Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung Projekt 1
SoSe 2012

Projekt 2. Semester Master Informatik

Philipp Kühn

Ein interaktiver Couchtisch

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Philipp Kühn

Ein interaktiver Couchtisch

Ausarbeitung Projekt 1 eingereicht

im Studiengang Informatik Master
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Abgegeben am 25.09.2012

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Parallel zur Veranstaltung Anwendungen 2 wurde der darin und bereits in [Küh11] vorgestellte Couchtisch in der Veranstaltung Projekt 2 entwickelt. Diese Entwicklung soll es ermöglichen, in praktischen Tests herauszufinden, in wie fern sich der vorgestellte Couchtisch als Interaktive Konsole für einen Haushalt eignet. Ziel war es hierbei ein Framework für Apps auf dem Couchtisch zu schaffen, sodass diese einfach integriert werden können. Diese Apps können nun in unterschiedlichen GUIs dargestellt werden, welche gegeneinander getestet werden sollen. Dies ist das Hauptziel des entwickelten Tisches: Es sollen die beiden GUIs gegeneinander getestet werden, um eine Aussage darüber treffen zu können, ob in einer Single User Umgebung eine frei drehbare, oder eine fest ausgerichtete GUI besser ist. In vielen Dokumenten über Tabletops wird nur von Multiuserszenarien gesprochen und somit eine frei drehbare GUI empfohlen. So auch in [Mic11] S. 9 / Abschnitt 3.1.1.a. Neben der Entwicklung eines Frameworks wurden in dieser Veranstaltung auch einige Apps geschrieben, damit an Ihnen möglichst reale Tests durchgeführt werden können.

1.2 Das Living Place Hamburg

Im Living Place Hamburg [Liv11] soll die Wohnung der Zukunft aufgebaut und erforscht werden. Dieses Projekt an der Hochschule für Angewandte Wissenschaften Hamburg läuft seit 2009. Die Wohnung bietet diverse Sensoren, mit welchen der Bewohner und sein physisches Befinden erfasst werden kann. Auf Basis von einer mehrschichtigen Interpretation sollen dann aus diesen Rohdaten Kenntnisse über Absichten gewonnen werden. Aus diesen Erkenntnissen soll dann die Wohnung intelligent auf seinen Bewohner und dessen Umstände reagieren. Als Beispiel sei hier der Wecker 2.0 [Ell10] genannt. An

Tabletops bietet das Livingplace bisher eine multitouchfähige Küchentheke, welche von Lorenz Barnkow [Bar10] erforscht wird.

1.3 Verwendete Hard-, Software & Frameworks

Als Tisch dient ein Samsung SUR40 [Sam12] mit Microsoft Pixelsense Technologie [Mic111]. Der Tisch ist ein 40“ LCD, welcher auf Beine montiert wird und somit als Tisch dient. Als Betriebssystem komm Microsoft Windows 7 zum Einsatz. Das Besondere an dem Display gegenüber herkömmlichen Multitouch Displays ist, dass in jedem Pixel ein Infrarotsensor verbaut wurde, sodass der Tisch „sehen“ kann. Eine ähnliche Technologie wurde bereits beim Microsoft Surface 1 (Nicht zu verwechseln mit dem Microsoft Surface Tablet) angewandt. Hier wurde ein Beamer genutzt, welcher von unten ein Bild auf eine Plexiglasscheibe projiziert und zur Eingabeerkennung wurden 4 Infrarotkameras im Korpus des Tisches montiert. Entgegen der Werbung haben Praxistests gezeigt, dass die verwendete Technologie im SUR40 nicht scannen kann. Sehr groß geschriebene Buchstaben können erkannt werden, ein normal geschriebener Text (Calibri, Schriftgröße 11) kann jedoch nicht erkannt werden. Aufgrund der schlechteren Erkennung kann der SUR40 auch keine Identity Tags (Eine Art 2D Barcode mit 64Bit Informationen) mehr erkennen, sondern unterstützt nur noch Byte Tags (8Bit Informationen). Nichtsdestotrotz können Finger erkannt und sogar deren Richtung bestimmt werden. Für weitergehende Erkennung ist es möglich, das Bild, welches der SUR40 „sieht“, abzugreifen und eigene Algorithmen zu schreiben. Dies wurde in diesem Projekt allerdings nicht genutzt.
Als Programmiersprache wurde C# und als Entwicklungsumgebung Visual Studio 2010 genutzt. Für C# gibt es bereits ein Surface SDK, welches alle Funktionen des SUR40 bereitstellt. Bis auf die Touch-optimierten Controls wurde jedoch kein weiterer Gebrauch davon gemacht. Zur Toucherkennung wurde also das normale Windows 7 Touch genutzt, welches bereits sehr gut mit C# und der verwendeten Oberfläche WPF arbeitet. Zum Laden von Apps wurde das Managed Extensibility Framework (MEF) genutzt, welches bereits bei .Net 4.0 mitgeliefert wird. MEF bietet die Möglichkeit, Verzeichnisse automatisch nach Klassen zu durchsuchen, welche vorher festgelegte Verträge erfüllen, diese zu instanziieren und eine Referenz auf diese Instanz in einer Variablen zu speichern. Verlangen mehrere Komponenten eine Instanz der gleichen Klasse, so wird diese nur einmal instanziiert und beide erhalten den gleichen Verweis. Ein Vertrag besteht dabei aus einem Typ und/oder einem Namen und wird per Annotation festgelegt.

