Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung Anwendungen 1
WiSe 2010

Marc Piechura

Eine Multitouch-fähige Küchentheke im Kontext des Living Place Hamburg

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Inhaltsverzeichnis

Abstract

In dieser Ausarbeitung werden die Schnittstellen und das Framework vorgestellt, die für die Entwicklung auf der Multitouch-fähigen Küchentheke benötigt werden. Weiterhin wird eine Zusammenfassung der Probleme und Lösungsansätze entwickelt, die im Zusammenhang mit der enormen Größe der Küchentheke auftreten oder das angestrebte Arbeiten mit mehreren Personen an einem Gerät betreffen.

1 Einführung in das Themengebiet

Wie wohnen wir in der Zukunft?

Diese Fragestellung soll an der Hochschule Angewandte Wissenschaften Hamburg durch das Projekt des Living Place [HAW-Hamburg (2009)] erforscht werden. Seit 2009 wird an dieser experimentellen Wohnung gearbeitet, sie ist als „Smart Home“ konzipiert und dient dazu verschiedene Bereiche der Ambient Intelligence zu erforschen. Die Wohnung wurde in einem Gebäude [Abbildung 5] der HAW-Hamburg eingerichtet und ist in einen Wohn- und Entwicklungsbereich [Abbildung 6] aufgeteilt.

Die Wohnung soll, neben verschiedenster Sensoren und Kameras, im Übergangsbereich zwischen Küche und Esszimmer eine Multitouch-fähige Küchentheke enthalten und dient den Bewohnern damit als zentrales Informationsterminal. Ein mögliches Anwendungsszenario wäre z.B. der Außendienstplaner von Barnkow[1]. Durch die Größe [Abbildung 7] der Küchentheke ergeben sich einige Probleme, die, neben den möglichen Lösungsansätzen, vorgestellt werde. Daraufhin wird auf die Schnittstelle eingegangen, die die Hardware mit dem Betriebssystem verbindet, um anschließend das Betriebssystem und das für die Entwicklung zuständige Framework vorzustellen. Abschließend wird der Arbeitsbereich für das Projekt abgegrenzt und die Ausarbeitung zusammengefasst.

2 Problemstellungen bei der Arbeit mit einer Multitouch-fähigen Küchentheke

In diesem Bereich werden die Problemstellungen und Lösungsansätze zusammengefasst, die in der Arbeit von Barnkow[2] für die Multitouch-fähige Küchentheke bereits analysiert und vorgestellt worden sind.

2.1 Territorialität und Kollaboration an Tischen

Da die Multitouch-fähige Küchentheke für die Gruppenarbeit genutzt werden soll, müssen auch die Probleme berücksichtigt werden, die bei einer solchen Arbeit auftreten können. Zum einen, der Anspruch einer Person in der Gruppe seinen eigenen Arbeitsbereich festzulegen, in dem er sich frei bewegen kann und nicht durch dritte gestört wird, zum anderen der Aspekt der Zusammenarbeit in einer Gruppe.

[1] [Barnkow, Lorenz (2009)]
[2] [Barnkow, Lorenz (2010)]

2.1.1 Territorialität

Durch die Auswertung von vorangegangenen Arbeiten und Beobachtungsstudien konnte ermittelt werden, dass beim Arbeiten in der Gruppe eine unbewusste Unterteilung in drei Territorien stattfindet. Um die Problematik der verschiedenen Territorien bei einer Multitouch-fähigen Küchentheke zu lösen, werden dem Benutzer folgende Bereiche zur Verfügung gestellt.

- **Personal Space** Dieses Territorium wird durch die Sitzposition festgelegt, die eine Person am Tisch einnimmt. Es dient der Durchführung von persönlichen Arbeiten und dessen Bereich liegt direkt an der Tischkante. Durch das Verschieben von Objekten in das persönliche Territorium werden Objekte für den jeweiligen Nutzer reserviert und sind somit für andere Gruppenmitglieder unzugänglich.
- **Group Space** Der Bereich der nicht durch den *Personal Space* reserviert ist, steht allen Gruppenmitgliedern zur Verfügung, um Daten auszutauschen und zusammenzuarbeiten.
- **Storage Space** In diesen Bereichen werden Objekte abgelegt, die nicht direkt für die Arbeit notwendig sind. Je nach Lage des Territoriums sind die Daten für die Gruppe zugänglich (*Group Space*) oder nur für eine bestimmte Person (*Personal Space*).

Dadurch ist gewährleistet, dass sich die Gruppenmitglieder untereinander austauschen können, ohne auf den Schutz der Privatsphäre zu verzichten.

2.1.2 Kollaboration

Für den Erfolg des Projektes ist es unabdingbar, dass die Zusammenarbeit in der Gruppe reibungslos und effizient durchgeführt werden kann. Eine Studie, in der 28 Probanden einige Experimente durchführen mussten, zeigt jedoch, dass das Zusammenarbeiten an aktuelle Tabletops nicht derart reibungslos und effizient vonstatten geht, wie man es an normalen Tischen gewöhnt ist. Durch diese Experimente ergaben sich fünf Problemstellungen, die einer Zusammenarbeit in der Gruppe im Wege stehen können.

