point. After each entry, retrieve the acceleration of the [g.data_point]

2.2.4.2. Pass Condition

It should return:

Object 1: (0, -1), (0, -1), (-1, 1)
Object 2: (3, 0), (0, 0), (0, -2)

2.2.5. Gestures

Priority: 1

Requirement Matrix: _single_point_exporting,_gestures,_atomic_gestures

2.2.5.1. Procedure

Use the procedure from Data Exporting to populate the history data for a data point. Define an atomic gesture as being the top half of a circle. Retrieve the currently processed gestures.
2.2.5.2. Pass Condition

Object 1 should return the circle gesture outlined above. Object 2 will not return any mapped gesture.

2.2.6. Export Rate

Priority: 1

Requirement Matrix: _data_export_rate,_guaranteed_updates,_fastest_update

2.2.6.1. Procedure

Create a subscriber similar to the library thread test. Set the subscriber to request 20 update events per second. In the update method, count the number of updates received. Retrieve the number of updates after exactly 9 seconds.

2.2.6.2. Pass Condition

There must be 180 updates +/- 1 update.

2.3. Data Representation

Priority: 1 REQ: _data_representation

Tests to test the data representation layer of the library

2.3.1. Reference Frames

Priority: 2 REQ: _reference_frames,_data_access,_data_conversion

2.3.1.1. First Frame

2.3.1.1.1. Procedure

Feed three points in as the input as follows: (1,2), (1,3), (2,4)

Call The Method: FirstRefFrame();

2.3.1.1.2. Pass Condition

And it should have the frame with data (1,2)

2.3.1.2. Next Frame

2.3.1.2.1. Procedure

Same Procedure as First Frame Call The Method: NextFrame();
2.3.1.2.2. Pass Condition

And it should have the frame with data (1, 3)

2.3.1.3. Next N Frames

2.3.1.3.1. Procedure

Same Procedure as First Frame

Call The Method: NextFrame(2);

2.3.1.3.2. Pass Condition

And it should have the frame with data (2, 4)

2.3.1.4. Previous Frame

2.3.1.4.1. Procedure

Same Procedure as Next Frame

Call The Method: PrevFrame();

2.3.1.4.2. Pass Condition

And it should have the frame with data (1, 2)

2.3.1.5. Previous N Frames

2.3.1.5.1. Procedure

Same Procedure as Next N Frames

Call The Method: PrevFrame(2);

2.3.1.5.2. Pass Condition

And it should have the frame with data (1, 2)

2.3.1.6. Get Frame

2.3.1.6.1. Procedure

Call The Method: GetFrame(2); #indexed at 0
2.3.1.6.2. Pass Condition

And it should have the frame with data \((2, 4)\)

2.3.2. Quaternion To Roll Pitch Yaw

**Priority:** 2  
**REQ:** _rotation,_rotational_conversion

### 2.3.2.1. Procedure

Produce a single random quaternion. Then call: QuaternionToRPY()

### 2.3.2.2. Pass Condition

The resultant structure fits conforms to the following equation

\[
\begin{bmatrix}
\phi \\
\theta \\
\psi
\end{bmatrix} = \begin{bmatrix}
\arctan(2(q_0q_2 + q_1q_3)) \\
\arcsin(2(q_0q_3 - q_1q_2)) \\
\arctan(2(q_0q_1 + q_2q_3)) \\
1 - 2(q_0^2 + q_1^2)
\end{bmatrix}
\]

*Courtesy of Wikipedia*

2.3.3. Roll Pitch Yaw To Quaternion

**Priority:** 2  
**REQ:** _rotation,_rotational_conversion

### 2.3.3.1. Procedure

Same Procedure as Quaternion To Roll Pitch Yaw. Produce a single random quaternion. Then call: RPYToQuaternion()

### 2.3.3.2. Pass Condition

The resultant quaternion should match the initial quaternion in prior to the initial call of QuaternionToRPY() with random data.

2.4. Data Transformation

Tests to test the transformation of data within the engine.

### 2.4.1. Standard Data Sets

The following data sets will be referenced throughout this section.