Abbildung 1.1 Samsung SUR40 mit Microsoft PixelSense [Sam12]

2 Das Framework

Das Framework besteht zum einen aus dem ApplicationProvider, welcher die einzelnen Apps lädt, zum anderen aus den GUIs, ALignedView und Scatterwiew, welche die von dem ApplicationProvider geladenen Apps anzeigen.

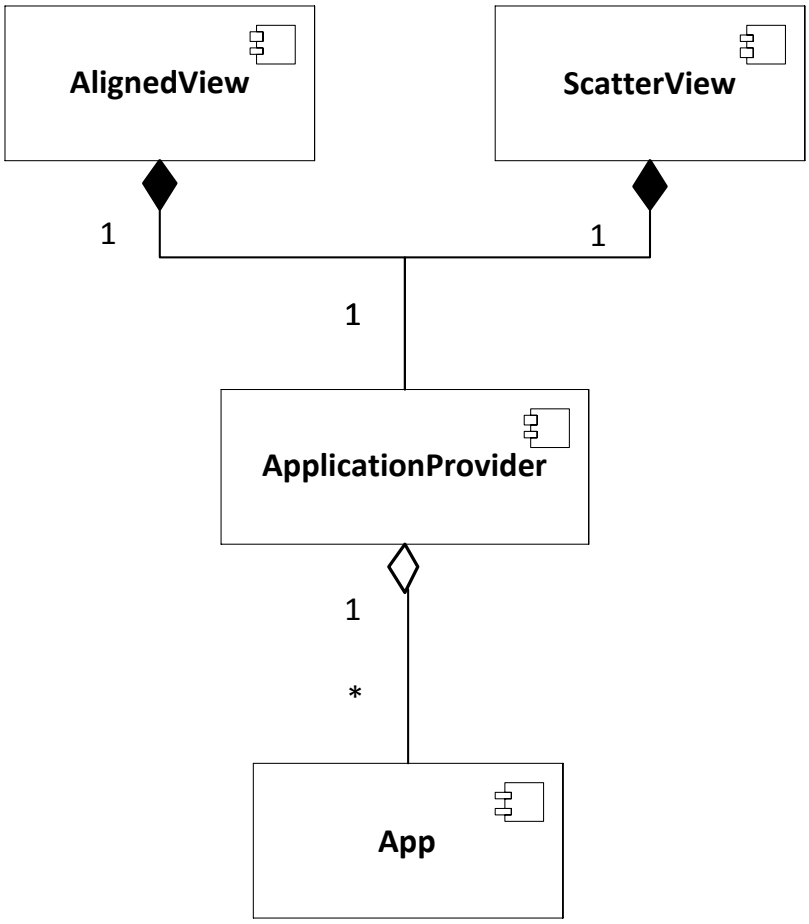

Abbildung 2.1 Frameworkstruktur

2.1 Appstruktur

Damit Apps von dem Framework erkannt werden, müssen diese Das Interface IApplication erfüllen. Jede Application besteht widerum aus mehreren IApplicationPart Teilen. Jeder IApplicationPart besteht wiederum aus einem Info- und einem Full-View. Im Full-View wird das normale Programm angezeigt, wohingegen im Info-View Programmname und kompakte Informationen angezeigt werden. Mithilfe von MEF werden alle .dlls, welche in den Apps Ordner kopiert wurden durchsucht und alle Klassen, welche dieses Interface implementieren geladen. Diese Liste von Apps stellt nun der ApplicationProvider zur Verfügung, sodass sie von den unterschiedlichen GUIs geladen und angezeigt werden kann. Zusätzlich können Apps Handler definieren. Diese dienen zur Inter-App Kommunikation. Die Browser App definiert z.B. einen Handler für Links, welcher von der Twitter App genutzt wird, um in Tweets enthaltene Links aufzurufen. Auch Handler werden Mithilfe von MEF erstellt und eingebunden, gehen allerdings nicht den Umweg über den ApplicationProvider, sondern werden direkt von anderen Apps eingebunden.

