**The process of anti-Shake**

In the most basic sense, the whole process of the system will involve the following.

1. Input
   1. The data/information that needs to be displayed
   2. The sensor data taken from the device to read its movement
   3. The data from the environment, like the location or activity the user is engaged in
   4. If possible, the ability of the device to locate where the user is looking
2. The Process will involve a method to calculate the intensity of the device vibration and by means of either a physical module/component or a software, counter-act the effect/distortion caused by the vibration thereby rendering a clear output to the user.
3. Output
   1. Clear image displayed
   2. Stabilized mobile device

**The System in which the App or device will interact with**

The System that will be effected by process is compromised of the 4 components stated above

1. The Human hands that carry the device
2. The Cover or external device that will in-between the hands and the device
3. The mobile device’s hardware components
   1. VGA/ image processing
   2. Display screen
   3. Sensors
4. The Mobile device software that is comprised of
   1. The O/S of the device (android/ Linux)
   2. The software that the process will affect

**Foreseen Problems**

1. If only the image is stabilized, then any input method that requires touch will become a hassle as the image is moving in relative to the device.
2. If the whole human body (including head & eyes) are moving/shaking, then the anti-shake could be counterproductive to the user
3. Processing the sensor data constantly to stabilize the display will consume the battery abundantly
   1. For this a threshold limiter should be applied, for when the stabilization should be active
4. Is extreme shakes worth stabilizing?
   1. For a camera, it is worth it, but can the same concept be useful in display?
5. Motion Sickness

**Questions to answer**

1. When would the anti-shake be useful? Which physical or mobile activity?
2. Would the rectangular shape of the display pose a problem?
3. Which implementation method is more feasible ad efficient? Physical, software, or hybrid?
4. Has the process been implemented outside a mobile system? Where has it succeeded where has it failed?
5. What techniques can counter-act the shaking?

# Input in an Anti-shake environment

Almost all modern mobile devices have a touch-screen as their major input method, from typing to interacting with the device/app (New York, 2011). This is due largely to the need to have more area for display while reducing the size of the phone needed to enable functionality. With the bloom in 3g and LTE the new Touchscreen phones have become a standard item everyone has (New York, 2011).

**Theory:**

Touch Screen input accuracy degrades with an anti-shake system.

**Premises**:

In a normal non-anti-shake mobile environment, the screen/display is fixed (not moving) relative to the human hands. But in an anti-shake system, the display/input screen will be moving relative to the hand. Which will cause problems if the only method of input is by touch.

**Test:**

To test this theory, an app will be designed to test the change in accuracy and timing needed for the input. It will be compromised of the following:

1. An anti-shake display (Setting the X, Y coordinates of the inner view according to the gyroscope)
2. An input numpad method (number buttons from 0-9) .
3. A task to complete with the users input and time took to complete the task saved.

The test will be comprised of entering a sequence of number displayed, in two methods/activities while the user is in motion(walking):

1. Fixed, to have a baseline to test against.
2. motion with flexible numpad.

The task is comprised of 15 number combinations that are comprised of 6-5 digits each, and the user has to enter as they walk. A textbox is used for the user to see their input and to know when to press “Enter” to submit their result. These results are saved simultaneously with the time taken to type each number. From this data, the results of both test can be used to calculate the rate of change in both the accuracy and time needed to input the number.

# Gyroscope sensor in android

“The gyroscope measures the rate of rotation in rad/s around a device's x, y, and z axis. The sensor's coordinate system is the same as the one used for the acceleration sensor. Rotation is positive in the counter-clockwise direction; that is, an observer looking from some positive location on the x, y or z axis at a device positioned on the origin would report positive rotation if the device appeared to be rotating counter clockwise. This is the standard mathematical definition of positive rotation and is not the same as the definition for roll that is used by the orientation sensor.” (Google, 2017)

The use of the Gyroscope to stabilize images is more of a mechanical aspect in camera’s like Sony and Fujifilm (Sony, 2010) (Fujifilms, 2011). The mechanical aspect is taken from tripod that are used in Steady-cams (PERRITANO, 2001).

