Bug Report: “Stuttering Input”

We have noticed that Unity 4.5.4 presents stuttering input to the running game. By “stuttering” we mean that the touch coordinates do not follow the smooth motion of the finger but lurch between points, sometimes remaining static for several frames. We have found this problem to be reproducible across multiple devices and iOS versions. The problem was not present in Unity 3.5.6.

Date: 27th October 2014
Platform: iOS (7 and 8)
Affected Unity ver: 4.5.4
Reporter: James Gratton from Yakuto

Background

We have a game in production on iOS, “Table Tennis Touch”. Our game translates the player’s finger position into the position of a table tennis bat in our game world. The game’s success hinges on the accurate translation of finger to bat position and then the bat tracking the finger as it moves across the screen.

For various operational reasons we remained on Unity 3.5.6 until recently. We accepted that the interface between iOS and Unity was solid. Last month we upgraded to Unity 4.5.4 where we immediately discovered that said interface is no longer solid; in fact, the interface is so unstable that we have been forced to revert to Unity 3.5.7 as our game became unplayable on 4.5.4.

In Unity 4.5.4 the finger tracking (touch) interface between iOS and Unity frequently misses inputs resulting in incorrect touch data being reported in Unity.

Given the popularity of Unity on iOS it seems likely that the problem must lie in our game. In this report we will remove our game and isolate the problem to prove that an issue exists in Unity 4.5.4 that presents stuttering input to the game code.

Test projects

We have created two test projects: one for Unity 3.5.6 and one for Unity 4.5.4. Both projects aim to achieve the same result: to demonstrate that the touch input received from iOS is accurate (3.5.6) or broken (4.5.4).

It was intended that each project be identical; however, due to slight compatibility issues between the two Unity versions, they differ slightly. We believe the differences are not sufficient to affect test results.

In order to demonstrate that Unity is not missing touch input due to skipping frames, we have:

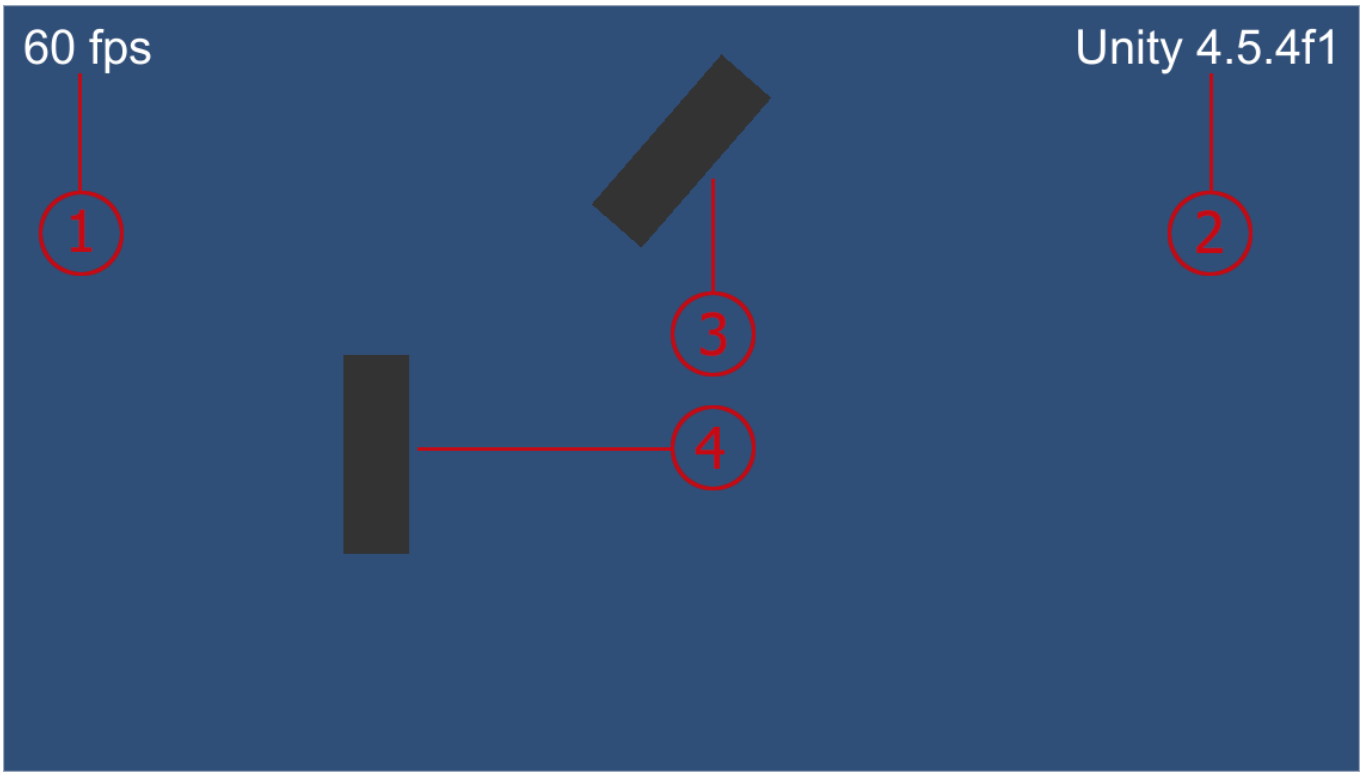
- kept the projects very simple;
- provided a rudimentary frame rate label;
- provided a rotating plane that is unaffected by the touch input; this provides a visual reference that the framerate is constant.