- **Displaced Incremental**
  
  \((1,2),(2,3),(3,4),(4,5)\)

- **Circular**
  
  \((0,5),(1,5),(2,4),(3,3),(4,2),(4,1)\)
<table>
<thead>
<tr>
<th>Data Set Type</th>
<th>Data Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Displaced Decremental</td>
<td>(7,2)(6,3)(5,4)(4,5)</td>
</tr>
<tr>
<td>Post-Intersection Frame</td>
<td>(5,6) and (3,6)</td>
</tr>
<tr>
<td>Fast Incremental</td>
<td>(1,1)(3,3)(5,5)(7,7)</td>
</tr>
<tr>
<td>Incremental</td>
<td>(1,1)(2,2)(3,3)(4,4)</td>
</tr>
<tr>
<td>Decremental</td>
<td>(10,10)(9,9)(8,8)(7,7)</td>
</tr>
<tr>
<td>Small Incremental</td>
<td>(1,1)(2,2)</td>
</tr>
<tr>
<td>Exponential</td>
<td>(1,1)(2,2)(4,4)(7,7)</td>
</tr>
</tbody>
</table>

### 2.4.2. Test Linear Regression

**Priority:** 1  
**REQ:** _linear_regression_

#### 2.4.2.1. Procedure

Create a data point and fill its history with the displaced incremental data set. Query the data point for the regression model.

#### 2.4.2.2. Pass Condition

The regression model should be a line with a slope of 1 and a Y-intercept of 1.

### 2.4.3. Test Circular Regression

**Priority:** 1  
**REQ:** _circular_regression,_a_href_g_data_point_data_point_a_analysis_

#### 2.4.3.1. Procedure

Create a data point and fill its history with the circular data set. Query the data point for the regression model.

#### 2.4.3.2. Pass Condition

The regression model returned should be a circle with radius 5 and center at (0,0) (this test has an acceptance range of +/- 0.5 on both the radius and the center location).

### 2.4.4. Test Data Point Collision Resolution

**Priority:** 1  
**REQ:** _resolving_a_href_g_data_point_data_point_a_collision,_finding_next_value,_data_extrapolation_

#### 2.4.4.1. Procedure

Create a data point exactly as you did in the linear regression test, then create a data point in the same fashion, except use the displaced decremental data set. Simulate an input frame.
with 2 input points from the post intersection frame. Retrieve the current position of each data point.

### 2.4.4.2. Pass Condition

The first data point should be at location (5,6), while the second should be at (3,6).

### 2.4.5. Test Update Distance Tolerance

**Priority:** 2  
REQ: _value_update_distance_tolerance,_data_extrapolation

#### 2.4.5.1. Procedure

Create a data point (data point 1) exactly as you did in the linear regression test. Create another data point (data point 2) in the same fashion, except use the fast incremental data set. Retrieve the update distance tolerance from each data point.

#### 2.4.5.2. Pass Condition

The update distance tolerance from data point 1 should be less than that of data point 2.

### 2.4.6. Test Data Point Loss

**Priority:** 1  
REQ: _loss_of_a_href_g_data_point_data_point_a,_lost_point_prediction,_data_reading_rate

#### 2.4.6.1. Procedure

Create a data point (data point 1) exactly as you did in the linear regression test. Simulate an input frame with 1 input point, (10,1). Retrieve the current position of the data point.

#### 2.4.6.2. Pass Condition

The current location of the data point should be (5,6).

### 2.4.7. Test point removal and waiting queue

**Priority:** 1  
REQ: _lost_a_href_g_data_point_data_point_a_removal,_waiting_queue,_a_href_g_data_point_data_point_a_priorities

#### 2.4.7.1. Procedure

Create 2 data points in the same fashion as in the linear regression test, one with the incremental data set and one with the decremental data set. Set the priority of the first data point to 1, and the priority of the second to 2. Set both of their time out limits to be 3 events.
Simulate 3 input frames with 2 input points, both (0,0). Then Simulate a 4th input frame with 1 input point, (5,5). Retrieve the values of each data point.

### 2.4.7.2. Pass Condition

The first data point should have a position of (5,5), the second data point should have a position of (4,4).

### 2.4.8. Test Dimensional Smoothing

**Priority:** 1 **REQ:** _dimensional_smoothing

#### 2.4.8.1. Procedure

Create a data point exactly as you did in the linear regression test. Set its error bounds to be 2. Set its update distance tolerance to 3. Simulate an input frame with 1 input point, (5,8.5). Retrieve the current position of the data point.