In our project, we implemented the gyroscope to counteract the shaking instead of the accelerometer. The gyroscope was a better option because:

1. Not affected by the gravity
2. Does not need a high/low filter that takes up computation power
3. The sensor handles angular motion in rad/s
4. All three axes can be used in the calculation for anti-shake

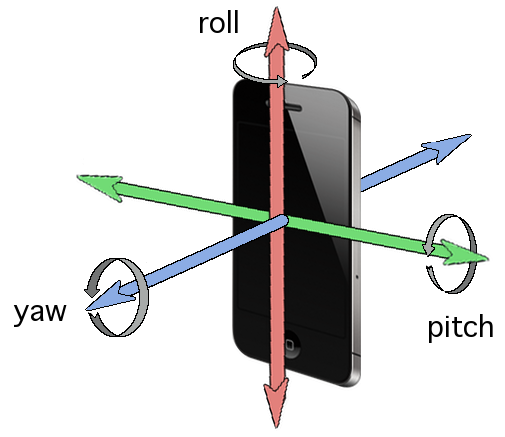


Figure https://www.google.de/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwjwyvf7sZLVAhWDtRQKHcy5BPIQjB0IBg&url=https%3A%2F%2Fstackoverflow.com%2Fquestions%2F24728022%2Fobtain-absolute-rotation-using-cmdevicemotion&psig=AFQjCNFwmvIhQfjof6

The anti-shake calculations are based on trigonometry of radians in a circle (Figure 2). For each access, it effects the shake in a specific way. The Roll effects the movement on the X-axis. The pitch effects the movement on the Y-axis of the image. And the Yaw effects both X and Y axis movement.

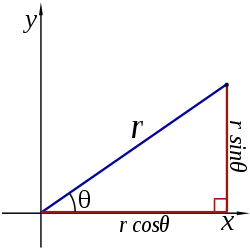


Figure https://math.stackexchange.com/questions/266832/x-and-y-coordinates-of-circle-giving-a-center-radius-and-angle

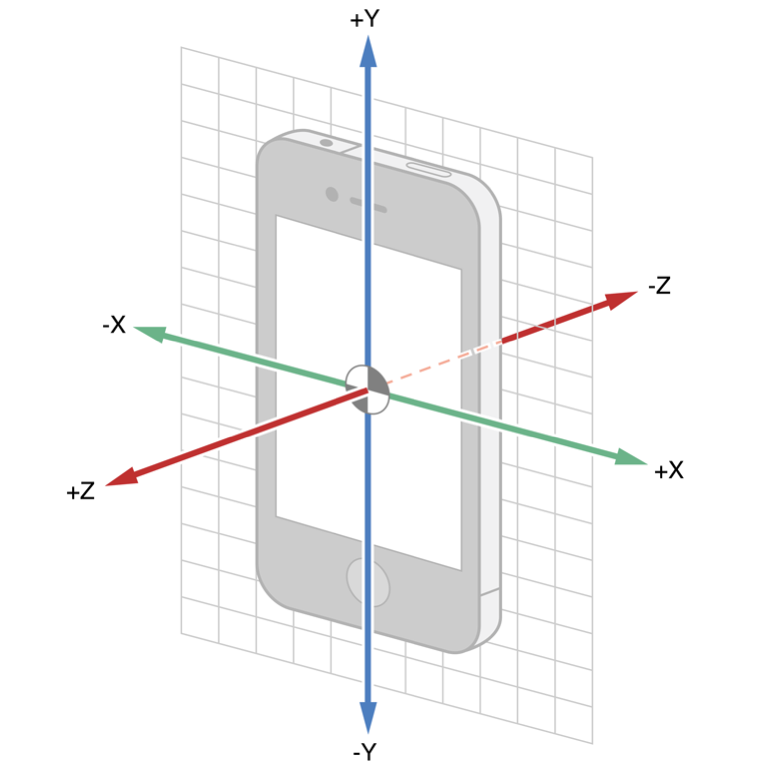


Figure https://www.google.de/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwj20MW239HVAhUCcRQKHQ39BrUQjB0IBg&url=http%3A%2F%2Fblog.denivip.ru%2Findex.php%2F2013%2F07%2Fthe-art-of-core-motion-in-ios%2F%3Flang%3Den&psig=AFQjCNFGdf78SgbteQiXE