- **Teilnehmer benötigen entfernte Objekte** Durch die Größe der Küchentheke reicht die eigene physikalische Reichweite nicht aus, um entfernte Objekte zu greifen. Um dennoch an die Objekte zu gelangen, müsste man aufstehen oder andere Gruppenmitglieder um Hilfe bitten.
- **Entnahme von Objekten aus fremden Personal Spaces** Wenn Nutzer Objekte aus einem fremden *Personal Space* benötigten, werden diese oft ohne Einverständnis des „Besitzers“ genommen.
- **Personal Space ändert sich mit der Zeit** Der Bereich wächst mit der Ansammlung von Objekten, dies kann dazu führen, dass diese Bereiche zu groß werden. Andererseits gibt es Nutzer die keine „eigenen“ Objekte besitzen und damit faktisch auch keinen *Personal Space*.
- **Koordination in kleinen Group Spaces schwierig** Bei kleinen *Group Spaces* kommt es teilweise zu konkurrierenden Zugriffen auf Objekte.
- **Teilnehmer orientieren bewegliche Objekte neu** Schon bei einer Diskussion von zwei sich gegenüberstehenden Personen muss das Objekt häufig gedreht werden.

Aus diesen Problemen wurden fünf Anforderungen an moderne Tabletops abgeleitet.

- **Bewegliche Arbeitsbereiche** Arbeitsbereiche müssen sich manipulieren lassen, um bei Bedarf neue Bereiche zu erzeugen, sie zu teilen oder wieder zusammenzuführen.

- **Zugriff auf entfernte Objekte** Dem Nutzer müssen Hilfsmittel angeboten werden, um auf entfernte Objekte zuzugreifen. Dies lässt sich z.B. durch bestimmte Gesten realisieren.
- **Besitzansprüche flexibel regeln** Die Ansprüche auf Objekte müssen kontexabhängig gehandhabt werden. Bei einem Puzzle ist der Besitz beispielsweise nicht so wichtig wie in einem Pokerspiel.
- **Objekte und Aktionen transparent halten** Wenn man die Bedeutung der Aktionen anderer sehen und interpretieren kann, dann lässt sich daraus ableiten, ob ein Objekt aktuell benötigt wird und ob man es deshalb aus einem fremden *Personal Space* entfernen darf.
- **Mobile Menüs und Werkzeuge** Individuelle Menüs und Wegzeuge, die vom System zur Verfügung gestellt werden, müssen beweglich sein, um sie neu zu platzieren und dem Benutzer folgen zu können.

2.2 Anzeigeorientierung

Die Orientierung von Objekten ist auf Tabletops von besonderer Bedeutung, da die Interaktion oft von verschiedenen Seiten erfolgt. Texte die auf dem Kopf stehen, lassen sich nur schwer lesen und auch Programmelemente können so nur schwer erkannt werden. Als Lösungsansatz für dieses Problem dient eine Kombination aus manueller und automatischer Ausrichtung. Bei der automatischen Ausrichtung wird ein Vektorfeld der Arbeitsfläche erstellt, welches die Ausrichtung der Objekte vorgibt. Die Vektoren werden in Richtung der nächste Kante initialisiert und außerdem geglättet, um abrupte Richtungsänderungen zu verhindern. Die manuelle Ausrichtung erfolgt mittels zweier Gesten, als erstes wird der zu manipulierende Bereich ausgewählt, um anschließend die neue Ausrichtung festzulegen.

2.3 Physische Reichweite der Benutzer

Bei den Anforderungen an moderne Tabletops in Unterunterabschnitt 2.1.2 wurde unter anderem die Erreichbarkeit entfernter Objekte besproben. Da sich der Nutzer gegebenenfalls über den Tisch beugen oder andere Gruppenmitglieder um Hilfe bitten muss und damit die Arbeitsabläufe stört, muss für diesen Fall eine Lösung gefunden werden, bei der der Nutzer die anderen Gruppenmitglieder nicht an der Arbeit hindert. Die Manipulation des gesamten Arbeitsbereiches durch Verschieben, Rotieren oder Skalieren entfällt wegen oben genannter Probleme. Ein anderer Lösungsansatz ist der sogenannte *I-Grabber*, hierbei handelt es sich um einen virtuellen Greifarm, der mit Hilfe einer Geste initialisiert wird, um dann entfernte Objekte, ohne Behinderung der Gruppenmitglieder, zu manipulieren.

3 Schnittstellen und Framework

In diesem Abschnitt werden die Software und Protokolle vorgestellt die zur Realisierung des Projektes notwendig sind. Zum einen handelt es sich dabei um das TUIO-Protokoll [Unterabschnitt 3.1], als Schnittstelle zwischen den Multitoucheingaben und dem Betriebssystem, dass Betriebsystem [Unterabschnitt 3.2] als solches sowie das Framework [Unterabschnitt 3.3] in der die Benutzeroberfläche entwickelt wird.

3.1 TUIO-Protokoll

Das TUIO-Protokoll [Kaltenbrunner, Martin et al. (2005)] bildet das Kommunikationsinterface zwischen der Multitouch-fähigen Küchentheke und den darunter liegenden Anwendungsschichten. Es ist dafür ausgelegt, die Zustände der Objekte, die sich auf der Multitouch-Oberfläche befinden, zu überwachen. Diese Objekte werden durch ein Sensorsystem erfasst, wodurch sich ihre Position und Orientierung bestimmen lässt. Zusätzlich zu den Objekten, die eine eindeutige ID besitzen, gibt es noch den Cursor, dieser ist lediglich durch seine Position bestimmt. Der Cursor definiert somit einen einfachen *Blob*, der durch eine Berührung mit dem Finger hervorgerufen wird. Das flexible Design von TUIO bietet jedoch Methoden, um zu selektieren, welche Informationen gesendet werden sollen.