Project structure

Both projects contain four elements:

1. Frame rate label	Shows the framerate at which the test is running.
2. Unity version label	Identifies the version of Unity being tested.
3. Rotating plane	Provides a visual reference to the consistency of the framerate. This plane is unaffected by user input and will always rotate.
4. Translating plane	Only visible when the user is touching the screen. It should track the user's finger movement, translating only in the screen's X axis. Its movement will remain smooth for Unity 3.5.6 and will become stuttery/erratic for Unity 4.5.4

Note: the test projects are designed to work at 1136 x 640, i.e. iPhone 5,c,s



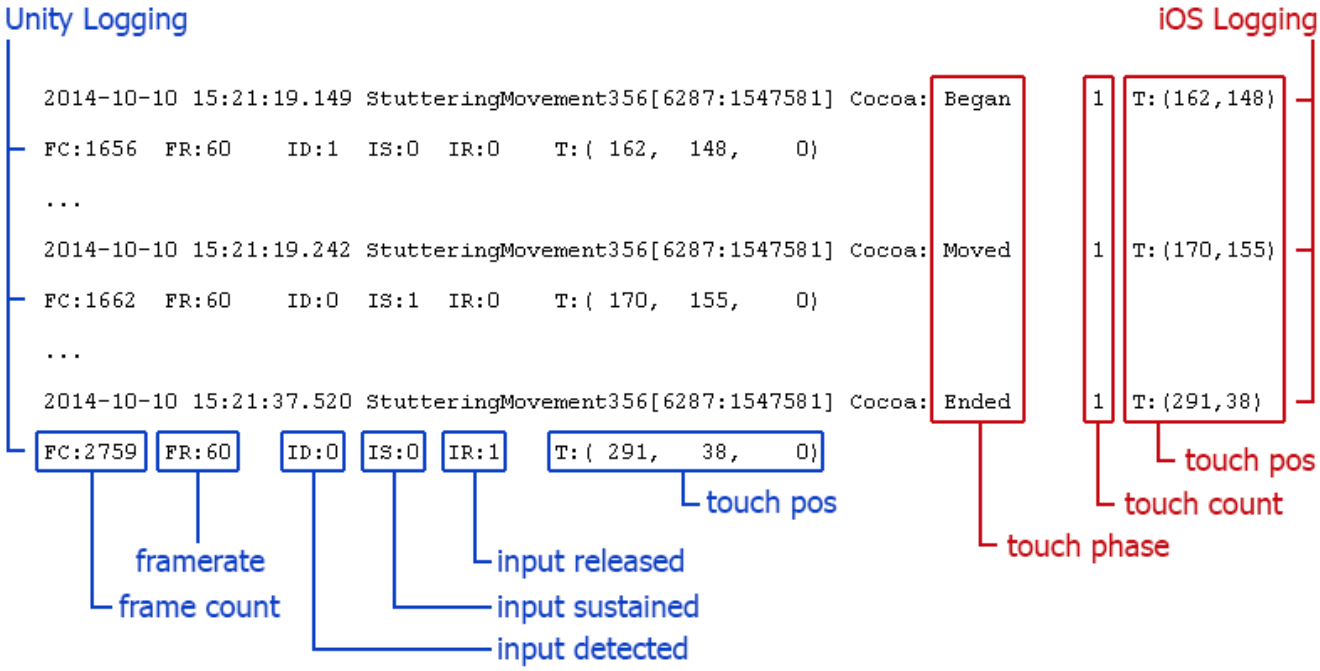
Project output

Both projects have two outputs: one visual and one textual.