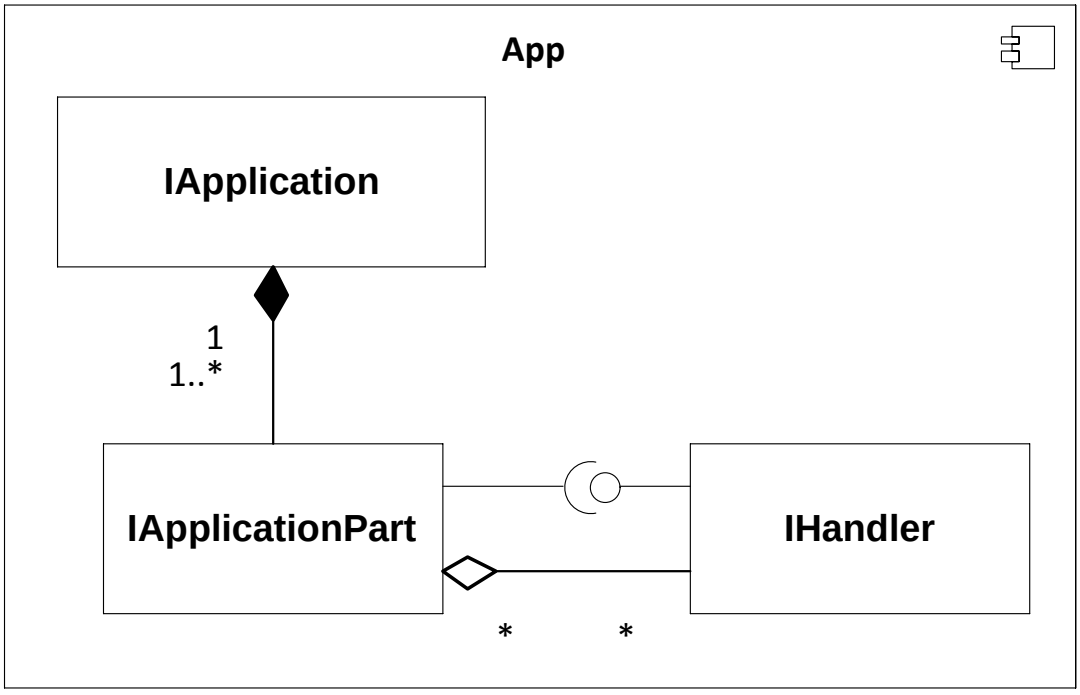

Abbildung 2.2 App Komponente

2.2 GUIs

Es wurden 2 GUIs entwickelt, wobei diese jeweils noch weitere Einstellmöglichkeiten bieten.

2.2.1 Scatter View

Die Scatter View zeigt alle Apps als kleine Quadrate an, welche über den Bildschirm verstreut sind. Initial wird hierbei die Info-View der Apps angezeigt. Werden diese Qudrate mit Hilfe einer Zoomgeste vergrößert, wird ab einem bestimmten Schwellwert die Full-View des Programms angezeigt. Beim Verkleinern wird wieder zur ursprünglichen Info-View zurückgekehrt. Die Qudrate können hierbei frei gedreht und skaliert werden. Die genutzte SurfaceScatterView bietet darüber hinaus eine einfache Physik, sodass die Apps auch über den Bildschirm „geschubst“ werden können. Diese GUI wurde entwickelt, da sie aufgrund der Physik natürlich wirkt und im Surface Entwicklerhandbuchvorgeschlagen wird, dass Anwendungen möglichst frei Dreh- und Skalierbar sind.
Als Möglichkeit, den zur Verfügung stehenden Platz besser zu nutzen, wurde außerdem ein Raster implementiert, an welchem die Apps ausgerichtet werden können. Hierbei wurden verschiedene Möglichkeiten der automatischen Ausrichtung getestet (Raster anzeigen / nicht anzeigen; Größe, Rotation und Position unabhängig / gleichzeitig rastern). Es stellte sich hierbei heraus, dass es für den User am verständlichsten ist, wenn das Raster sichtbar ist und nur, wenn aufgrund Größe, Rotation und Position zugleich erkennbar ist, dass der User das Objekt ausrichten möchte, eine automatische Rasterung vorgenommen wird.