#### 2.4.8.2. Pass Condition

The position of the smoothed data point should be (5,6.5).

### 2.4.9. Test Future Computation Using Raw Data

**Priority:** 2 **REQ:** _future_computation_using_extrapolated_data

#### 2.4.9.1. Procedure

Create a data point with the small incremental data set. Set its error bounds to 1, and its update distance tolerance to 3. Simulate an input event with an input point with the following value (3,5). Store its current position to a variable, A. Now simulate an input event with an input point with a value of (10,10). Retrieve its current position.

#### 2.4.9.2. Pass Condition

The value of the position stored in A should be (3,3,5), and the current position should be (4,6).

### 2.4.10. Test Derivative Computation

**Priority:** 3 **REQ:** _derivatives

#### 2.4.10.1. Procedure

Create a data point with the exponential data set. Retrieve the next position, as well as velocity and acceleration of the data point.
2.4.10.2. Pass Condition

The next position should be (11,11). The velocity should be (3,3). The acceleration should be (1,1).

2.5. Abstract Data Processing

Scenarios to test the gesture recognition ability.

2.5.1. Test Atomic Gesture

Priority: 1 REQ: _gestures,_atomic_gestures,_custom_gestures

2.5.1.1. Procedure

Define an atomic gesture to be a point moving horizontal from the users left to right for 6 inches. Add a listener for the gesture to fire an event when the gesture is detected. Perform the gesture.

2.5.1.2. Pass Condition

The event fires when the user simulates the hand gesture defined.

2.5.2. Test Complex Gesture

Priority: 2 REQ: _gestures,_complex_gestures,_custom_gestures

2.5.2.1. Procedure

Define an additional atomic gesture to be the mirror image of the one defined previously (a point moving from right to left for 6 inches). Define a complex gesture to be the simultaneous execution of both this gesture and that of the previous test. Add a listener for the gesture to fire an event is detected. Ensure there is no listener for the individual atomic gestures. Perform each atomic gesture individually, then perform them both at the same time (use both hands).

2.5.2.2. Pass Condition

The event fires only when both gestures are detected and not when a single gesture was detected.
3. Diagnostic Utilities Tests

3.1. Data Raw Graph

Priority: 2 REQ: _real_time_graphs,_raw_data

3.1.1. Procedure

Use the following input: (0, 0), (1, 2), (3, 2), (2, 3)

3.1.2. Pass Condition

3.2. Data Smoothed Graph

Priority: 2 REQ: _real_time_graphs,_smoothed_data

3.2.1. Procedure

Use the following input: (0, 0), (1, 2), (3, 2), (2, 3)

3.2.2. Pass Condition
3.3. Data Accuracy Graph

**Priority:** 3 REQ: _real_time_graphs,_data_accuracy

### 3.3.1. Procedure

Use the following input: \((0, 0), (1, 2), (3, 2), (2, 3)\)

### 3.3.2. Pass Condition
4. Interface Functionality Tests

4.1. 3D Interface

Priority: 1 REQ: _3d_interface

The 3D Interface section revolves around the testing of the three dimensional interface

4.1.1. Standards

4.1.1.1. Viewing Angle

The viewing angle of the wii remote is 45 degrees. This is determined by the placement of the hardware and is not user-configurable.

4.1.1.2. IR Dot Size

One of the many possible data values that can be retrieved from the wii remote is dot size. This is a representation of the magnitude of the IR dot. The magnitude of the IR dot is perceived completely by the wii remote, and is determined by the quality and output rating of the IR LEDs attached to the player.

4.1.2. Tests

4.1.2.1. Tilt Recognition

Priority: 1 REQ: _3d_interface,_dimensional_smoothing,_raw_data_storage

This tests the library's ability to compensate for movements that often get mis-interpreted. When the plane that supports the two IR dots is tilted sideways, it makes the camera think that the distance between the two dots has just decreased dramatically. Without proper smoothing, this would result in a false reading of a large movement towards the wii remote.

4.1.2.1.1. Procedure

1. Place the IR emitters on a pivot-able surface in front of the wii remote.
2. Start to record the 2D digital position determined by the library.
3. After 5 seconds, rotate the emitters quickly by 20 degrees.
4. Immediately rotate the emitters back to their original orientation.
4.1.2.1.2. Pass Condition

This test passes if the recorded 2D location does not make a large jump towards the end of the recorded data.