The visual output shows three elements:

1. the framerate that is consistently at, or very close to, 60fps via the framerate label.
2. the rotating plane remaining smooth, demonstrating a consistent framerate.
3. the translating plane tracking the user's finger; this will be either smooth (3.5.6) or stuttering (4.5.4)

The textual output is written to the Xcode console when the project is being debugged by Xcode. There are two types of log entries, as demonstrated below: iOS logging and Unity logging.



The iOS logging is written from either the *EAGLView* (3.5.6) or the *UnityView* (4.5.4) in the touchesBegan, touchesEnded, touchesCancelled and touchesMoved event handlers. This is immediately before the calls to the *UnitySendTouches** methods. We should therefore expect a one-one match of iOS log entry to Unity entry, as per the figure above.

Test results

We have run the tests many times on various devices (see later) with iOS 7.1.1 and 8.0.2, and with Unity 3.5.6 and Unity 4.5.4. The results appear conclusive: with Unity 3.5.6 there is very occasionally a little input stutter but it always recovers; on Unity 4.5.4 when input stutter begins, the stuttering is never corrected, remaining a constant problem for any game that relies on accurate input.

Visual results

We have recorded four videos that demonstrate smooth and stuttering input. All were recorded against the same iPhone 5 on iOS 8.0.2 (at the time the latest iOS).

We made the recordings via external camera rather than via screen capture to avoid placing extra load on the device and so affecting the tests.

All videos were recorded at 60fps and played back at 30fps, i.e. they run at half speed to allow more time to observe the smoothness/stuttering of the input. We have also included demonstrations of the problem in production:

Production Demo 01 Unity 3.5.6	This capture shows our production game running on **Unity 3.5.6**. The input is smooth and the game plays as intended.
Production Demo 02 Unity 4.5.4	This capture shows our production game running on **Unity 4.5.4**. The input begins smoothly but begins to stutter. The game is near impossible to play. Note the way the bat jumps from one position to the next rather than smoothly following the finger.
Unity Test 01 Unity 3.5.6	This capture shows the **Unity 3.5.6** test project running. The (small) frame rate label shows ~60fps and the rotating plane remains smooth, confirming that frames are not being skipped. The translating plane follows the finger smoothly. This is the intended behaviour.
Unity Test 02 Unity 4.5.4	This capture shows the **Unity 4.5.4** test project running. The frame rate label shows ~60fps and the rotating plane remains smooth, confirming that frames are not being skipped. The translating plane begins following the finger smoothly but its movement soon becomes stuttery illustrating the problem.

The visual results offer compelling evidence that Unity 4.5.4 does not provide smooth touch input to games running on iOS.