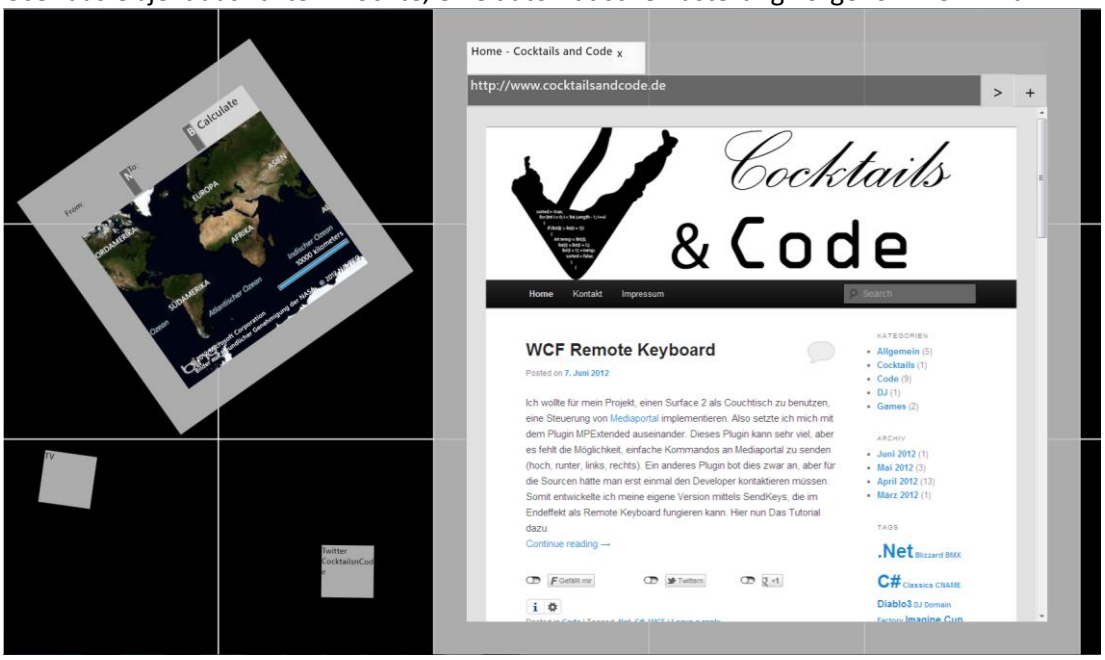

Abbildung 2.3 ScatterView

2.2.2 Aligned View

Die Aligned View GUI bietet auf der linken Seite eine Liste aller Programme, wobei das aktuell gewählte Programm im Hauptbereich angezeigt wird. Die Liste zeigt dabei die Info-Views der Programme an, sodass der User einen ständigen Überblick behält. Im Hauptbereich wird dann die Full-View angezeigt. Diese GUI wurde entwickelt, um ein Maximum aus dem zur Verfügung stehenden Platz herauszuholen, da im Gegensatz zur Scatter View alles immer mit dem Monitor ausgerichtet ist und so keine unnutzbare Bereiche entstehen.

Durch Integration mit dem im Living Place Hamburg genutzten Ubisense [Ubi12], welches die Möglichkeit bietet die Position von bestimmten Sendern zu bestimmen wurde eine Funktion entwickelt, welche das Interface immer zum Benutzer ausrichtet und je nach Entfernung auch vergrößert. Hierfür gibt es auf dem genutzten Apache AcitveMQ [Apa12] das Topic „UbisenseTracking“ [Liv12], von welchem Positionsdaten des Users gelesen werden können. Die Drehung erfolgt hierbei in 90° Schritten, sodass eine Ausrichtung zum Bildschirm gewahrt bleibt. Die Skalierung erfolgt kontinuierlich, wobei der Nullpunkt die obere linke Ecke ist, sodass bei großer Entfernung vom User zum Tisch nur noch die Programmliste, nicht aber der Hauptbereich sichtbar ist. Auf diese Weise kann der User auch von weiter Entfernung noch Einige Informationen ablesen.

