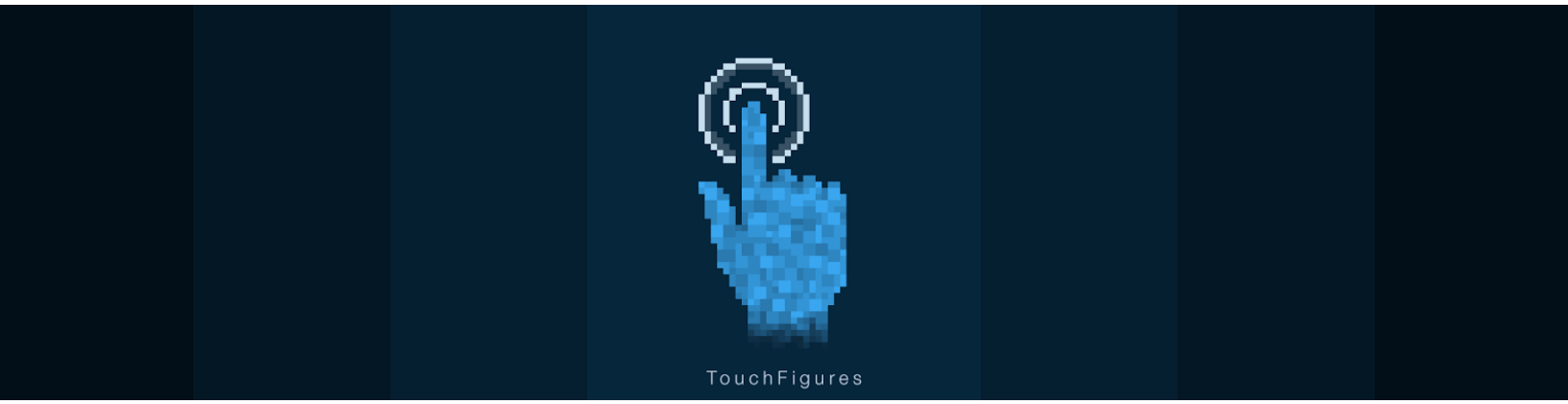
HAW Hamburg - Department DMI

Mobile Systeme SoSe 2014

iOS App - TouchFigures



Kai Leistner

Matrikel Nummer: 2079757

Inhalt

Die App

Name

Touch Figures

Zusammengefasst

Touch Figures misst die Behrührungen des Users auf dem Bildschirm des Smartphones und gibt dem Benutzer verschiedene Daten bezüglich der Touchscreen Interaktion auf dem Display aus. Es wird die Distanz der Berührungen, sowie die Zeit dieser Aktionen gemessen. Außerdem gibt es eine Heatmap, welche die häufigkeit der Berührungen in Bereichen anzeigt.

Platform

Als Betriebssystem wurde **iOS** gewählt.

Verwendete IDE

Also Entwicklungsumgebung wurde **Xcode** benutzt.

Testgerät

Zum testen der App wurde der Xcode interne **iPhone Simulator** und ein externer **iPod Touch 5G** verwendet.

Funktionen

Distanz

Was wird gemessen?

Es wird gemessen, welche Distanz der Finger des Benutzers auf der Oberfläche des Smartphone Displays zurücklegt. Hierbei soll es egal sein, ob der Benutzer mit seinem Finger gerade Linien, oder auch zickzack, kreise, kurven oder sonstige Formen auf die Oberfläche zeichnet.
Als einheit wird Centimeter (kurz cm) verwendet. Diese Information wird dem Benutzer als Text-Beschriftung in der App angezeigt. Wenn der Benutzer eine sehr große distanz zurücklegt, wird die angezeigte Einheit von Centimeter in Meter umgewandelt, damit die Zahl nicht all zu lang wird und das ablesen übersichtlich bleibt.
Angezeigt werden für die Distanz zwei Werte. Zum einen der Weg, den der Anwender mit seinem letzten durchgängigen Touch zurückgelegt hat und zum anderen die gesamte Distanz, die während der Nutzung der App bewältigt wurde.

Wie wird es berechnet?