Textual results

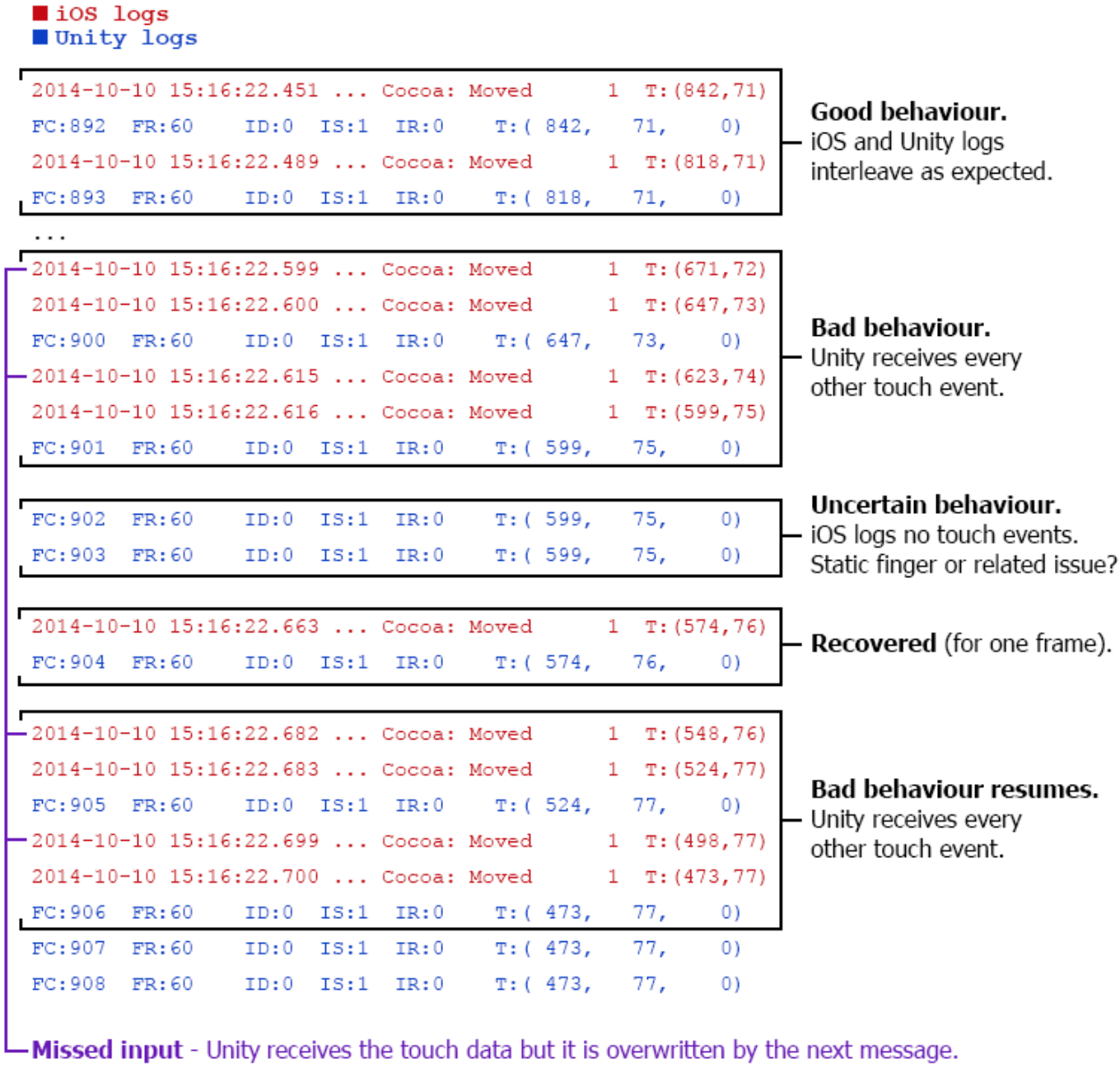
The two Unity tests above were run with the Xcode debugger attached. The logs were captured. There is a notable difference in the data recorded from Unity 3.5.6 and 4.5.4.

For Unity 3.5.6 there is generally a one-one match between iOS originated log entries and those from Unity. This suggests that every input message received by iOS is passed to Unity and made available to the game, resulting in smooth input tracking. We say “generally” as there are odd occasions where the bad behaviour described below occurs; however, it is not persistent, recovering almost immediately.

For Unity 4.5.4 we may or may not begin with the good behaviour described above but the one-one match between iOS originated log entries and those from Unity soon breaks down. We start to see two or more iOS entries per Unity entry then periods of no iOS input whilst Unity reports static input.

Note: There are no iOS log entries when the finger is static. In iOS there is no event that fires when the finger has made contact but remains stationary. Unity does report a static touch (presumably because it knows that the finger has not been removed and so reports its last known position).

An excerpt of the logs for [Unity Test 02] (Unity 4.5.4) is shown below:



The log clearly shows that Unity is missing input, for example, at:

- 15:16:22:599 iOS tells Unity that the finger is at (671,72)
- 15:16:22:600 iOS tells Unity that the finger is at (647,73)
- FC (frame count) 900 Unity reports the touch at (647,73)

We understand that Unity ignores multiple calls to *UnitySendTouchesMoved* in the same frame so (671,72) is overwritten by (647,73), i.e. Unity misses 24 pixels of finger travel in the X axis. We believe such behaviour is responsible for, or at least contributes to, the stuttering input demonstrated in the test videos.