Im TUIO-Protokoll wurden zwei Hauptklassen definiert, die sogenannte *set message* und die *alive message*. Zusätzlich zu den Hauptklassen wurde die *fseq message* eingeführt.

Set messages enthalten Informationen über den Status, die Position, die Orientierung sowie andere erkannte Zustände. Sie werden erzeugt, wenn ein neues Objekt hinzugefügt wird oder sich der Zustand eines vorhandenen Objektes verändert.

Alive messages enthalten eine Liste eindeutiger *Session IDs*, die alle Objekte darstellen, die sich derzeit auf der Multitouch-Oberfläche befinden. Die Nachrichten werden periodisch und beim Entfernen eines Objektes erzeugt.

Fseq message ist eine aufsteigende Zahl mit der jedes UDP-Paket versehen wird. Die Nummer ist für alle Pakete gleich, die Daten enthalten, die zu einem bestimmten Zeitpunkt aufgenommen worden sind.

Um einer Verzögerung bei der Datenübertragung vorzubeugen, wird das UDP-Protokoll für die Kommunikation mit der darunter liegenden Anwendungsschicht verwendet. Auf Grund des möglichen Paketverlustes, der bei der Übertragung mittels UDP auftreten kann, und den damit verbundenen inkonsistenten Daten, wurde auf die Implementierung von expliziten *add* oder *remove messages* verzichtet. Der Empfänger muss also anhand von *set messages* und *alive messages* feststellen, ob ein Objekt hinzugefügt oder entfernt wurde. Aus Gründen der Effizienz wird der Raum im UDP-Paket voll ausgenutzt. Dafür werden mehrere *set messages* zu einem Paket zusammengefasst und mit einer zusätzlichen *alive message* im UDP-Paket verstaut. Bei einer großen Anzahl von Objekten, ist es möglich, dass der Platz im UDP-Paket nicht ausreicht. In diesem Fall werden mehrere Pakete verschickt, von denen jedes eine *alive message* enthält. Zuletzt wird jedes Paket mit einer *frame sequence ID (fseq) message* versehen. Damit der Empfänger auch bei einer inaktiven Benutzeroberfläche eine konsistente Liste von Objekten gewährleisten kann, werden, abhängig von der Qualität der Verbindung, in einem festgelegten Zeitintervall, z.B. jede Sekunde, *alive messages* verschickt.

3.1.1 Nachrichtenformat

Seit der Implementierung der *Open Sound Cotrol (OSC)* [Wright, Matthew et al. (2003)] nutzt TUIO auch dessen generelle Syntax. Für jeden Nachrichtentyp können unterschiedliche Profile angelegt werden, je nach Profil werden unterschiedliche Informationen in der *set message* verschickt.

Der Empfänger muss auf folgende Nachrichten achten:

/tuio/[profileName] set sessionID [parameterList]

/tuio/[profileName] alive [list of active sessionIDs]

/tuio/[profileName] fseq (int32)

Die Parameter der *set message* stellen die Eigenschaften eines Objektes dar, einige der Eigenschaften (ID, Position und Ausrichtung) werden direkt durch den Sensor erfasst. Andere (Geschwindigkeit und Beschleunigung) werden durch zeitliche Informationen der vorher genannten Eigenschaften ermittelt.

s	sessionID, temporary object ID, int32
i	classID, fiducial ID number, int32
x, y, z	position, float32, range 0...1
a, b, c	angle, float32, range 0...2PI
X, Y, Z	movement vector (motion speed & direction), float32
A, B, C	rotation vector (rotation speed & direction), float32
m	motion acceleration, float32
r	rotation acceleration, float 32
p	free parameter, type defined by OSC packet header

Tabelle 1. Semantische Typen der *set message*
Kaltenbrunner, Martin et al. (2005)

Die *sessionID* ist für jedes Objekt eindeutig, dies ist notwendig, um die Objekte auch dann zuordnen zu können, wenn mehrere Objekte mit derselben *classID* gleichzeitig auf der Benutzeroberfläche angezeigt werden. Außerdem lassen sich dadurch unbekannte Objekte in aufeinanderfolgenden Frames identifizieren.

3.1.2 Profile

Im TUIO-Protokoll sind drei Hauptprofile (2D, 2.5D, 3D) angegeben, hinzu kommen zwei Profile (raw, custom) bei denen sich die Parameter definieren lassen. Im Profil wird zwischen einer Objekt- und einer Cursor-Nachricht unterschieden, je nach Typ werden unterschiedliche Parameter übergeben.