# The App

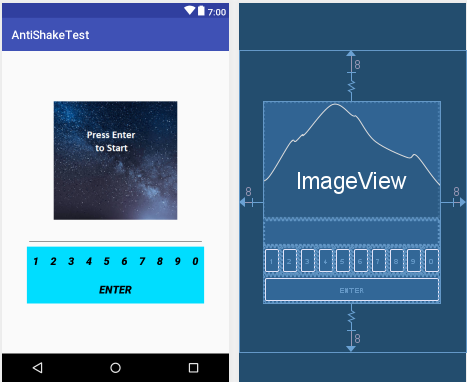


Figure The layout

As seen above, the layout is comprised of a constraintLayout that allows for control and movement of the components inside of it. “ConstraintLayout allows you to create large and complex layouts with a flat view hierarchy (no nested view groups). It is similar to RelativeLayout in that all views are laid out according to relationships between sibling views and the parent layout, but it is more flexible than RelativeLayout and easier to use with Android Studio's Layout Editor.” (Dev, 2017). The flexibility will allow for the second activity to move the components relative to the gyroscope motion sensor. To create a more linear movement to all the Items, they have been placed into a tableLayout. This way, the only component that needs to be manipulated is the position of the table and the other components will move with it accordingly.

The ImageView cycles through the pictures with numbers. The pictures are set up in the following order on both activities:

1. A welcome/start image to start the test.
   1. This allows for the user to start the test on his own.
   2. Any readings taken before the test starts are ignored.
   3. And allows for a break between the test results.
2. Numbered images
   1. 15 images with numbers on them
   2. Each number is between 5-6 digits
   3. They cycle through a predefined order, not randomly
3. Ending image
   1. This is to tell the user that the first part of the exam is over
   2. that they should prepare for the second one.
   3. Or that the experiment is over after the second activity.

The number pad is comprised of 11 buttons as seen in *Figure 4*. The numbers range from 0 till 9, with 0 being at the end as the normal keyboard standard on android. The Enter button is used to submit and change the picture.

The last part is the textbox. This is for used for both testing purposes and user information. The numbers the user types can be seen as they are typing, but it can’t be altered in any way.

## The code behind

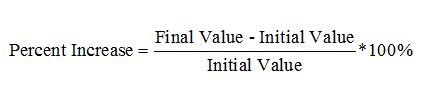
1. **imageArray** = **new int**[18];
   1. **this is to hold the images that are needed to be displayed and manipulate**
2. **Request storage permmissions**
   1. **To allow access to the internal storage to save the two text files with the input data and time required**
3. **buttonE(View v)** 
   1. **At X = 16 to switch to the next activity or end the application**
   2. **(ending/closing application doesn’t work, as google doesn’t allow it for some reason)**
   3. **Switching the image**
   4. **Writing to the txt file**
4. **Second Activity implements SensorEventListener onSensorChanged**
   1. px = (float) Math.sin(event.values[1]);
      1. directly convert to the x coordinate from roll
   2. py = (float) Math.sin(event.values[0]);
      1. Directly convert to the Y coordinate from pitch
   3. tx = (float) Math.sin(event.values[2]);
      1. Directly convert to the x-coordinate of yaw
   4. ty = (float) r -Math.cos(event.values[2]);
      1. directly convert to the x-coordinate of yaw
   5. new Thread
      1. to run the calculations on a different thread as a method to ease the work required
   6. tableLayout.setX tableLayout.setY to move the table