Berechnung der Distanz mithilfe des Satz des Pythagoras:
Um die Distanz zwischen zwei Punkten zu berechnen kann folgende Formel verwendet werden, welche vom Satz des Pythagoras ($a^2 + b^2 = c^2$) abgeleitet ist:

$$distance := \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Im Programmcode sieht das dann folgendermaßen aus:

```
// Calculate the distance of the motion
    double touchDistance = sqrt(pow(touchPositionOld.x - touchPositionNew.x, 2.0) +
pow(touchPositionOld.y - touchPositionNew.y, 2.0));
```

Dieser Code wird wiederholt in der `touchesMoved` Methode aufgerufen, also wenn der Benutzer seinen finger auf dem Touchscreen bewegt. Durch das einsetzen der aktuellen (`movePositionNew`) und alten (`movePositionOld`) Position des Fingers, wird die Distanz während der Bewegung gemessen und zur gesamten Distanz der kompletten Touch-Geste addiert. Somit werden auch Bewegungen, die nicht eine Linie darstellen, möglichst akkurat gemessen. Eventuell können Ungenauigkeiten auftreten, sollte der Nutzer seinen Finger extrem schnell über den Bildschirm bewegen, da in diesem Fall die Methode nicht häufig genug vom System aufgerufen wird und somit die punkte nicht ganz akkurat der realen Touch-Linie folgen. Dies ist jedoch eher schlecht zu umgehen und stellt zudem einen Ausnahmefall dar.

Für den Letzten Touch werden die Punkte für einen kontinuierlichen Touch zusammengezählt. Bei erneutem berühren des Bidlschirms wird der Wert wieder auf 0 gesetzt und für die folgende Touchgeste neu berechnet. Für den Totalen Wert werden einfach alle Punkte (ohne zurücksetzen des wertes auf 0 beim erneuten Berühren) zusammengezählt.

Da das System intern in Points rechnet, muss dieser wert in die gewollte Maßeinheit umgerechnet werden. Für Centimeter wird hierbei einfach der Wert durch die vorhandene Pixeldichte pro Centimeter (ppc) gerechnet (Hierbei entspricht ein Point auch einem Pixel). Dieser Wert entspricht dann der physikalischen Distanz auf dem Display.

Zeit

Was wird gemessen?

Genauso wie die Distanz wird auch die Zeit gemessen, die der Benutzer mit seinem Finger auf dem Display des Smartphones verbringt. Der Zähler beginnt sobald das Display berührt wird, auch wenn keine Bewegung des Fingers stattfindet, und endet sobald sich der Finger des Anwenders wieder vom Touchscreen erhebt.
Die gemessene Zeit wird in Sekunden auf dem Bildschirm angezeigt. Sollte die Zeitmessung größere Ausmaße erreichen wird von Sekunden in Minuten umgerechnet.
Ebenfalls wie bei der Distanz werden auch hier zwei Werte angezeigt. Die Zeit, die der Benutzer für den letzten Touch gebraucht hat und die gesamte Zeit der Touchscreen-Berührung, die im laufe der Anwendung zusammen gekommen ist.

Wie wird es berechnet?

Mittels eines Timers wird die Zeit berechnet. Dieser timer wird in `touchesBegan` zugewiesen und gestartet.

```
// Start timer to measure touch time
    self.timer = [NSTimer scheduledTimerWithTimeInterval:0.01 target:self
selector:@selector(counter) userInfo:nil repeats:YES];
```

Jede 10 Millisekunden wird die `counter` Methode aufgerufen. Dies geschieht so lange, bis der Anwender seinen finger wieder vom Touchscreen löst und `touchesEnded` (oder `touchesCancelled`) aufgerufen und der timer beendet wird.

```
[self.timer invalidate];    // Cancel the timer
```

In `counter` wird die Zeit hochgezählt und in `timeTouched` (bzw. `totalTimeTouched`) gespeichert. Dann wird der text aktualisiert und falls die Zeit 60 Sekunden, also 1 Minute, überschritten haben sollte, wird die ausgabe in Minuten statt Sekunden umgerechnet.

Heatmap

Was wird gemessen?