Die Profile sind wie folgt definiert:

2D Interactive Surface

/tuio/2Dobj set s i x y a X Y A m r
/tuio/2Dcur set s x y X Y m

2.5D Interactive Surface

/tuio/25Dobj set s i x y z a X Y Z A m r
/tuio/25Dcur set s x y z X Y Z m

3D Interactive Surface

/tuio/3Dobj set s i x y z a b c X Y Z A B C m r
/tuio/3Dcur set s x y z X Y Z m

raw profile

/tuio/raw_[profileName]
/tuio/raw_dtouch set i x y a

custom profile

/tuio/_[formatString]
/tuio/_sixyP set s i x y 0.57

Die Parameter für eine Objekt-Nachricht des *raw profiles* sind durch den jeweiligen Sensor definiert, der zum Erfassen der Objekte verwendet wird.

3.2 Windows 7 Touch

Für die Arbeit an einem Multitouch-Table benötigt man ein spezielles Betriebssystem, das sich darauf versteht die Gesten und Eingaben, die mit den Fingern erzeugt werden, zu verarbeiten. Microsoft bietet in der neuesten Version seines Betriebsystems *Windows 7 Touch* diese Möglichkeiten und stellt dafür einige grundlegende Gesten bereit, die auf der Projekthompage [Microsoft Corporation (2010b)] beschrieben werden. Zusätzlich zu den bereitgestellten Gesten wurden einige Anwendungen, darunter Wordpad, Paint, die Foto-Gallery oder der Internet Explorer, für die Verwendung mit Multitouch-Gesten optimiert [Abbildung 1]. Da nun keine physikalische Tastatur mehr existiert, werden Texte über die

Abbildung 1: Verschieben von Objekten in Windows 7 [Neumann, Kai (2010)]

Bildschirmtastatur von *Windows 7 Touch* eingegeben. In *Windows 7 Touch* wird zwischen zwei Arten von Gesten unterschieden, zum einem die *Navigationsgeste* und zum anderen die *Bearbeitungsgeste*. Beispielsweise wird beim Navigieren in einer Webseite durch eine Fingerbewegung nach oben nach unten gescrollt und bei einer Fingerbewegung nach unten wird nach oben gescrollt. Der Rechtsklick der Maus lässt sich durch ein einfaches halten des Fingers, an einer bestimmten Position, auslösen.

Eine detailliertere Aufstellungen aller Gesten von *Windows 7 Touch* befindet sich Abbildungsverzeichnis[3] und auf [Microsoft Corporation (2010c)].

Architektur von *Windows 7 Touch*

Im Folgenden wird ein kleiner Überblick über die *Windows Touch API* geben und aufgezeigt wie sie sich in die *Windows 7 Architektur* einbettet.

Windows 7 Touch interpretiert die Nachrichten, die von der *Multitouch-Hardware* generiert werden, um diese anschließen an die Anwendung weiterzuleiten.

[3] [Abbildung 8]

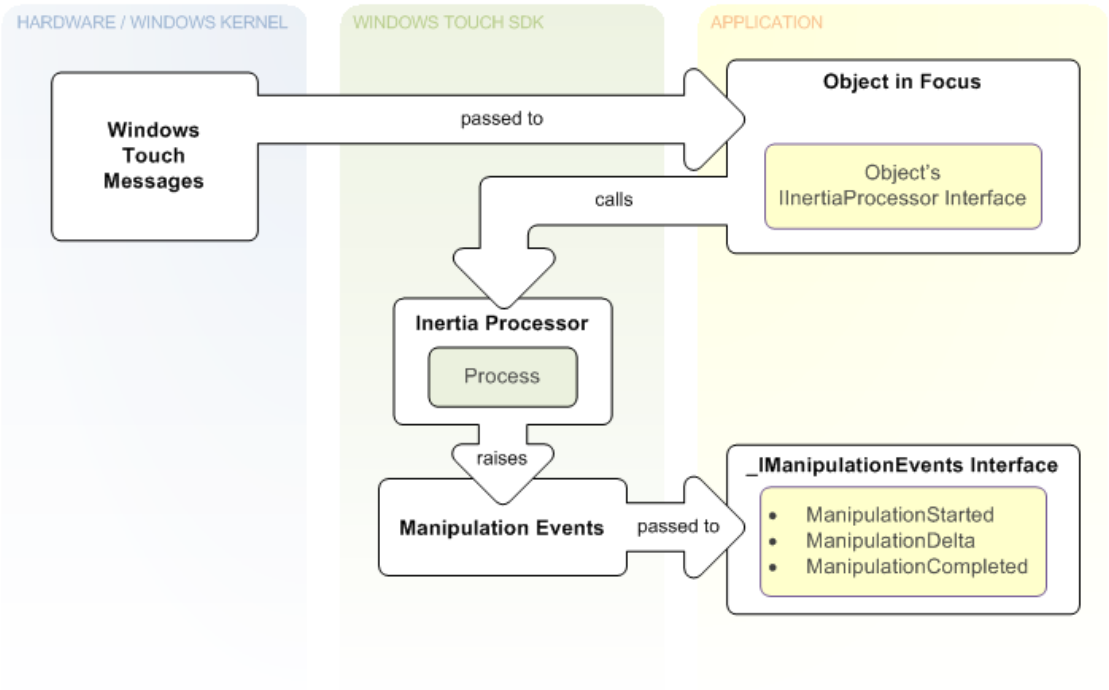

Abbildung 2: Windows 7 sendet Nachrichten von der Hardware an die Anwendung [Microsoft Corporation (2010a)]

In der linken Spalte der Abbildung[4] empfängt die Multitouch-Hardware eine Benutzereingabe, daraufhin kommuniziert der Hardware-Treiber mit dem Betriebssystem. Empfängt das Betriebssystem eine solche Nachricht, erzeugt es daraus eine *WM_TOUCH* oder *WM_GESTURE* Nachricht und sendet diese an die Anwendung weiter. Um Nachrichten aus der Zeit der physikalischen Benutzereingaben zu unterstützen, mappt *Windows 7 Touch* diese auf eine passende *WM_GESTURE* und leitet sie mittels *SEND* oder *POST* an die Anwendung weiter. Eine detaillierte Aufstellung der gemappten „Altgesten“ findet sich in der Abbildung[5].