Darüber hinaus wurde die Möglichkeit von mehreren Hauptbereichen getestet, welche durch Aufziehen von der rechten Seite des Tisches entstehen sollten. Programme würden hierbei per Drag & Drop in den jeweiligen Hauptbereich gezogen. Aufgrund eines Fehlers im SDK, welcher verursachte, dass das Event, welches anzeigt, dass der User mit dem Aufziehen eines neuen Hauptbereiches fertig ist, bereits ausgelöst wurde, als die Manipulation noch im Gange war, wurde diese Möglichkeit jedoch verworfen.

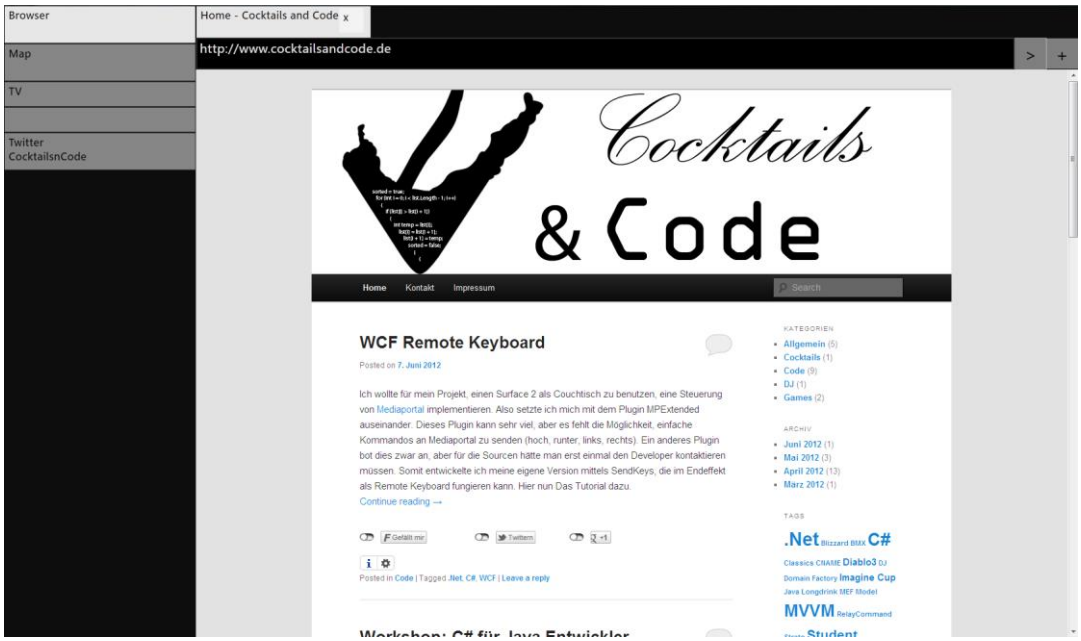

Abbildung 2.4 Aligned View

3 Apps

Es wurden 4 unterschiedliche Apps entwickelt, welche hier beschrieben werden sollen.

3.1 Browser

Ein Browser, welcher das sog. Tabbed Browsing unterstützt wurde entwickelt. Da im Internet surfen eine der Hauptanwendungen ist, welche gerne nebenbei auf der Couch ausgeführt werden, hatte diese App die höchste Priorität. WPF bietet von Haus aus ein WebBrowser Control an, welches auf dem Internet Explorer basiert. Die Touch Unterstützung von diesem Control ist sehr gut, so können alle vom Smartphone oder Tablet

bekannten Gesten zum Scrollen und Zoomen von Webseiten genutzt werden. Intern greift dieses Control dabei auf ein Window Handle des Internet Explorers zurück. Da normale Fenster unter Windows nicht gedreht werden können, kann auch das WebBrowser Control, welches von WPF zur Verfügung gestellt wird, nicht gedreht werden. Somit schied dies aus, da für beide entwickelte GUIs eine Drehbarkeit von Nöten ist. Stattdessen wurde auf Awesomium [Khr12], ein auf Chromium basierendes WPF Control zurückgegriffen. Dieses Control kann, im Gegensatz zum WebBrowser Control von WPF, frei gedreht werden. Der Nachteil dieses Controls ist allerdings, dass es keine spezielle Touch Unterstützung bietet. So werden keine Gesten zum Scrollen oder Zoomen erkannt und es muss hierbei mit dem Windows typischen Scrollbalken gearbeitet werden. Dieser Umstand muss bei Tests berücksichtigt werden, da sich dies mit ziemlicher Sicherheit auf die User Experience auswirken wird.