Mittels der Heatmap wird gemessen, wo und wie häufig sich der Anwender mit dem Finger auf dem Touchscreen befand. Dargestellt wird dieses mittels vieler kleiner Blöcke, welche heller / farbintensiver dort sind, wo sich der Finger häufig befand. In dunklen bereichen hat sich der Anwender nicht so häufig mit seinem Finger aufgehalten.
Zudem werden je nach wechselnder Ansicht Werte innerhalb der Blöcke angezeigt, die den Intensivitäts-Wert noch einmal numerisch verdeutlichen.

Wie wird es berechnet?

Berührt der Anwender den Touchscreen oder bewegt den Finger über diesen, wird die `updateBlockValue` Methode aufgerufen.
Die Blöcke besten aus UIViews (genauer genommen aus TFXBoxView, welches von UILabel abgeleitet ist). Der Wert dieser Subviews wird nun an entsprechender Stelle um 1 erhöht. Dann wird überprüft, ob sich der Höchstwert geändert hat und gegebenenfalls wird dieser aktualisiert. Zuletzt wird die `updateBlockViews` Methode aufgerufen.

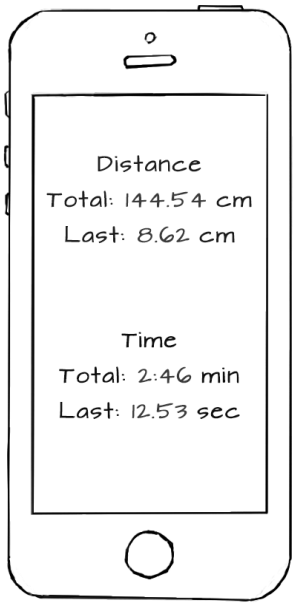
In `updateBlockViews` wird durch alle Block-Subviews iteriert und der Alpha wert des Blockes im verhältnis zum Höchstwert gesetzt.

```
// Set tranparency in relation of highest touch value
            double alphaValue = (float)blockView.value / (float)highestBlockValue;
            [blockView setBackgroundColor:[UIColor colorWithRed:(55.0/255.0)
green:(166.0/255.0) blue:(239.0/255.0) alpha:alphaValue]];
```

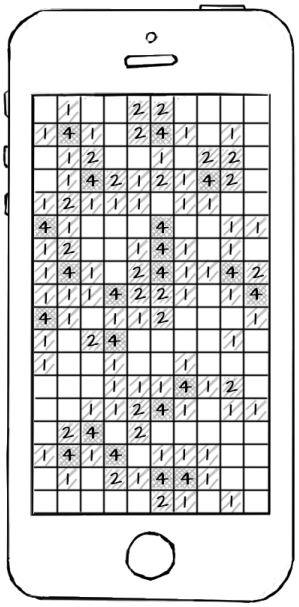
Danach wird, falls nötig, der Wert des Blocks als Textlabel auf diesen gesetzt.