4.1.2.2. Comparison of Digital Results to Real World Positions

**Priority:** 1  
**REQ:** _3d_interface,_dimensional_smoothing

This test ensures that the library can correctly model a user's movements digitally.

4.1.2.2.1. Procedure

1. Set up a test user in a set position in the wii remote's viewable area.
2. Pre-define a multiple patterns for the user to talk on.
3. Have the computer record the person digitally and have a video camera record the user visually. Use a timecode system so that the data can be compared later.

4.1.2.2.2. Pass Condition

This test passes if the digital location data matches up to the visual locations at 3 second intervals.

4.2. Spatial Interface

**Priority:** 1  
**REQ:** _spatial_tracking_interface

The sections in "Spatial Interface" concern themselves with the correctness and testability of the spatial interface.

4.2.1. Standards

4.2.1.1. Blob Size

This quantity is dependent on the hardware viewing the blob, and as such, will not be referenced in the tests below.

4.2.1.2. Colors

The following table maps the common usage of color names (red, yellow, etc.) to the specific hues that will be used in the testing procedures.

<table>
<thead>
<tr>
<th>Informal Color Name</th>
<th>Pantone™ Matching System Color</th>
<th>RGB triplet</th>
</tr>
</thead>
<tbody>
<tr>
<td>yellow</td>
<td>Pantone Yellow</td>
<td>#fce016</td>
</tr>
<tr>
<td>Informal Color Name</td>
<td>Pantone™ Matching System Color</td>
<td>RGB triplet</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>red</td>
<td>Red 032</td>
<td>#ef2b2d</td>
</tr>
<tr>
<td>blue</td>
<td>PMS 293</td>
<td>#0051ba</td>
</tr>
<tr>
<td>green</td>
<td>PMS 368</td>
<td>#5bbf21</td>
</tr>
</tbody>
</table>

4.2.1.3. Sample Gestures

This section describes several different atomic and complex gestures for use in the tests below.

- **Atomic Gesture 1 (AG1):** A yellow blob, one foot away from the camera and raised four feet off the ground, moves to another point three feet off of the ground. It takes two seconds to do so.

- **Atomic Gesture 1R (AG1R):** A gesture with the same specification as AG1, but the movement steps are performed in reverse.

- **Atomic Gesture 2 (AG2):** Same specification as AG1, with the difference that the blob is blue instead of yellow.

- **Atomic Gesture 3 (AG3):** A yellow blob, one foot away from the camera and raised four feet off the ground, moves to another point one foot to the left of its original position. It takes two seconds to do so.

- **Atomic Gesture 4 (AG4):** A yellow blob, one foot away from the camera and raised four and a half feet off the ground, rotates in a circle of radius half a foot around a point half a foot below it. It takes four seconds to do so.

- **Atomic Gesture 5 (AG5):** Same specification as AG4, with the difference that the gesture takes one second to complete instead of four.

- **Atomic Gesture 6 (AG6):** Same specification as AG5, with the difference that the blob is blue instead of yellow.

- **Atomic Gesture 7 (AG7):** Same specification as AG5, with the difference that the blob is red instead of yellow.

- **Complex Gesture 1 (CG1):** A combination of the following gestures in succession: AG1, AG1R, AG4.

4.2.2. Tests

4.2.2.1. Blob Comprehension

**Priority:** 1 REQ: _gesture_specification_, _atomic_gestures_, _complex_gestures_
This test is run to ensure that blobs used for the invocations of gestures can be differentiated from the rest of the environment that the blobs are captured in.

### 4.2.2.1. Procedure

1. For each test color in Colors, create an artificial background to the web camera's view. This background will be a flood fill of a color that is a set quantity away from the test color in hue, saturation, and brightness.

2. For each test color, create a programmatic version of gesture AG1 with the color replaced with the test color.

3. Perform each version of AG1 in front of the artificial background for that color.

### 4.2.2.1.2. Pass Condition

This test will pass if each blob of a different color is recognized in front of the artificial background behind it to within a certain percentage of confidence.