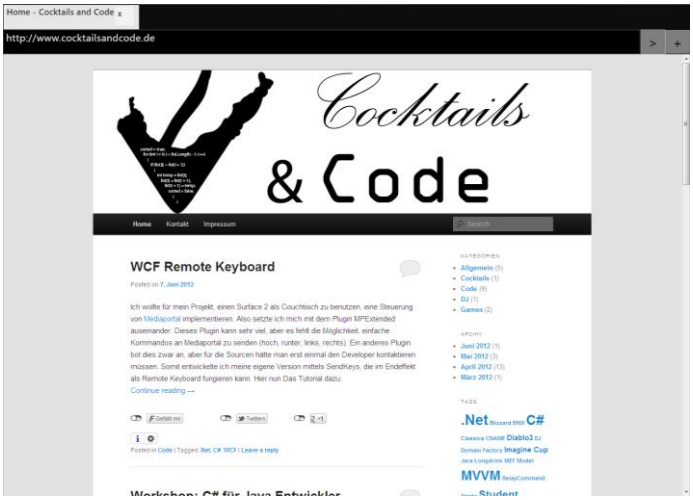

Abbildung 3.1 Browser App

3.2 Twitter

Die Twitter App kann die Tweets einer einzelnen Person darstellen. Für die Verbindung zu Twitter wurde hierbei auf die Twitterizer Bibliothek [Twi12] gesetzt. Hierdurch kann sich das Programm per OAuth gegenüber der Twitter API identifizieren. Im Info View wird hierbei neben dem Appnamen der Name der Person angezeigt, von welcher Tweets dargestellt werden. Im Full View werden alle Tweets untereinander als Liste dargestellt. Enthällt ein Tweet einen Link, kann dieser angeklickt werden, woraufhin sich ein neuer Tab in der Browser App öffnet und die Webseite anzeigt. Es muss dabei darauf geachtet werden, dass tatsächlich der Link gedrückt wird. Eine Berührung des restlichen Tweets hat keine Auswirkung. Bei einem angeklickten Link, öffnet sich dieser im Hintergrund, sodass der Lesefluss nicht gestört wird.

3.3 Map

Für die Implementation einer Karte wurde auf Bing Maps [Mic12] zurückgegriffen, da es hier bereits ein vorgefertigtes WPF Control gibt. Dieses Ruft über einen Webservice Daten von Bing Maps ab, sodass eine einfache Implementation gegeben ist. Zusätzlich wurde eine Routenfunktion implementiert, welche eine Route zwischen 2 gegebenen Orten auf der Karte anzeigt. Um dies zu realisieren, muss ein weiterer Webservice von Bing Maps konsumiert werden und die Positionen der Kreuzungen, welche auf der Route liegen müssen von Hand in der Karte durch Linien verbunden werden. Das Größte Problem stellte hierbei das Erkennen von eingegebenen Orten dar. Der Webservice von Bing Maps, welcher hierfür zuständig ist, arbeitet nicht besonders gut. Gibt man einen Ort ein und lässt diesen mehrfach durch den Webservice abbilden, so kann es vorkommen, dass hierbei unterschiedliche Orte oder aber auch ein leeres Ergebnis (null) zurückgegeben werden. Um diesen Fall abzufangen, muss der zurückgegebene Ort noch einmal lokal überprüft werden.