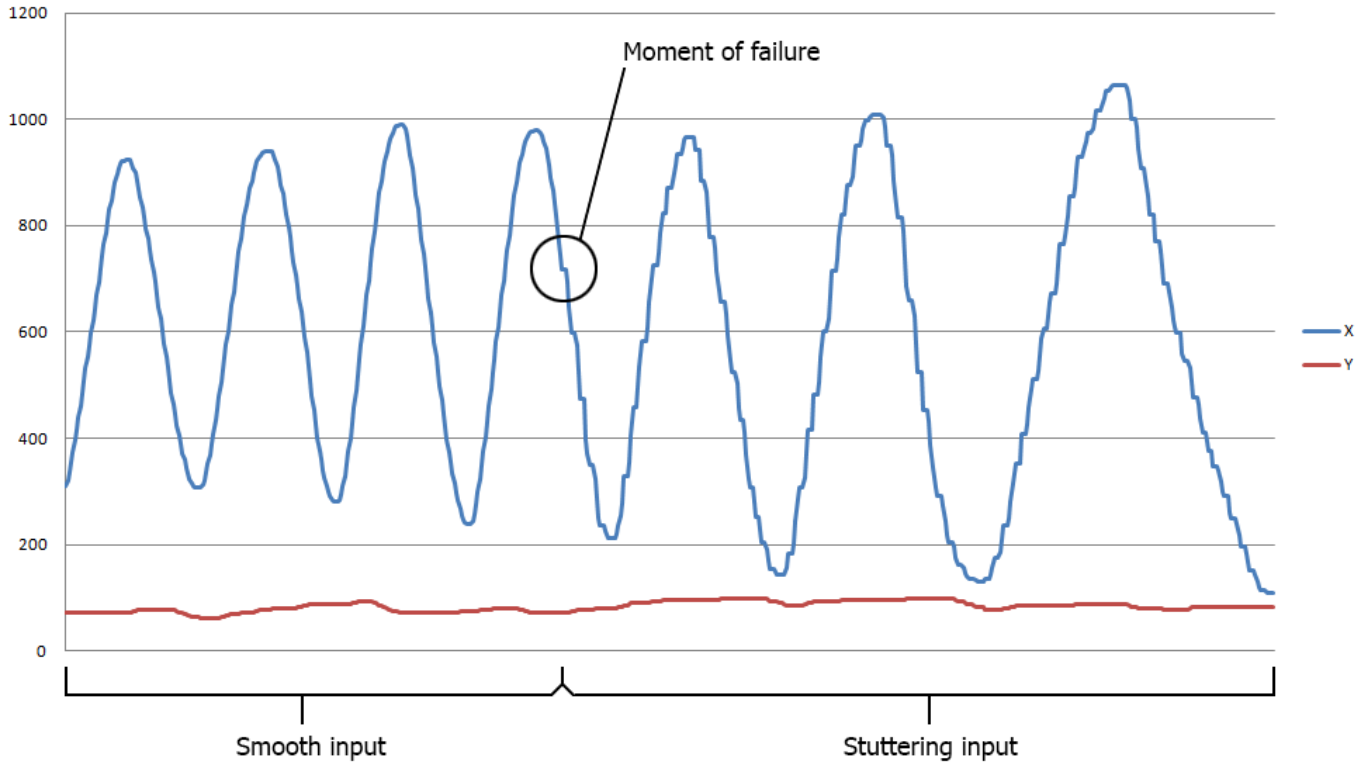
Note: The time between the problem iOS entries above is tiny followed by relatively long periods of inactivity.

Missing input continues throughout our 4.5.4 logs, following the similar pattern of:

- iOS move event
- iOS move event
- Unity input detected
- Unity input detected (static)
- Unity input detected (static)
- iOS move event
- Unity input detected
- repeat, varying the number of contiguous iOS move events, static inputs etc.