Aufbau

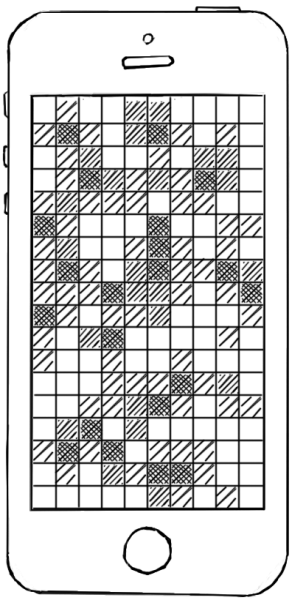
Screens



Distanz / Zeit

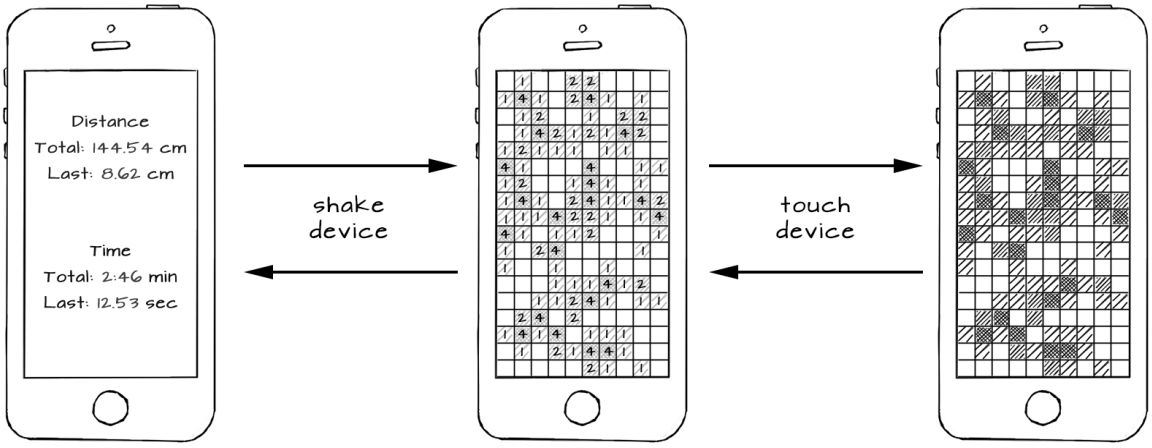


Heatmap mit Werten



Nur Heatmap

Ckickfkow



Aussehen

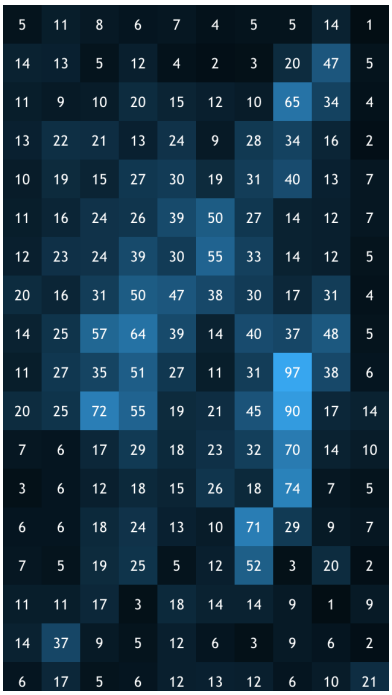
Flat Design

Das Aussehen der App ist im Flat Design gehalten. Diese Art des Design basiert auf schlichten Formen und wie der Name es sagt Flachen Elementen. Seit der Einführung von iOS 7 ist das der Design Standard und sollte mehr oder weniger bevolgt werden, damit die App sich gut in das restliche System einpasst. Das Flat Design steht im starken kontrast zum Skeumorphistischem Design ansatz des alten iOS 6 Systems, welches eine möglichst realistische Darstellung von Objekten und Bedienelementen bevorzugt.

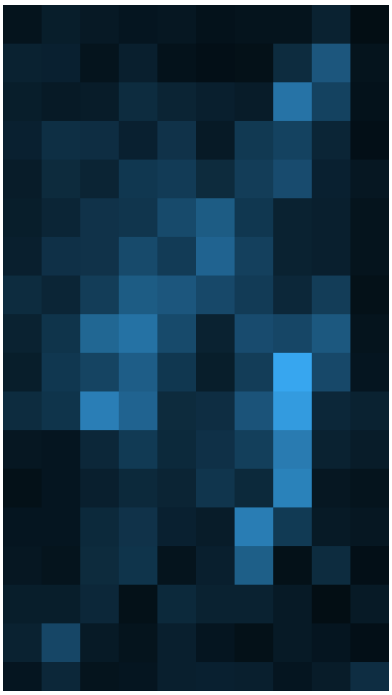
Farbpalette

Screenshots

Distanz / Zeit



Heatmap mit Werten



Nur Heatmap

Sonstiges

Schwierigkeiten

Zuerst wurde für die Heatmap ein anderer Ansatz gewählt. Die Heatmap sollte mittels eines Brush Images auf eine Gradient Map Pixel für Pixel gezeichnet werden. Und diese Gradient Map sollte dann ausgelesen und daraus die zu zeigende Heatmap ersellt werden. Das ganze sollte im prinzip wie auf folgender website beschrieben ist erfolgen:
http://philippseifried.com/blog/2011/09/30/generating-heatmaps-from-code/
Das Problem hierbei ist jedoch, dass es sehr umständlich ist, unter iOS auf die Pixelebene eines Bildes zuzugreifen. Zudem sind einige Funktionen die man anwenden muss um das zu ermöglichen nicht sehr sicher oder stabil.
Nach einem ersten Versuch der implementierung mit einigen Hindernissen, und nachdem die Implementierung als größere Blöcke, die stilistisch auch zum Flat design passen, in den Sinn kam, wurde der Code verworfen und letzte Idee wurde in angriff genommen.

