Tangible Programming Interfaces

Auf dem Weg zu programmierbaren Wohnumgebungen

Dejan Pilav

Betreuer: Marc Langheinrich

Abstract

Billige Mikroprozessoren in Haushaltsgeräten geben uns immer mehr Kontrolle über Vorgänge in unseren Wohnumgebungen. Der Benutzer, der die entsprechenden Applikationen nutzt, kann aber leicht überfordert werden von der Fülle an Möglichkeiten, die diese bieten. Möglicherweise lässt sich diese Problematik aber schon durch geeignete Schnittstellen zur Programmierung der entsprechenden Geräte lösen. In diesem Bericht werde ich versuchen aufzuzeigen, was die konkreten Herausforderungen auf diesem Gebiet sind, und inwiefern sich tangible programming dazu eignet, ihnen zu begegnen. Im zweiten Abschnitt stelle ich dann ein aktuelles Forschungsprojekt vor.

1 Einführung

1.1 Problemstellung

Durch das Aufkommen von billigen, ubiquitären Mikroprozessoren sehen wir uns immer öfter mit interaktiven Umgebungen konfrontiert. Insbesondere die Unterhaltungs- und Haushaltselektronik gibt uns immer mehr Kontrolle über verschiedene Abläufe in unseren Wohnungen. Dieser Trend wird sich in den nächsten Jahren voraussichtlich fortsetzen. Ein wesentlicher Faktor, der dazu beiträgt, ist, dass Prozessoren nach wie vor immer billiger und damit auch immer verbreiteter werden. Ein anderer ist, dass der Einbau von weiteren Softwarefunktionen in Geräte, in welchen Speicher und Prozessor bereits vorhanden sind, billig ist, gleichzeitig aber dazu dienen kann, sich von der Konkurrenz zu differenzieren.

Scheinbar unvermeidlich steigt parallel zur Mächtigkeit auch die Komplexität der Bedienung für den Endbenutzer. Es gibt Vorschläge den erforderlichen Input durch den Benutzer auf ein Minimum durch sinnvolle Standardvorgaben und machine learning Algorithmen zu beschränken. Dieser Lösungsansatz wird das Problem aber in absehbarer Zeit nicht lösen können. Denn der Bewohner möchte seine Wohnumgebung entsprechend seinem Lebensstil, der sich unter Umständen schnell ändern kann, konfigurieren. Manchmal lassen sich seine momentanen Vorlieben schlicht nicht erraten, egal wie ausgefeilt die Algorithmen auch sein mögen. Ein weiteres Problem ist, dass Haushaltsgeräte inzwischen oft

Netzwerkschnittstellen anbieten. Da laufend neue Arten von Geräten erscheinen, haben wir eine richtige kombinatorische Explosion von möglichen Interaktionen zwischen Maschinen. Diese müssen aber oft vom Endbenutzer genauer spezifiziert werden, da es unmöglich ist für alle Kombinationen der Geräte ein sinnvolles Standardverhalten zu definieren. Daraus folgt, dass wir zumindest vorerst auf die Programmierung durch den Benutzer nicht verzichten können.

1.1 Tangible Programming

Vielleicht ist die oben erwähnte steigende Komplexität für den Benutzer aber auch nur ein Problem der Benutzerschnittstelle. Insbesondere die „dinghaften“ Schnittstellen(„tangible interfaces“) können unter Umständen helfen, die Kluft zwischen der virtuellen und der realen Welt zu schliessen. Ulmer und Ishii definieren diesen Begriff folgendermassen: „[Tangible interfaces are] user interfaces employing physical objects, surfaces, and spaces as representations and controls for computationally mediated associations.” [1] Suzuki und Kato haben daraus abgeleitet den Begriff „tangible programming language“ geprägt. Im folgenden werde ich mich beim Begriff „tangible programming“ aufs Programmieren durch tangible interfaces beziehen. Dabei beschränkt sich das Wort „Programmieren“ nicht aufs klassische Entwickeln von Software, sondern meint oft auch das Konfigurieren und Personalisieren. (So „programmiert“ man etwa auch den Videorekorder im allgemeinem Sprachgebrauch wenn man eine Sendung aufnehmen will, auch wenn viele Softwareingenieure diese Tätigkeit nicht derart bezeichnen würden.)