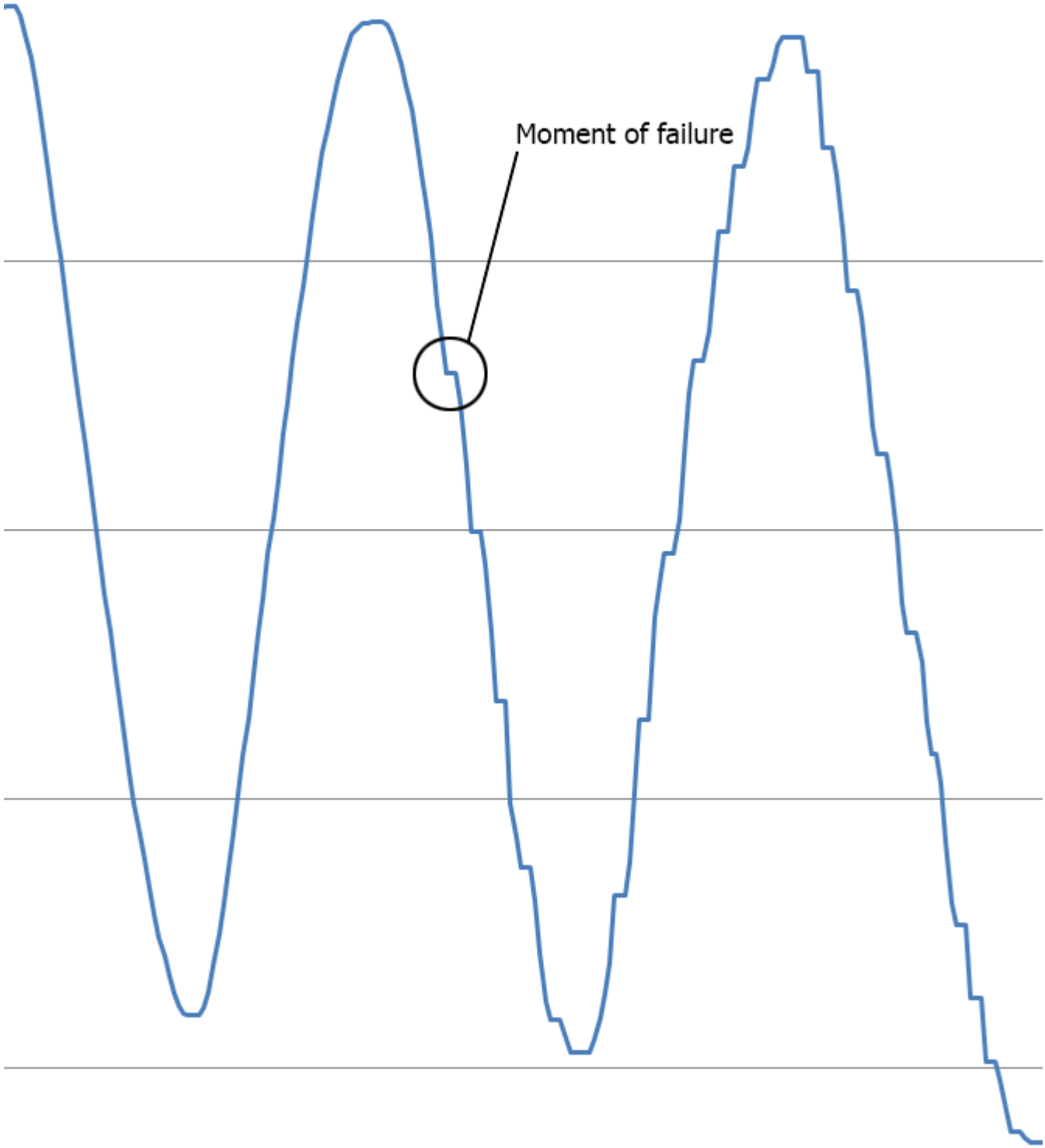
Plotting the coordinates demonstrates the problem very clearly.

The chart below plots the (X,Y) coordinates logged by Unity. It shows the moment in video Unity Test 02 where the input changes from being smooth to stuttering. It is notable how the X waveform becomes jagged, plateauing as the X position remains static before continuing to track the touch input.



Note: The Y axis also exhibits the same problem; however, in this test the finger was moved in a preominantly left-right motion, i.e. most movement was recorded in the X axis.