Nicht implementierte Funktionen

- Multitouch wurde nicht implementiert, da dazu die Zeit gefehlt hat, was der funktionsweise der App aber an sich überhaupt nicht schadet

- Man hätte sicherlich noch weitere informationen über den Touch des Anwenders heraufinden und darstellen können, aber für diese App wurde sich auf die offensichtlichen Messdaten Distanz und Zeit konzentriert. Dazu stellt die Heatmap ein sehr schickes Feature dar.
-

Anhang

Quelltext

TFXViewController.h

```
//
//  TFXViewController.h
//  TouchFiguresX
//
//  Created by Kai Leistner on 02.06.14.
//  Copyright (c) 2014 Kai Leistner. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface TFXViewController : UIViewController

@property (nonatomic, strong) IBOutlet UILabel *touchLabelText;
@property (nonatomic, strong) IBOutlet UILabel *touchDistanceText;
@property (nonatomic, strong) IBOutlet UILabel *totalTouchDistanceText;

@property (nonatomic, strong) IBOutlet UILabel *timeLabelText;
@property (nonatomic, strong) IBOutlet UILabel *timeTouchedText;
@property (nonatomic, strong) IBOutlet UILabel *totalTimeTouchedText;
@property (nonatomic, strong) IBOutlet UILabel *infoLabelText;
@property (nonatomic, strong) NSTimer *timer;

@end
```