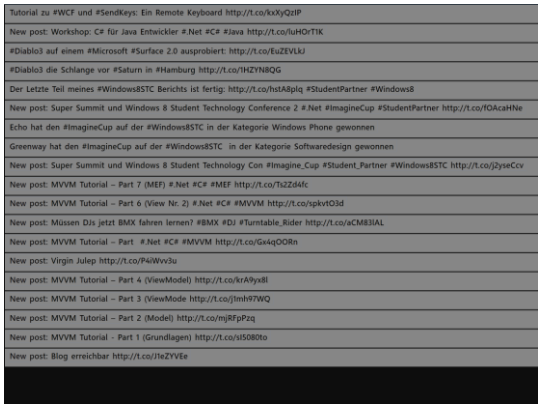

Abbildung 3.2 Twitter App

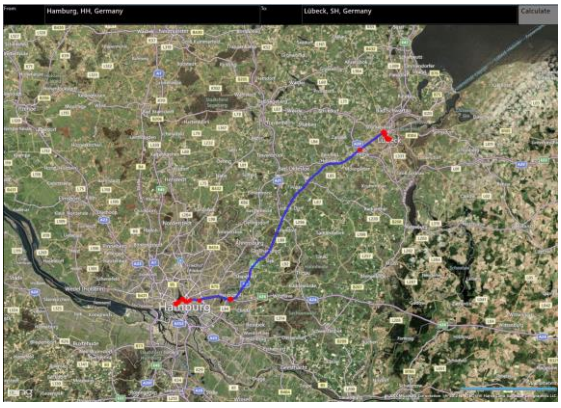

Abbildung 3.3 Map App

3.4 Fernseher Fernsteuerung

Da eine typische Beschäftigung auf einer Couch das Fernsehgucken ist, ist eine Fernsteuerung für diesen obligatorisch. Als Fernsehprogramm wurde hierbei MediaPortal [Med12]gewählt. Es ist Aufgrund seiner Vielseitigkeit und guten Erweiterbarkeit bestens dazu geeignet, in diesem Szenario den Fernseher darzustellen. Es ist außerdem in der gleichen Programmiersprache (C#) geschrieben und Open Source. Des Weiteren gibt es bereits Plug-Ins, welche eine Fernsteuerung ermöglichen (MPExtended [MPE12] und Wifi Remote [Cor12]). Auch die Client-Server Struktur von MediaPortal bietet sich für eine Integration in das Livinplace an, da so ein Server mit mehreren TV-Karten mehrere Bildschirme bedienen kann.

Die App bietet nun 2 Funktionen, welche über Tabs an der Oberkannte erreicht werden können. Zum einen eine normale Fernsteuerung, welche Befehle wie hoch, runter, links, rechts, ok und abbrechen unterstützt. Zum anderen eine EPG Ansicht des Fernsehprogramms des aktuellen Tages. Dieses kann per Touch frei verschoben werden, sodass sowohl andere Programme, als auch andere Zeiten, als die aktuell angezeigten sichtbar werden. Die Sender sind hierbei in der linken Spalte und die Zeiten in der obersten Zeile ständig sichtbar. Durch einen Klick auf einen Sender schaltet der Fernseher auf diesen um.

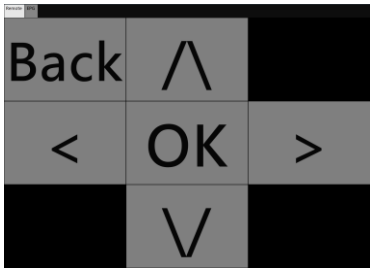

Abbildung 3.4 Fernseher App - Remote

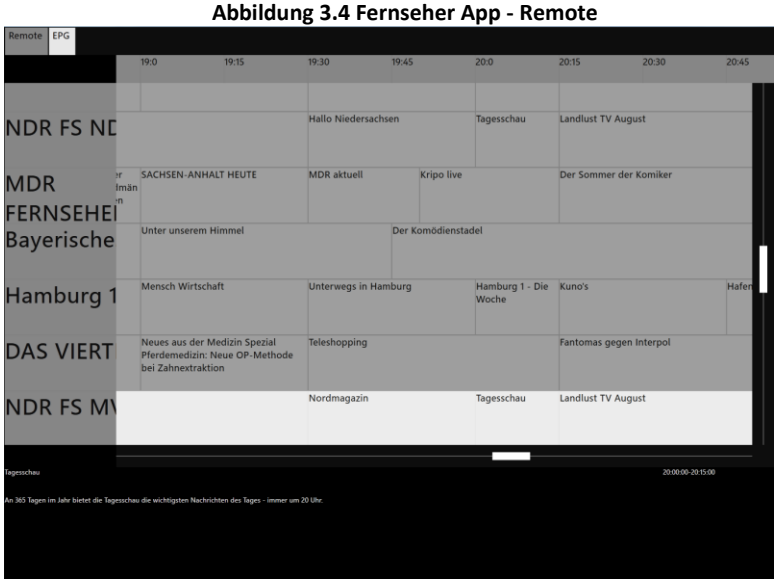

Abbildung 3.5 Fernseher App - EPG

4 Zusammenfassung und Ausblick

In diesem Kapitel sollen die Erkenntnisse aus dem Vorangegangenen Abschnitt zusammengefasst und ein Kurzer Ausblick auf die weitere Entwicklung des interaktiven Couchtisches gegeben werden.