### 4.2.2.2. Initial Gesture Recognition

**Priority:** 1  
**REQ:** _gesture_library, gesture_specification, atomic_gestures, complex_gestures_

This test ensures that gestures, when initially being invoked in front of the web camera, are recognized and begin to get tracked from that point onward.

#### 4.2.2.2.1. Procedure

1. Specify the sample gestures as described in Sample Gestures.

2. For each atomic gesture specified in the sample gestures, perform the first point-to-point translation of the gesture's steps.

#### 4.2.2.2.2. Pass Condition

This test passes if, after the invocation of every partial gestures, the library asserts that a gesture is being performed, and begins tracking it.

### 4.2.2.3. Percentage of Gesture Completed

**Priority:** 2  
**REQ:** _gesture_library, gesture_specification_

This test ensures that the library can correctly identify, based on the most promising guess of what a half-completed gesture is, the percentage of the gesture that has been completed.

#### 4.2.2.3.1. Procedure

1. Specify the sample gestures as described in Sample Gestures.
2. For each gesture, perform a measurable percentage of the gesture, monitoring the internal state of the library for each invocation.

4.2.2.3.2. Pass Condition

This test passes if the measured percentage of each gesture performed shows up as the internally measured percentage in the library.

4.2.2.4. Certainty of Gesture Recognition

The following tests are written with the intention of ensuring the certainty of gesture recognition.

4.2.2.5. Gesture Composition

Priority: 1
REQ: _gesture_library, _gesture_specification, _atomic_gestures, _complex_gestures

This test ensures that defined gestures can be defined through the basic composite elements defined in the Spatial Interface API. Gestures are defined as a series of GestureStep s, which are each composed of a series of point positions and vectors (directions and speeds) to the next GestureStep. It also ensures that any gesture that can be composed as described in the specification can be programatically generated.

4.2.2.5.1. Procedure

1. Specify the sample gestures as described in Sample Gestures.

2. For each gesture, define the individual components that make up the gesture.

3. Store each gesture as a combination of its individual components, and ask the diagnostics utilities to recreate the gesture on-screen.

4.2.2.5.2. Pass Condition

This test passes if the recreated gesture on-screen is an appropriate approximation of the defined gesture. This is an eyeball test, and thus must either be performed manually or through the use of pattern recognition on the part of the testing utility. If the gesture in question is a complex gesture, then the eyeball test is performed at the end of the gesture's invocation, and thus would be an image of the last atomic gesture performed. Thus, all complex gestures should be tested at the end of every internal atomic gesture.

4.2.2.6. Practical Generation of Gestures

Priority: 1
REQ: _gesture_library, _gesture_specification, _atomic_gestures, _complex_gestures
This test ensures that a gesture may be generated practically — that is, through an emulation of the gesture the developer wishes the library to store. Thus we are testing the ability of the library to both interpret and generate gestures.

4.2.2.6.1. Procedure

1. Specify the sample gestures as described in Sample Gestures.

2. For each gesture, perform the gesture in front of the web camera as specified.

3. Allow the software to attempt to interpret and provide the programmatic interpretation for each gesture.

4.2.2.6.2. Pass Condition

This test passes if the programmatic interpretation for each gesture is within a low deviation percentage when compared to the developer-written versions of each gesture. This percentage must be constant across gestures: the program must interpret each gesture just as accurately as the next. In the case of complex gestures, each individual atomic gesture that makes it up must be within the low deviation percentage: the average low deviation percentage of the complex gesture is not an acceptable substitute.

Especially important in this test is the color of each individual gesture point. The camera's ability to find and track distinct blobs of color will be tested here as each several gestures will be performed with no difference except in the color used to perform them.

4.2.2.7. Potential Gesture Recognition

Priority: 1  REQ: _gesture_library,_gesture_specification,_atomic_gestures,_complex_gestures

This test ensures that the library correctly identifies alternate gestures that a currently-invoked gesture might end up being. For example, if there are two gestures, one shaped like a V, and one shaped like a W, and the first two strokes are invoked, the gesture may either be a V or a W. Both of these gestures should exist in the potential gestures array while the command is being invoked.

4.2.2.7.1. Procedure

1. Specify the sample gestures as described in Sample Gestures.

2. Group complex gestures by common first, intermediary, and end atomic gestures, and atomic gestures by common first, intermediary, and end gesture parts.