TFXViewController.m

```
//
//  TFXViewController.m
//  TouchFiguresX
//
//  Created by Kai Leistner on 02.06.14.
//  Copyright (c) 2014 Kai Leistner. All rights reserved.
//

#import "TFXViewController.h"
#import "TFXBoxView.h"

@interface TFXViewController ()
{
    // Touch
    CGPoint touchPositionStart;
    CGPoint touchPositionEnd;
    double totalTouchDistance;

    CGPoint movePositionOld;
    CGPoint movePositionNew;
```

```
    double totalMoveDistance;
    double totalMoveCentimeter;
    double lastMoveDistance;
    double lastMoveCentimeter;

    // Time
    double timeTouched;
    double totalTimeTouched;

    // Heatblocks
    int columnCount;
    int rowCount;
    int blockSize;

    CGFloat screenWidth;
    CGFloat screenHeight;

    CGPoint blockPos;
    int blockField[18][10];
    TFXBoxView *blockViews[18][10];
    int blockAtTouchPosX;
    int blockAtTouchPosY;
    int highestBlockValue;

    bool showValues;
    bool shakedDeviceOnce;

    UIView *layerView;
}

@end


@implementation TFXViewController

- (void)viewDidLoad
{
    [super viewDidLoad];

    self.view.backgroundColor = [UIColor colorWithRed:(2.0/255.0) green:(12.0/255.0)
blue:(18.0/255) alpha:1.0];

    highestBlockValue = 0;      // highest block touch value for reference
    showValues = true;          // should the values in the blockviews be shown
    shakedDeviceOnce = false;   // was the davice shaked once? (for info label)

    blockSize = 32;             // the size of the blocks
    columnCount = (int)[[UIScreen mainScreen] applicationFrame].size.width/blockSize;
    rowCount = (int)ceil([[UIScreen mainScreen] applicationFrame].size.height/blockSize);

    // Setup the block views
    int xPosition = 0;
    int yPosition = 0;
    for (int y = 0; y < rowCount; y++) {
        xPosition = 0;
        for (int x = 0; x < columnCount; x++) {
            TFXBoxView *view = [[TFXBoxView alloc] initWithFrame:CGRectMake(xPosition,
yPosition, blockSize, blockSize)];

            [view setFont:[UIFont fontWithName: @"Trebuchet MS" size: 12.0f]];
            [view setTextColor:[UIColor whiteColor]];
```

```
            [view setTextAlignment:NSTextAlignmentCenter];

            [self.view addSubview:view];

            // save referneces to the view blocks in an array
            blockViews[y][x] = view;
            blockViews[y][x].value = 0;

            xPosition += blockSize;
        }
        yPosition += blockSize;
    }

    // create layer that hides the boxmap
    layerView = [[UIView alloc] initWithFrame:CGRectMake(0, 0, [[UIScreen mainScreen]
applicationFrame].size.width, [[UIScreen mainScreen] applicationFrame].size.height)];
    [layerView setBackgroundColor:[UIColor blackColor]];
    [self.view addSubview:layerView];

    // bring the text views in front of all other views
    [self.view bringSubviewToFront:self.touchLabelText];
    [self.view bringSubviewToFront:self.touchDistanceText];
    [self.view bringSubviewToFront:self.totalTouchDistanceText];
    [self.view bringSubviewToFront:self.timeLabelText];
    [self.view bringSubviewToFront:self.timeTouchedText];
    [self.view bringSubviewToFront:self.totalTimeTouchedText];

    [self.view bringSubviewToFront:self.infoLabelText];
    self.infoLabelText.alpha = 0.0;
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

- (NSUInteger) supportedInterfaceOrientations {
    return UIInterfaceOrientationMaskPortrait;
}

- (UIInterfaceOrientation) preferredInterfaceOrientationForPresentation {
    return UIInterfaceOrientationPortrait;
}


#pragma mark - Touches

- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event
{
    lastMoveDistance = 0;   // Reset move distance on new touch
    timeTouched = 0;        // Reset time of touch on new touch

    for (UITouch *touch in touches) {
            // Send to the dispatch method, which will make sure the appropriate
subview is acted upon
            touchPositionStart = [touch locationInView:self.view];
        movePositionNew = touchPositionStart;
       }

    // Start timer to measure touch time
```

```
    self.timer = [NSTimer scheduledTimerWithTimeInterval:0.01 target:self
selector:@selector(counter) userInfo:nil repeats:YES];

    [self updateBlockValue];    // Update the blockvalue at current position

}

- (void)touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event
{
    movePositionOld = movePositionNew;
    for (UITouch *touch in touches) {
                // Send to the dispatch method, which will make sure the appropriate
subview is acted upon
                movePositionNew = [touch locationInView:self.view];
        }
    // Calculate the distance of the motion
    double moveDistance = sqrt(pow(movePositionOld.x - movePositionNew.x, 2.0) +
pow(movePositionOld.y - movePositionNew.y, 2.0));
    totalMoveDistance = totalMoveDistance + moveDistance;
    totalMoveCentimeter = totalMoveDistance / 64;   // Convert distance into actual
centimeter

    lastMoveDistance = lastMoveDistance + moveDistance;
    lastMoveCentimeter = lastMoveDistance / 64;     // Convert distance into actual
centimeter

    // Update label views
    NSString *distanceFormatString = NSLocalizedString(@"Total: %.2f cm", @"Format string
for info text for distance");
    self.touchDistanceText.text = [NSString stringWithFormat:distanceFormatString,
totalMoveCentimeter];

    NSString *lastDistanceFormatString = NSLocalizedString(@"Last: %.2f cm", @"Format
string for info text for total distance");
    self.totalTouchDistanceText.text = [NSString
stringWithFormat:lastDistanceFormatString, lastMoveCentimeter];

    [self updateBlockValue];
}

- (void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event
{
    for (UITouch *touch in touches) {
                // Send to the dispatch method, which will make sure the appropriate
subview is acted upon
                touchPositionEnd = [touch locationInView:self.view];
        }

    // Calculate Distance
    double touchDistance = sqrt(pow(touchPositionEnd.x - touchPositionStart.x, 2.0) +
pow(touchPositionEnd.y - touchPositionStart.y, 2.0));
    totalTouchDistance = totalTouchDistance + touchDistance;
    [self.timer invalidate];    // Cancel the timer
    [self animateViewBlocks];   // Switch digits on and off the block views
}

- (void)touchesCancelled:(NSSet *)touches withEvent:(UIEvent *)event
{
    [self.timer invalidate];    // Cancel the timer
}
```

```
#pragma mark - Timer

- (void)counter {
    // Count the touch time up