Die verschickten Nachrichten enthalten X- und Y-Koordinaten, um das Objekt zu identifizieren, welches im Fokus des Nutzers liegt. Dieses Objekt ruft dann den sogenannten *Inertia Processor* auf der die physikalischen Berechnungen durchführt, die z.B. beim Verschieben eines Objekten anfallen. Das *Inertia Processor-Interface* ruft dafür mit Hilfe eines Timers oder einer Schleife die *Process*- oder *ProcessTime*-Methode in einem separaten Thread auf. Während der Bearbeitung werden *ManipulationsEvents* vom Typ *_IManipulationEvents* geworfen, diese dienen dem Programmierer dazu, in die Manipulation der Objekte einzugreifen. Außerden wird abhängig von den Events das UI dazu veranlasst, sich zu aktualisieren.

3.3 Windows Presentation Foundation 4

In diesem Abschnitt wird die *Windows Presentation Foundation* vorgestellt und eine kleine Einführung in das Framework gegeben. Im Projekt bildet es die Grundlage für neue Anwendungen, die für eine Bedienung durch Gesten optimiert sind.

[4] [Abbildung 2]
[5] [Abbildung 3]

Gesture	Description	Message(s) Generated
Pan	The pan gesture maps to using the scroll wheel.	**WM_VSCROLL** **WM_HSCROLL**
Press and Hold	The press and hold gesture maps to right clicking the mouse.	**WM_RBUTTONDOWN** **WM_RBUTTONUP**
Zoom	The zoom gesture triggers messages that are similar to holding the CTRL key, and spinning the mouse wheel to scroll.	**WM_MOUSEWHEEL** with MK_CONTROL set in the *lParam*

Abbildung 3: Gemappt Altgesten [Microsoft Corporation (2010c)]

Die *Windows Presentation Foundation*, früher unter dem Namen „Avalon“ bekannt, ist für die grafischen Aspekte bei der Entwicklung von .Net 4.0 Anwendungen verantwortlich. Sie enthält unter anderem Klassen für die Darstellung von 2D- und 3D-Grafikgen und Animationen. Eine Neuerung bildet die Sprache *XAML* (*eXtensible Application Markup Language*), mit der sich Oberflächen, neben C#- oder Visual Basic-Code, in einer XML-Syntax beschreiben lassen. Durch diese verbesserte Trennung von Logik und Darstellung, können sich Grafikdesigner um die Benutzeroberfläche kümmern. Eine weitere Neuerung ist der Umstand, dass UI-Komponenten nicht mehr durch das Betriebssystem gezeichnet werden, sondern durch die WPF. Dadurch entfallen einige Limitierungen und die Darstellungen können effizienter aktualisiert werden.

Da die WPF mit Versionsnummer 3 ins Leben gerufen wurde, werden hier kurz die Verbesserungen aufgezeigt, die seitdem im Framework implementiert wurden.

- Neue Steuerelemente: unter anderem DataGrid, Calendar, DatePicker und ein nativer Web-Browser.
- Verbesserungen beim Zeichnen von 2-D Grafiken und beim Rendern.
- Einfache Handhabung von Animationen, was für die Multitouch-fähige Küchentheke von besonderer Bedeutung ist, da sich die Bedienung nach Möglichkeit wie auf einem „echten“ Tisch „anfühlen“ sollte.
- Durch den *Visual State Manager* ist es einfach möglich das Design der Steuerelemente zu veränden, um sie z.B. an das Design der HAW-Hamburg anzupassen.
- WPF unterstützt *Windows 7 Multitouch*, wodurch sich z.B. Bilder ohne Änderungen am Quellcode durch Gesten skalieren, drehen, verschieben und zoomen lassen.

Architektur von WPF

WPF nutzt eine mehrschichtige Architektur, wie in der Abbildung[6] zu sehen ist.

Die *Managed WPF API*-Schicht beinhaltet grafische Typen wie Fenster, Panels und verschiedene Controls. Außerdem sind dort die übergeordneteren Abstraktionen und Stile definiert. Diese Schicht beinhaltet somit alle Klassen und Typen, die man direkt bei der Implementierung, der grafischen Benutzeroberfläche, verwendet.