3. Perform, for each group above, all the gestures in the group.
4.2.2.7.2. Pass Condition

This test passes successfully if every gesture performed within a group is recognized as potentially every other gesture in the group to within a certain threshold. For the test to be successful, the threshold must be constant across gestures in a group. Different groups may have different thresholds, and thresholds may be different depending upon whether or not two gestures have a common movement at the beginning, middle, or end of their invocations.

4.2.2.8. Gesture Storage

**Priority:** 1  
**REQ:** gesture_library, gesture_specification, atomic_gestures, complex_gestures

This test ensures that the library can correctly store programmatically defined gestures.

4.2.2.8.1. Procedure

1. Specify the sample gestures as described in Sample Gestures.

2. For every gesture, determine a programmatic definition that meets the ability of the library to store it.

3. Store each gesture in a runtime instance of the library.

4. For each gesture, perform it with several random mutations to ensure that the library can corroborate its runtime storage with the analysis of the interpreter — that is, when the interpreter recognizes that a gesture is being performed, the library can tell which gesture it is.

4.2.2.8.2. Pass Condition

This test passes successfully if every gesture performed is recognized by the runtime instance of the library.

4.2.2.8.3. Procedure

1. Specify the sample gestures as described in Sample Gestures.

2. For each gesture, determine a programmatic definition that meets the ability of the library to store it.

3. For each gesture, change one part of the invocation that is not realistically possible. This may entail changing a color halfway through a gesture, or a starting point not being in a realistic place after the transformation of the previous point.

4. For each gesture, ensure that the library throws an error on the attempt to store the impossible gesture changes.
4.2.2.8.4. Pass Condition

This test passes successfully if the library refuses to store gestures that are realistically impossible.
5. Demo Tests

5.1. 3D Demo

Priority: 1 REQ: _3d_demo

The 3D Demo section revolves around the testing of the three dimensional demonstration application.

5.1.1. Tests

5.1.1.1. Player Movement

Priority: 1 REQ: _3d_demo

This tests the demo's ability to correctly move the in-game avatar with respect to the player's real world movements.

5.1.1.1.1. Procedure

1. Initiate a new instance of the demo.

2. Have the user move around.

3. Compare the user's movements to the avatar's movements on screen.

5.1.1.1.2. Pass Condition

This test will pass if the on-screen avatar moves in the same ways as the user.

5.1.1.2. Physics Gravity Test

Priority: 1 REQ: _3d_demo

This tests the dynamics engine. It should be able to correctly calculate the trajectory and velocity of a tennis ball.

5.1.1.2.1. Procedure

1. Create an instance of the demo with a single ball floating in the air at a height of 5 meters.

2. Have the demo keep track of when the ball hits the ground, and have it store this value in a text file.

3. It should take approximately 1.01015 seconds for it to hit the ground. Assuming \( g=9.8\text{ m/s}^2 \)
4. Repeat this demo for heights of 2 and 9 meters. The results for those should be 0.6389 and 1.355 respectively.

5.1.1.2.2. Pass Condition

This test passes if the recorded times are equal to the real world values.

5.1.1.3. Ball Launcher Test

Priority: 1 REQ: _3d_demo

This tests the ball launcher's accuracy and ability to fire in multiple directions.

5.1.1.3.1. Procedure

1. Create an instance of the demo with the AI ball launcher enabled in continuous mode.
2. The launcher should rotate a few degrees and then fire.
3. It should repeat until the demo is ended.
4. Create an instance of the demo with the AI ball launcher enabled in single mode.
5. The ball launcher should sit dormant until the ball launch button on the controller is pressed.
6. Once the button is pressed, the launcher should rotate and then launch just one ball.
7. repeat the last step 100 times.

5.1.1.3.2. Pass Condition

This test passes if the ball launcher does the required actions and does not fire balls outside of the "in bounds" area.

5.2. Spatial Demo Tests

Priority: 1 REQ: _spatial_demo

The purpose of the spatial demo is to prove the functionality of the spatial interface.

5.2.1. Basic Application Integration

Priority: 1 REQ: _blender

This test ensures that the application used to interface with the Clementine library (currently the three-dimensional modelling program "Blender") recognizes basic signals from the library.
5.2.1.1. Procedure

1. Generate Python hooks for the current build of the library, and connect them to the exposed API of the application.

2. Run the application with the integrated hooks and ensure it is listening for signals from the library.

5.2.1.2. Pass Condition

This test passes if the application appropriately listens for signals from the library.