    timeTouched = timeTouched + 0.01;
    totalTimeTouched = totalTimeTouched + 0.01;

    if ( !shakedDeviceOnce && totalTimeTouched >= 5.0) {
        [UIView animateWithDuration:1.0 animations:^{
            self.infoLabelText.alpha = 1.0;     // Display the "shake device" info label
        }];
    }

    // Convert seconds into minutes if needed and update label views
    int minutes, seconds;
    if (totalTimeTouched >= 60.0) {
        minutes = (int) totalTimeTouched/60;
        seconds = (int) totalTimeTouched%60;

        self.totalTimeTouchedText.text = [NSString stringWithFormat:@"Total: %02d:%02d
min", minutes, seconds];
    }
    else {
        self.totalTimeTouchedText.text = [NSString stringWithFormat:@"Total: %05.2f sec",
totalTimeTouched];
    }
    if (timeTouched >= 60.0) {
        minutes = (int) totalTimeTouched/60;
        seconds = (int) totalTimeTouched%60;

        self.timeTouchedText.text = [NSString stringWithFormat:@"Last: %02d:%02d min",
minutes, seconds];
    }
    else {
        self.timeTouchedText.text = [NSString stringWithFormat:@"Last: %05.2f sec",
timeTouched];
    }
}

#pragma mark - Box Heatmap

- (void)updateBlockViews
{
    // Iterate through every viewblock
    for (int y = 0; y < rowCount; y++) {
        for (int x = 0; x < columnCount; x++) {

            TFXBoxView *blockView = blockViews[y][x];

            // Set tranparency in relation of highest touch value
            double alphaValue = (float)blockView.value / (float)highestBlockValue;
            [blockView setBackgroundColor:[UIColor colorWithRed:(55.0/255.0)
green:(166.0/255.0) blue:(239.0/255.0) alpha:alphaValue]];

            if (showValues) {
                int valueDigit = (int) (blockViews[y][x].value);

                if (valueDigit == 0) {
                    [blockView setText: [NSString stringWithFormat: @""]];
// Hide values
                } else {
```

```
                    [blockView setText: [NSString stringWithFormat: @"%d", valueDigit]];
// Show values
                }
            }
        }
    }
}

- (void)updateBlockValue
{
    blockAtTouchPosX = (int) movePositionNew.x/blockSize;
    blockAtTouchPosY = (int) movePositionNew.y/blockSize;
    TFXBoxView *blockView = blockViews[blockAtTouchPosY][blockAtTouchPosX];
    blockView.value += 1;
    if(blockView.value > highestBlockValue) {
        highestBlockValue = blockView.value;    // Update highest touch value
    }
    [self updateBlockViews];
}

#pragma mark - Shake Gesture

- (BOOL)canBecomeFirstResponder {
    return YES;
}

- (void)motionEnded:(UIEventSubtype)motion withEvent:(UIEvent *)event
{
    // Device was shaked
    shakedDeviceOnce = true;
    self.infoLabelText.hidden = true;

    if (motion == UIEventSubtypeMotionShake)
    {
        [self updateBlockViews];

        // Animations

        // Slide the text outside the display and hide it
        [UIView animateWithDuration:0.5 delay:0.0
options:UIViewAnimationOptionCurveEaseInOut animations:^{
            short dist = 240;
            self.touchLabelText.alpha = (self.touchLabelText.alpha == 1.0 ? 0.0 : 1.0);
            self.touchLabelText.center = (self.touchLabelText.alpha == 1.0 ?
CGPointMake(self.touchLabelText.center.x, self.touchLabelText.center.y + dist) :
CGPointMake(self.touchLabelText.center.x, self.touchLabelText.center.y - dist));
            self.touchDistanceText.alpha = (self.touchDistanceText.alpha == 1.0 ? 0.0 :
1.0);
            self.touchDistanceText.center = (self.touchDistanceText.alpha == 1.0 ?
CGPointMake(self.touchDistanceText.center.x, self.touchDistanceText.center.y + dist) :
CGPointMake(self.touchDistanceText.center.x, self.touchDistanceText.center.y - dist));
            self.totalTouchDistanceText.alpha = (self.totalTouchDistanceText.alpha == 1.0
? 0.0 : 1.0);
            self.totalTouchDistanceText.center = (self.totalTouchDistanceText.alpha ==
1.0 ? CGPointMake(self.totalTouchDistanceText.center.x,
self.totalTouchDistanceText.center.y + dist) :
CGPointMake(self.totalTouchDistanceText.center.x, self.totalTouchDistanceText.center.y -
dist));

            self.timeLabelText.alpha = (self.timeLabelText.alpha == 1.0 ? 0.0 : 1.0);
```

```
            self.timeLabelText.center = (self.timeLabelText.alpha == 1.0 ?
CGPointMake(self.timeLabelText.center.x, self.timeLabelText.center.y - dist) :
CGPointMake(self.timeLabelText.center.x, self.timeLabelText.center.y + dist));
            self.timeTouchedText.alpha = (self.timeTouchedText.alpha == 1.0 ? 0.0 : 1.0);
            self.timeTouchedText.center = (self.timeTouchedText.alpha == 1.0 ?
CGPointMake(self.timeTouchedText.center.x, self.timeTouchedText.center.y - dist) :
CGPointMake(self.timeTouchedText.center.x, self.timeTouchedText.center.y + dist));
            self.totalTimeTouchedText.alpha = (self.totalTimeTouchedText.alpha == 1.0 ?
0.0 : 1.0);
            self.totalTimeTouchedText.center = (self.totalTimeTouchedText.alpha == 1.0 ?
CGPointMake(self.totalTimeTouchedText.center.x, self.totalTimeTouchedText.center.y -
dist) : CGPointMake(self.totalTimeTouchedText.center.x,
self.totalTimeTouchedText.center.y + dist));

        }completion:nil];


        // Layer hiding the boxmap
        [UIView animateWithDuration:0.5 animations:^{
            layerView.alpha = (layerView.alpha == 1.0 ? 0.0 : 1.0);
        }];
    }
}

- (void)animateViewBlocks
{
    // Animate and show or hide the values on top of the block heatmap
    for (int y = 0; y < rowCount; y++) {
        for (int x = 0; x < columnCount; x++) {

            TFXBoxView *blockView = blockViews[y][x];

            [UIView animateWithDuration:0.25 animations:^{
                blockView.transform = CGAffineTransformMakeScale(0, 0);
            } completion:^(BOOL finished){
                [UIView animateWithDuration:0.25 animations:^{
                    blockView.transform = CGAffineTransformMakeScale(1, 1);
                }];

                if (showValues) {
                    int valueDigit = (int) (blockViews[y][x].value);
                    if (valueDigit == 0) {
                        [blockView setText: [NSString stringWithFormat: @""]];
                    } else {
                        [blockView setText: [NSString stringWithFormat: @"%d",
valueDigit]];
                    }
                } else {
                    [blockView setText: [NSString stringWithFormat: @""]];
                }
            }];
        }
    }
    showValues = !showValues;
}

@end
```