[6] Abbildung 4

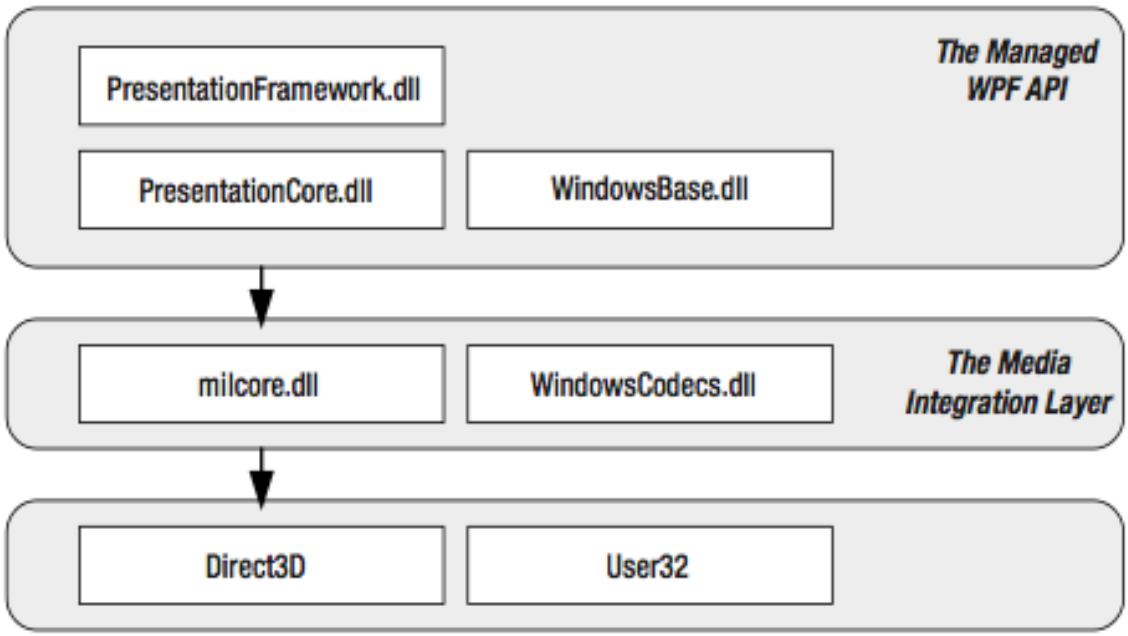

Abbildung 4: Architektur von der WPF 4 [Matthew MacDonald (2010)]

In der *Media Integration Layer*-Schicht wird z.B. die Umwandlung von .Net Objekten in *Direct3D Texturen* durchgeführt. Da dieser Bereich direkt mit der *Render*-Schicht kommuniziert und extrem performanceabhängig ist, ist er in nativem Code geschrieben. Hinzu kommt noch eine API, die die Unterstützung für das Darstellen und Skalieren von Bildern bereitstellt.

In der untersten Ebene wird **alles** gerendert was in WPF zu zeichnen ist, ob Videos, Bilder oder Komponenten. Dadurch ist es unerheblich welche Grafikkarte man besitzt, was man zeichnen möchte und auf welchem Windows Betriebssystem die Anwendung laufen soll. Außerdem werden sogar zweidimensionale Objekte und normaler Text in Dreiecke umgewandelt und durch die 3D-Pipeline geschickt, dadurch muss nicht auf GDI+ oder User32 zum zeichnen von zweidimensionalen Objekten zurückgegriffen werden.

Eine detaillierte Aufstellung der Komponenten der verschiedenen Schichten finden Sie im Buch von MacDonald[7]

4 Abgrenzung zu eigenen Arbeitszielen

Dieses Projekt beschränkt sich vorrangig auf die Entwicklung von Anwendungen, die für die Nutzung auf der Multitouch-fähigen Küchentheke optimiert sind. Ein mögliches Szenario stellt der Außendienstplaner dar, den Barnkow[8] vorgestellt hat.

Die Inhalte des Projekts partizipieren nicht an der technischen Umsetzung der Küchentheke oder der Implementierung des TUIO-Protokolls, da für diese Arbeitsbereiche bereits andere Kommilitonen zuständig sind.

[7] [Matthew MacDonald (2010) S.13].
[8] [Barnkow, Lorenz (2009)]

5 Zusammenfassung

Es wurden die Problemstellungen aufgezeigt, die bei einer Gruppenarbeit an einer so großen Multitouch-fähigen Küchentheke auftreten. Für diese Probleme hat Barnkow[9] Anforderungen und Lösungsansätze vorgestellt, die für einen effizienten und reibungslosen Ablauf bei der Arbeit an einem Tabletop unabdingbar sind.

Im zweiten Abschnitt wurde die Schnittstelle zum Betriebssystem, das Betriebssystem als solches und das Framework, auf dem die Anwendungen aufbauen, vorgestellt. Neben dem detaillierten Einblick in das TUIO-Protokoll [Kaltenbrunner, Martin et al. (2005)], welches die Kommunikation zwischen der Hardware und dem Betriebssystem herstellt, wurden die Architekturen, auf denen die Verarbeitung der Nachrichten im Betriebssystem und die Anwendungen aufbauen, dargestellt.

Die variablen Profile des TUIO-Protokolls bilden einen perfekten Unterbau für die Übermittlung der Informationen, die vom Sensor geliefert werden. Durch die Unterstützung von Multitouch in *Windows 7 Touch* ist es als Plattform für die Software gut geeignet und verringert, dank der eingebauten Gesten, den Programmieraufwand. Die Windows Presentation Foundation in Version 4 stellt umfangreiche Funktionen bereit, um grafisch anspruchsvolle Anwendungen auf einem Tabletop zu entwickeln. Die integrierten Animationen vermitteln dem Nutzer außerdem ein realistischeres Gefühl, z.B. beim Verschieben von Fotos, was für die Akzeptanz eine große Rolle spielt.