5.2.2. Gesture Recognition

Priority: 1 REQ: _gestures,_blender

This test ensures that the application is able to sample and recognize gestures provided to it by the library.

5.2.2.1. Procedure

1. Specify the sample gestures as described in Sample Gestures.

2. Programmatically generate all of the gestures into the runtime library.

3. Attempt to invoke each gesture.

5.2.2.2. Pass Condition

This test passes if the application recognizes all gesture invocations that are performed. Simple dialog messages may be used, instead of complex mappings to commands within the application.

5.2.3. Vertex Selection Functionality

Priority: 1 REQ: _gestures,_vertex_selection

This test ensures that the application is able to move and select individual vertices.

5.2.3.1. Procedure

1. Specify the sample gestures as described in Sample Gestures.

2. Programmatically generate appropriate gestures to hold and translate vertices.

3. Attempt to invoke these gestures on a sample model.
5.2.3.2. Pass Condition

This test passes if the vertex is appropriately translated through the invocation of the defined gestures.

5.2.4. Vertex Merging Functionality

Priority: 2 REQ: _gestures,_vertex_merging

This test ensures that the application is able to select two vertices and merge them.

5.2.4.1. Procedure

1. Specify the sample gestures as described in Sample Gestures.
2. Programmatically generate appropriate gestures to select and merge vertices.
3. Attempt to invoke these gestures on a sample model.

5.2.4.2. Pass Condition

This test passes if the selected vertices are appropriately merged through the invocation of the defined gestures.

5.2.5. Face Creation Functionality

Priority: 1 REQ: _gestures,_face_creation

This test ensures that the application is able to create a new face from vertices separate from the model being edited.

5.2.5.1. Procedure

1. Specify the sample gestures as described in Sample Gestures.
2. Programmatically generate appropriate gestures to create vertices, select vertices, and create faces from selected vertices.
3. Attempt to invoke these gestures on a sample model.

5.2.5.2. Pass Condition

This test passes if the vertices created in the model are selected and created into a face.

5.2.6. 3D Transformation Functionality

Priority: 1 REQ: _gestures,_3d_transformations
This test ensures that the application is able to select and manipulate a 3D object (through rotation, translation, and scaling).

### 5.2.6.1. Procedure

1. Specify the sample gestures as described in Sample Gestures.
2. Programmatically generate appropriate gestures to select and manipulate 3D objects.
3. Attempt to invoke these gestures on a sample model.

### 5.2.6.2. Pass Condition

This test passes if the application is able to select and manipulate 3D objects through the specifications above.

### 5.2.7. 3D Manipulation Functionality

**Priority:** 2  
**REQ:** _gestures,_3d_manipulation

This test ensures that the application is able to select and manipulate a 3D object (through extrusion and lathing).

#### 5.2.7.1. Procedure

1. Specify the sample gestures as described in Sample Gestures.
2. Programmatically generate appropriate gestures to select and manipulate 3D objects.
3. Attempt to invoke these gestures on a sample model.

#### 5.2.7.2. Pass Condition

This test passes if the application is able to select and manipulate 3D objects through the specifications above.

### 5.2.8. Menu Navigation Functionality

**Priority:** 2  
**REQ:** _gestures,_menu_navigation

This test ensures that the application is able to navigate through its own menus.

#### 5.2.8.1. Procedure

1. Specify the sample gestures as described in Sample Gestures.
2. Programmatically generate appropriate gestures to navigate the menus of the application.
3. Attempt to invoke these gestures on an instance of the application.

5.2.8.2. Pass Condition

This test passes if the application is able to navigate its menus based on the gestures programmed into it in the test specification.
Glossary

2D Projection
   The representation of a 3D environment projected onto a 2D plane.

Angle Axis
   An alternate structure for storing rotational data. It is defined by a vector and a rotation about that vector.

Avatar
   The object in an application created by the developer that the user directly interacts with/controls.

Blob
   A group of similarly colored pixels found in a frame of web camera input, defined by an area in pixels.

Bluetooth
   Bluetooth is a short-distance wireless protocol for portable devices. It is used within the scope of our project, as in the Wii Remote, to provide wireless connectivity between devices and a computer.