TFXBoxView.h

```
//
//  TFXBoxView.h
//  TouchFiguresX
//
//  Created by Kai Leistner on 12.06.14.
//  Copyright (c) 2014 Kai Leistner. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface TFXBoxView : UILabel

@property int value;

@end
```

TFBoxView.m

```
//
//  TFXBoxView.m
//  TouchFiguresX
//
//  Created by Kai Leistner on 21.06.14.
//  Copyright (c) 2014 Kai Leistner. All rights reserved.
//

#import "TFXBoxView.h"

@interface TFXBoxView ()

@end

@implementation TFXBoxView

@end
```

Übersichtsplakat

TouchFigures

Strecke / Zeit

Oben wird die **Distanz** angezeigt, die auf dem Touchscreen zurück gelegt wurde.

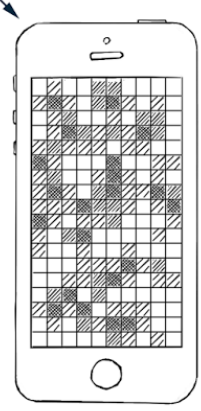
Unten wird die **Zeit** angezeit, die der Touchscreen berührt wurde.

((Schüttel das Gerät um zwischen den Ansichten zu wechseln.))

Heatmap

Die Heatmap zeigt an, wo sich der Finger auf dem Touchscreen am häufigsten befand.

Nach jedem Touch wechselt die Anzeige:
Werte auf den Blöcken anzeigen
oder **nur die Blöcke** anzeigen



TouchFigures // Kai Leistner