Literatur

[Barnkow, Lorenz 2009] Barnkow, Lorenz: *Ausarbeitung zur Veranstaltung Anwendungen 1*. 2009. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master09-10-aw1/barnkow/bericht.pdf. – Letzter Aufruf am 11 Februar 2011

[Barnkow, Lorenz 2010] Barnkow, Lorenz: *Ausarbeitung zur Veranstaltung Anwendungen 2*. 2010. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2010-aw2/barnkow/bericht.pdf. – Letzter Aufruf am 11 Februar 2011

[Dirk Frischalowski 2007] Dirk Frischalowski: *Windows Presentation Foundation*. München : Addison-Wesley Verlag, 2007

[HAW-Hamburg 2009] HAW-Hamburg: *Living Place Hamburg - Projektbeschreibung*. Living Place Hamburg, Department Informatik, HAW Hamburg, Berliner Tor 11, 20099 Hamburg, Germany. 2009. – URL http://livingplace.informatik.haw-hamburg.de/content/LivingPlaceHamburg_en.pdf. – Letzter Aufruf am 11 Februar 2011

[Kaltenbrunner, Martin et al. 2005] Kaltenbrunner, Martin et al.: *TUIO: A Protocol for Table-Top Tangible User Interfaces*. 2005. – URL http://www.audiomulch.com/~rossb/writings/tuio_gw2005.pdf. – Letzter Aufruf am 11 Februar 2011

[Klingenberg, Thole 2008] Klingenberg, Thole: *Mustererkennung und Protokolle für Multitouch-Geräte*. 2008. – URL http://tap.informatik.uni-oldenburg.de/downloads/Seminarausarbeitung-Mustererkennung_und_Protokolle_fuer_Multitouch_Geraete.pdf. – Letzter Aufruf am 11 Februar 2011

[Matthew MacDonald 2010] Matthew MacDonald: *Pro WPF in C# 2010: Windows Presentation Foundation in .NET 4.0*. New York : Paul Manning, 2010

[9] [Barnkow, Lorenz (2010)]

[Microsoft Corporation 2010a] MICROSOFT CORPORATION: *Windows 7 - Architectural Overview*. 2010. – URL http://msdn.microsoft.com/en-us/library/dd371413(VS.85).aspx. – Letzter Aufruf am 01 März 2011

[Microsoft Corporation 2010b] MICROSOFT CORPORATION: *Windows 7 - Verwenden von Gesten*. 2010. – URL http://windows.microsoft.com/de-DE/windows7/Using-touch-gestures. – Letzter Aufruf am 01 März 2011

[Microsoft Corporation 2010c] MICROSOFT CORPORATION: *Windows Touch Gestures Overview*. 2010. – URL http://msdn.microsoft.com/de-de/library/dd940543(v=VS.85).aspx. – Letzter Aufruf am 01 März 2011

[Neumann, Kai 2010] NEUMANN, KAI: *Interaktionsmöglichkeiten an einem Multitouch-Table durch die Analyse von Bewegungsgrößen der Benutzereingabe*. 2010. – URL http://dok.bib.fh-giessen.de/opus/volltexte/2010/4197/pdf/Kai_Neumann_Diplomarbeit.pdf. – Letzter Aufruf am 11 Februar 2011

[Rahimi, Mohammadali und Vogt, Matthias 2009] RAHIMI, MOHAMMADALI UND VOGT, MATTHIAS: *Projektbericht: Aufbau des Living Place Hamburg*. 2009. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master09-10-proj/rahimi-vogt.pdf. – Letzter Aufruf am 11 Februar 2011

[Wright, Matthew et al. 2003] WRIGHT, MATTHEW ET AL.: *Open Sound Control: State of the Art 2003*. 2003. – URL http://opensoundcontrol.org/files/Open+Sound+Control-state+of+the+art.pdf. – Letzter Aufruf am 13 Februar 2011

Abbildungsverzeichnis

Abbildung 5: Aussenansicht des Gebäudes [HAW-Hamburg (2009)]