Circular regression
   Finding a circle that most closely represents the trends of a set of independent points of data.

Context-sensitive
   An action that is different depending on the current area of interaction. For example: a context menu, when requested on a file, shows "Delete", "Rename", "Edit", et cetera.

Data Point
   A data point is a point being tracked by the engine. Its values are what are reported to the subscriber.

Derivative
   The slope of a line. The rate of change in a function over time.

Developer
   The target audience for our library. This is the person that will develop custom applications for users to use.
Error Bounds
   The amount of error tolerated for a certain type of input. When the input

Extrapolation
   Estimating the next value of an input event based on previous data.

Extrusion
   Extrusion is an operation performed on an object in three-dimensional modeling. A plane of an object,
   when extruded, expands out or in from its original location, while its edges are still attached to the
   connecting edges of the object.

Gimble lock
   An event that occurs when using primitive forms of rotational representation. When rotating 2 of
   the 3 dimensions at the same time, the 3rd will become locked and any rotation on that axis will be
   nullified.

Global Positioning System/Service
   A service that provides the position on the earth of a receiver. It uses a minimum of 3 satelites to
   triangulate surface position. The civilian service is accurate to within 3-5 meters.

Graphics Library
   A library containing algorithms and classes for displaying 3D environments on a screen.

Infrared
   Infrared is a form of light invisible to human vision. It is commonly used to provide basic
   connectivity, wirelessly, between devices, such as television remotes.

Input Frame
   An input frame is a set of input points that represent all of the read points coming from the input
   device at that time.

Input Point
   An input point is an XY value for a point obtained from an input device.

Lathing
   Lathing is an operation performed on a two-dimensional plane. The final result of lathing is a three-
   dimensional shape whose edge entirely around is the same shape as the external edge of the two-
   dimensional plane.
Library
A library is a collection of classes that can be used by a programmer when writing an application. A library usually performs a common or complex function that many applications can use.

Line of best fit
The line that best represents a grouping of correlated or semi-random data. It is computed using linear regression.

Linear regression
Finding a line that most closely represents the trends of a set of independent points of data.

Multipoint object
Multipoint Object - an object that contains multiple data points, all statically positioned.

Parallax Effect
The parallax effect is a phenomenon that occurs when an object is viewed from two viewpoints causing the object to appear at its proper depth. In the scope of our project, when an object moves and rotates appropriately opposite on screen to the viewers position in the room, an illusion of depth is created.

Precompilation Directive
A preprocessor directive is code that is interpreted as a compiler compiles code. These segments tell the compiler what sections of code to compile and how to compile them. Examples are #ifndef … #endif.

Preprocessor Variable
A variable that can be set in a compilation call, and can be used by percompilation directives.

Quaternions
Quaternions are the standard structure for storing rotational data. They are immune to gimble lock.

Reference frame
A reference frame is a set of axes within which a position in space can be measured.

Regression
A statistical method for finding the mean of a data set.
Smoothing
Smoothing in the scope of this project refers to the removal of outliers from a data set in order to yield a more consistent function, as well as bounding accurate data to within an error range to consider it acceptable data.

STI
The STI, or Spatial Tracking Interface, is a set of functionality that allows programmers to provide manipulation of their application through observable gestures that the user makes. See Spatial Tracking Interface for more information.

Subscriber
A subscriber is an agent that requests data point values from the library.

Thread
A thread is a process in a computer. A single processor can only process one thread at a time, though processed threads are switched rapidly in most modern computers. Having multiple threads allows for a long running process to run independent of a quicker one. Multi-core machines take advantage of threads by executing multiple threads at the same time.

User
The target audience of our developer. This is the person that will use the custom application that the developer creates.

Variance
The variance of a random variable is the average of the squared distance of its values from the statistical mean.

Wii Remote
A Wii Remote is an input device created by Nintendo for their Wii video game console. It has several buttons for digital manipulation, as well as an accelerometer, for measuring relative velocity, Bluetooth for connectivity, and infrared sensors for measuring contextual position.

Yaw-Pitch-Roll
An alternate structure for storing rotational data. It is defined by a rotation amount in each of the 3 axes, X, Y and Z, or Pitch, Yaw, and Roll, respectively. This form of representation is vulnerable to gimble lock.