4.1 Zusammenfassung

Mit dem Framework, den Apps und den GUIs wurde eine Grundlage für den interaktiven Couchtisch geschaffen. Durch die einfache Appstruktur kann der Tisch des Weiteren jeder Zeit einfach um neue Funktionen erweitert werden. Es konnten nicht alle geplanten Features umgesetzt werden (vgl. 2.2.1). Allerdings bieten die beiden entwickelten GUIs noch genug Möglichkeiten, sodass eine gute Grundlage für kommende Tests gegeben ist.

4.2 Ausblick

In der kommenden Veranstaltung „Projekt 2“ sollen Tests auf dem entwickelten Tisch durchgeführt werden. Hierbei sollen sowohl erfahrene, als auch unerfahrene Benutzer sich in Tests mit dem Tisch beschäftigen. Da davon auszugehen ist, dass mit der Zeit ein besseres Verständnis für die verwendeten GUIs und Bedienkonzepte eintritt, sollen über einen längeren Zeitraum verteilt immer wieder Tests mit denselben Personen durchgeführt werden, sodass die Lernkurve beobachtet werden kann.

Literaturverzeichnis

[MPE12] *MPExtended.* [Online] [Zitat vom: 01. 08 2012.] http://mpextended.github.com/.
[Apa12] **Apache Software Foundation. 2012.** Apache ActiveMQ ™. [Online] 28. 01 2012. [Zitat vom: 20. 02 2012.] http://activemq.apache.org/.
Ein interaktiver Cuchtisch. [Küh11] **Kühn, Philipp. 2011.** Hamburg, Germany : s.n., 2011. Anwendungen 1.
Ein Wecker in einem ubicom Haus. [Ell10] **Ellenberg, Jens. 2010.** Hamburg, Germany : s.n., 2010. Anwendung 2.
Eine Multitouch-fähige Küchentheke: Im Kontext des Living Place Hamburg. [Bar10] **Barnkow, Lorenz. 2010.** Hamburg, Germany : s.n., 2010. Anwendungen 1.
[Khr12] **Khrona LLC.** *Awesomium | Web-Browser Framework.* [Online] [Zitat vom: 01. 08 2012.] http://awesomium.com/.
[Liv12] **Livingplace.** ActiveMQ Messages – Living Place Wiki. [Online] [Zitat vom: 30. 04 2012.] http://livingplace.informatik.haw-hamburg.de/wiki/index.php/ActiveMQ_Messages#Topic:_UbisenseTracking.
[Liv11] **Livingplace Hamburg. 2011.** Livingplace Hamburg. [Online] 2011. [Zitat vom: 07. 11 2011.] http://www.livingplace.org.
[Med12] **MediaPortal.** *MEDIAPORTAL - a HTPC Media Center for free!* [Online] [Zitat vom: 01. 08 2012.] http://www.team-mediaportal.com/.
[Mic12] **Microsoft.** *Bing Maps WPF Control.* [Online] [Zitat vom: 01. 08 2012.] http://msdn.microsoft.com/en-us/library/hh750210.
[Mic11] **Microsoft. 2011.** *Microsoft® Surface® 2 Design and Interaction Guide.* [Online] 11. 7 2011. [Zitat vom: 01. 08 2012.] http://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=26713.
[Mic111] **Microsoft. 2011.** *Welcome to Microsoft PixelSense.* [Online] 2011. [Zitat vom: 01. 08 2012.] http://www.pixelsense.com.
[Sam12] **Samsung.** SUR40 mit Microsoft® PixelSense™ - ÜBERSICHT | SAMSUNG. [Online] [Zitat vom: 05. 08 2012.] http://www.samsung.com/de/consumer/notebooks-displays/large-format-displays/surface-2/LH40SFWTGC/EN.

[Cor12] **Schiffer, Cornelius.** *wifiremote - A MediaPortal plugin to send and receive remote commands via TCP/IP - Google Project Hosting.* [Online] [Zitat vom: 01. 08 2012.] http://code.google.com/p/wifiremote/.
[Twi12] **Twitterizer.** *Twitterizer | We want to give your app Twitter.* [Online] [Zitat vom: 01. 08 2012.] http://www.twitterizer.net/.
[Ubi12] **Ubisense.** Echtzeit-Ortungssystems (RTLS) und Geospatial-Beratung - Ubisense. [Online] [Zitat vom: 05. 08 2012.] http://de.ubisense.net/en/.

Abbildungsverzeichnis