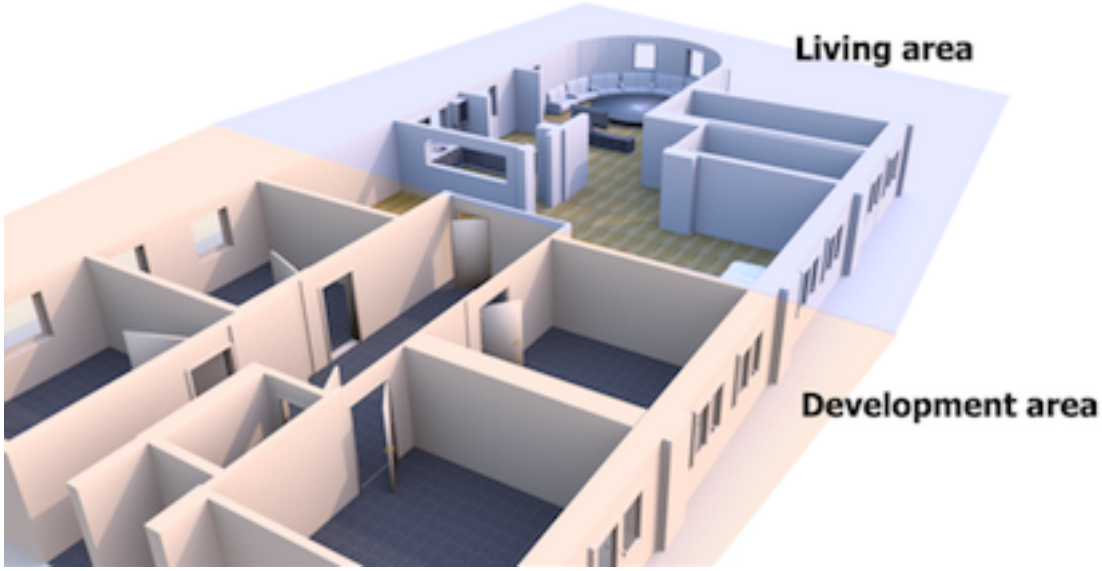

Abbildung 6: Raumaufteilung der Wohnung [HAW-Hamburg (2009)]

Abbildung 7: Grundgerüst der Küchentheke [Barnkow, Lorenz (2010)]

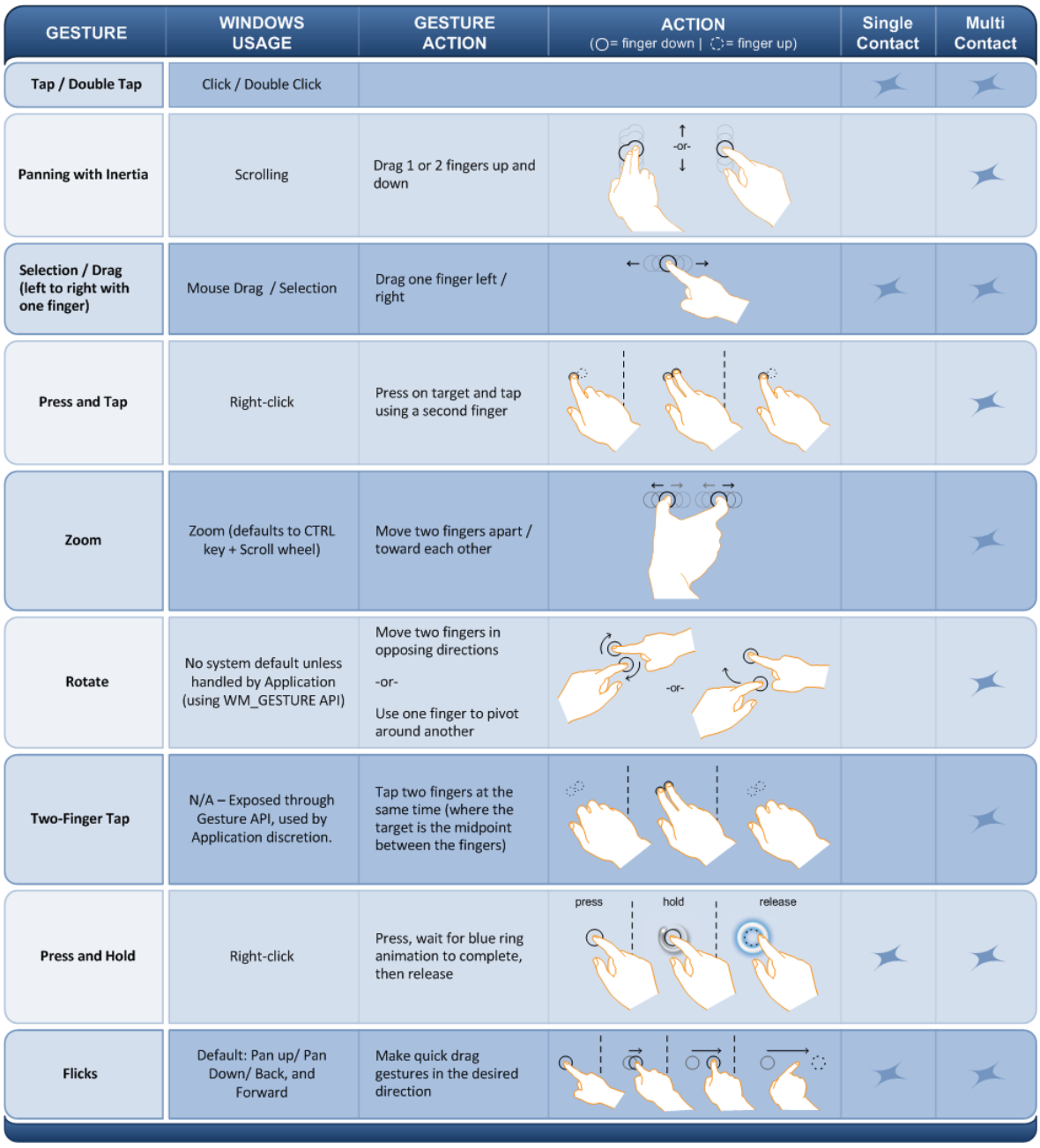

Abbildung 8: Windows Touch Gestures Overview [Microsoft Corporation (2010c)]