# Preliminary Results:

To results are collected as follows

1. a base test that collects the users normal rate of error while typing and the time required
2. a shake test that collects the users rate of error and time required in an anti-shake environment
3. The error of the digits is calculated as follows
   1. Any mismatching digit is one point
   2. Any extra digit is one point
   3. The percentage is the sum of the above points divided by the amount of digits
4. the error of the shake is tested against the base to find the rate of change
   1. positive is an increase and vice versa

the percentage increase will be calculated as follows:



In our case it will be =

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **User 1 Results** | | | | | | | | | |
| **Numbers** | **Base** | **Base Time(s)** | **Base Error** | **Base Error Rate** | **Test** | **Test Time(s)** | **Test Error** | **Test Error Rate** | **Time Difference** |
| **57031** | 57031 | 5121 | 0 | 0.00% | 57031 | 4323 | **0** | 0.00% | -15.58% |
| **99643** | 99643 | 3763 | 0 | 0.00% | 99643 | 4718 | **0** | 0.00% | 25.38% |
| **620457** | 620457 | 7016 | 0 | 0.00% | 620457 | 6071 | **0** | 0.00% | -13.47% |
| **29740** | 29740 | 3928 | 0 | 0.00% | 29749 | 4758 | **1** | 20.00% | 21.13% |
| **384109** | 384109 | 8113 | 0 | 0.00% | 384109 | 6567 | **0** | 0.00% | -19.06% |
| **47291** | 47291 | 6940 | 0 | 0.00% | 47291 | 4519 | **0** | 0.00% | -34.88% |
| **725882** | 725882 | 3808 | 0 | 0.00% | 625872 | 5746 | **2** | 33.33% | 50.89% |
| **728154** | 728154 | 4867 | 0 | 0.00% | 728054 | 5382 | **1** | 16.67% | 10.58% |
| **88421** | 88421 | 3278 | 0 | 0.00% | 88421 | 5002 | **0** | 0.00% | 52.59% |
| **02946** | 02946 | 7749 | 0 | 0.00% | 02945 | 5143 | **0** | 0.00% | -33.63% |
| **20095** | 30095 | 3620 | 1 | 20.00% | 20096 | 3731 | **0** | 0.00% | 3.07% |
| **846291** | 846291 | 6063 | 0 | 0.00% | 846291 | 5458 | **0** | 0.00% | -9.98% |
| **992277** | 992277 | 3311 | 0 | 0.00% | 992277 | 2985 | **0** | 0.00% | -9.85% |
| **74921** | 74921 | 3439 | 0 | 0.00% | 74921 | 3357 | **0** | 0.00% | -2.38% |
| **02946** | 02946 | 4700 | 0 | 0.00% | 02946 | 4890 | **0** | 0.00% | 4.04% |
| **Test 1 Sum Results** | | ***75716*** | ***1*** | **20.00%** |  | ***72650*** | ***4*** | **70.00%** | **28.85%** |
|  |  |  |  |  |  |  |  |  |  |
| Total increase in time for completion of activity in ms: | | | | -3066 |  |  |  |  |  |
| Percentage increase of Time Completion: | | | | -4.05% |  |  |  |  |  |
| Sum of percentage increase time differences: | | | | 28.85% |  |  |  |  |  |
| Avg Percentage Increase in Time Difference per number: | | | | 1.92% |  |  |  |  |  |
| Increase in Number of input Errors: | | | | 3 |  |  |  |  |  |
| Percentage increase in input Error: | | | | 300.00% |  |  |  |  |  |
| Percentage Increase of error typing rate: | | | | 250.00% |  |  |  |  |  |
| AVG Percentage Increase of typing error rate per number: | | | | 16.67% |  |  |  |  |  |

Table 1

Explanation of Table 1:

Definitions:

* *Activity*
  + The environment in which the user inputs the Numbers
  + In the App is split into two activities
* *Numbers:*
  + The provided numbers to the users that the inputs are validated against
* *Base:*
  + The user’s input of the Numbers provided in the first (non-shake) activity
* *Base Time:*
  + The time the user required to input one Numbers in the non-shake activity
* *Base Error:*
  + The number of digits that have been typed/input wrong (does include extra digits)
* *Base Error Rate* 
  + The percentage of Base Error per length of the Number
* *Test*:
  + The user’s input of the Numbers provided in the Second (Anti-Shake) Activity
* *Test Time:*
  + The time the user required to input one Number in the second Activity
* *Test Time Error*
  + The percentage of Test Error per length of the Number
* *Time Difference:*
  + The percentage of time Increase from Base Time to Test Time for each Number.
* Test 1 Sum Results:
  + The total Sum of each column, where applicable.
* *Total increase in time for completion of activity in ms:*
  + The Time Difference between the completion of the First Activity vs the Second Activity in milliseconds
* *Percentage increase of Time Completion:*
  + The Percentage increase in the Total time needed to complete each activity
* *Sum of percentage increase time differences*
  + The Sum of Each Time Difference column
* Avg Percentage Increase in Time Difference per number:
  + The Average amount of difference between Base Time and Test Time
* *Increase in Number of input Errors:*
  + The Difference between the sum of the Test Error minus the Base Error
* *Percentage increase in input Error:*
* Percentage Increase of error typing rate:
  + The Increase in the Error Rate from the Sum Base Error Rate to the Sum Test Error Rate compared to the length of the Number the error occurred in
  + 100% is not needed as both are already percentages
* AVG Percentage Increase of typing error rate per number:
  + 15 is the amount of Numbers per activity

|  |  |  |
| --- | --- | --- |
| End Results | | |
| Type | Total | Avg |
| *Total increase in time for completion of activity in ms:* | 43804 | 4867.111 |
| *Percentage increase of Time Completion:* | 62.15% | 6.91% |
| *Sum of percentage increase time difference* | 1213.68% | 134.85% |
| *Avg Percentage Increase in Time Difference per number:* | 80.91% | 8.99% |
| *Increase in Number of input Errors:* | 45 | 5 |
| *Percentage increase in input Error:* | 2300.00% | 255.56% |
| *Percentage Increase of error typing rate:* | 1124.30% | 124.92% |
| *AVG Percentage Increase of typing error rate per number:* | 74.95% | 8.33% |

Table 2 End Results collected from the Users

Figure Preliminary results

Figure 5 and Table 2 shows the end results that were collected from the testing. Figure 5 handles only the average data of table 2. The Data collected is found in the Appendix.

## OBSERVATION:

Although this data is only preliminary, there is an increase in error rate per input as a result of the anti-shake activity. There is also an increase in the time spent on each input, compared to the time taken to input the numbers when the anti-shake system is not in effect; however, this is negligible.

I

# References

Dev, G. (2017, July 01). *Build a Responsive UI with ConstraintLayout*. Retrieved from Developer Android: https://developer.android.com/training/constraint-layout/index.html

Fujifilms. (2011, January 05). *Fujifilm FinePix HS20EXR features*. Retrieved from Fujifilms: http://www.fujifilm.ca/products/digital\_cameras/hs/finepix\_hs20exr/features/page\_09.html

Google. (2017, June 10). *Motion Sensors*. Retrieved from Developer android: https://developer.android.com/guide/topics/sensors/sensors\_motion.html

New York, N. Y. (2011, Aug 26). *97% of All Smartphones Will Have Touchscreens by 2016*. Retrieved from ABI Research: https://www.abiresearch.com/press/97-of-all-smartphones-will-have-touchscreens-by-20/

PERRITANO, T. H. (2001, November 22). *How Steadicams Work*. Retrieved from How Stuff work: http://entertainment.howstuffworks.com/steadicam.htm

Sony. (2010, April 01). *Sony DSC-HX5V Features*. Retrieved from Sony : http://www.sony.co.uk/product/dsc-h-series/dsc-hx5v#/Overview