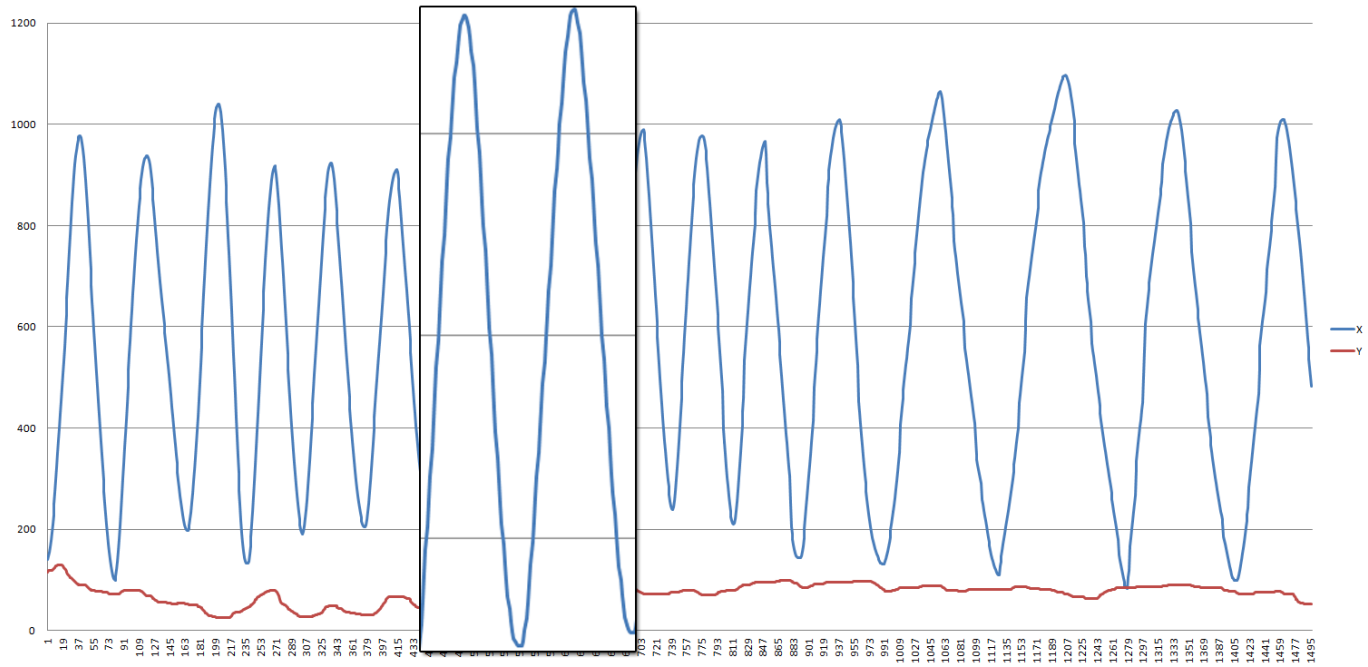
Zooming in on the moment of failure, the contrast in the prior smoothness and post stuttering of the input is very clear:



Additional observations

Input from iOS is smooth

The chart below shows the input logged from iOS during Unity Test 02 (Unity 4.5.4). It is smooth with no sign of the jagged waveform that we associate with the problem.



Triggering the problem

If a device appears not to exhibit the problem with Unity 4.5.4 we have found that rotating the device can sometimes provoke the problem. This is not conclusive and we have insufficient data to correlate the two events.

Other devices

We have tested on the following devices. The iPhone 6 (Plus) exhibited the problem on occasion but soon recovered sufficiently to consider the test passed.

Device	iOS	Unity 3.5.6	Unity 4.5.4
iPhone 4s	7.1.1	PASS	FAIL
iPhone 4s	8.0.2	PASS	FAIL
iPhone 5	7.1.1	PASS	FAIL
iPhone 5	8.0.2	PASS	FAIL
iPhone 6	8.0.2	PASS	PASS
iPhone 6 Plus	8.0.2	PASS	PASS
iPad 3	8.0.2	PASS	FAIL
iPad air	8.0.2	PASS	FAIL

Pass: The input remained smooth with no noticeable stuttering present in the translation of the plane. The text logs may occasionally show non-interleaved entries for iOS and Unity; however, this is not persistent, recovering within a few frames.

Fail: The input either starts erratic or becomes erratic after a short period of time, within a minute of starting the test. The translation of the plane stutters with the plane lurching between points. The logs will show non-interleaved entries for iOS and Unity over an extended period of time. It is rare for the problem to fully recover.

Accompanying files

The following files accompany this report:

- logs
  - UnityTest01.txt - log for the Unity Test 01 video.
  - UnityTest02.txt - log for the Unity Test 02 video.

- testProjects
  - UnityTest01_356 - Unity **3.5.6** project for Unity Test 01.
  - UnityTest02_454 - Unity **4.5.4** project for Unity Test 02.