Sind diese tangible interfaces aber die richtige Antwort auf das im vorigen Abschnitt genannte Problem? Ich bin der Auffassung, dass sie das Potenzial dazu haben. Vielen Menschen fällt das Lernen leichter, wenn sie ihren Körper in den Lernprozess einbringen können. In dem wir physische Objekte manipulieren, anstatt virtuelle auf dem Bildschirm, nehmen wir auch das Programmieren aus einem für manche Menschen abschreckenden Kontext der Computerwelt. Schliesslich erlaubt uns diese direkte Interaktion mit der Schnittstelle in der Regel auch eine unkomplizierte Zusammenarbeit von mehreren Menschen beim Ausüben der Tätigkeit, was sowohl den Prozess der Wissensvermittlung vereinfachen, wie auch den Vorgang an sich interessanter und unterhaltsamer machen kann.

Abb. 1: LogoBlocks: Benutzer bei der Zusammenarbeit

1.2 Herausforderungen

Bevor wir dem Endbenutzer eine zufriedenstellende Lösung zur Programmierung von häuslichen Umgebungen zur Verfügung stellen können, gilt es, einige Probleme zu lösen. Ich möchte im Folgenden die wichtigsten vorstellen.

- **Veränderungen der Gerätelandschaft**

  Bisherige Studien haben gezeigt, dass die Bewohner eines Hauses konstant ihre Wohnumgebung verändern, um sie an ihre Bedürfnisse anzupassen. Das schliesst auch die Technologien ein, die dort benutzt werden. [2] Edwards und Grinter wiederum weisen darauf hin, dass das Haus der Zukunft in einem evolutionärem Prozess entstehen wird, Stück für Stück, anders als bei vielen aktuellen Forschungsprojekten, bei denen von Anfang an ein bestimmtes Konzept vorliegt und umgesetzt wird. [3] Dies, und die oben erwähnte kombinatorische Explosion von Interaktionsmöglichkeiten zwischen immer neuen Geräten, legen uns nahe, unsere Programmiersprache mächtig und flexibel zu gestalten, um alle mögliche Arten von Komponenten, die es im Moment möglicherweise noch gar nicht gibt, integrieren zu können. Es ist allerdings nicht klar ersichtlich, wie mächtig sie dazu mindestens sein muss. Dies wäre aber nützlich zu wissen, um die Komplexität der Sprache möglichst niedrig zu halten.

Heterogene Benutzerlandschaft

Der grösste Teil der bisherigen Forschung um die Programmierung durch den Endbenutzer hat sich an mehr oder weniger klaren Zielgruppen orientiert, wie etwa an Büroangestellten oder Grundschülern. Wir sprechen bei der Programmierung von Wohnumgebungen eine sehr breite Benutzerbasis an. Eigentlich alle Menschen, die in Häusern wohnen, und diesseits der digitalen Kluft leben. Es ist selbstverständlich, dass wir dementsprechend mit einer grossen Bandbreite an Vorkenntnissen und kognitiven Fähigkeiten bei den Endbenutzern rechnen müssen. Es ist darum nicht klar, welche Anforderungen wir an sie stellen dürfen. Andererseits wissen wir, dass bei der Haushaltselektronik die Toleranz für nicht ergonomische Lösungen gering ist, die Benutzer also ihrerseits hohe Ansprüche an unsere Lösung haben werden.

Abstraktionen in der Tangible Welt

Wenn wir programmieren, befassen wir uns unweigerlich mit Abstraktionen. Das ist sogar trivial wahr, wenn wir das durch eine Tastatur und einen Bildschirm machen, denn der Bildschirm kann uns höchstens Repräsentationen von Objekten aus der realen Welt anzeigen. Indem wir berührbare Schnittstellen benutzen, scheint intuitiv klar, dass wir einen grossen Teil der Abstraktion verschwinden lassen können. Komplett geht das aber nicht; einige Dinge lassen sich nun mal nicht berühren. Einerseits wären das Abstraktionen über Zeit(„Wie sage ich dem Videorekorder, er soll morgen meine Sendung aufnehmen?“), andererseits Abstraktionen über Klassen(„Der Videorekorder soll alle Star Trek Episoden in der chronologischen Reihenfolge abspielen.“). Wie integrieren wir nun solche Abstraktionen in unserer tangible Programmiersprache?

Im nächsten Abschnitt werden wir uns ein Forschungsprojekt aus der tangible programming interfaces Forschung anschauen, und sehen, wie dort diesen Herausforderungen begegnet wird.

2 Tangible Programming

2.1 Einführungsbeispiel - LogoBlocks

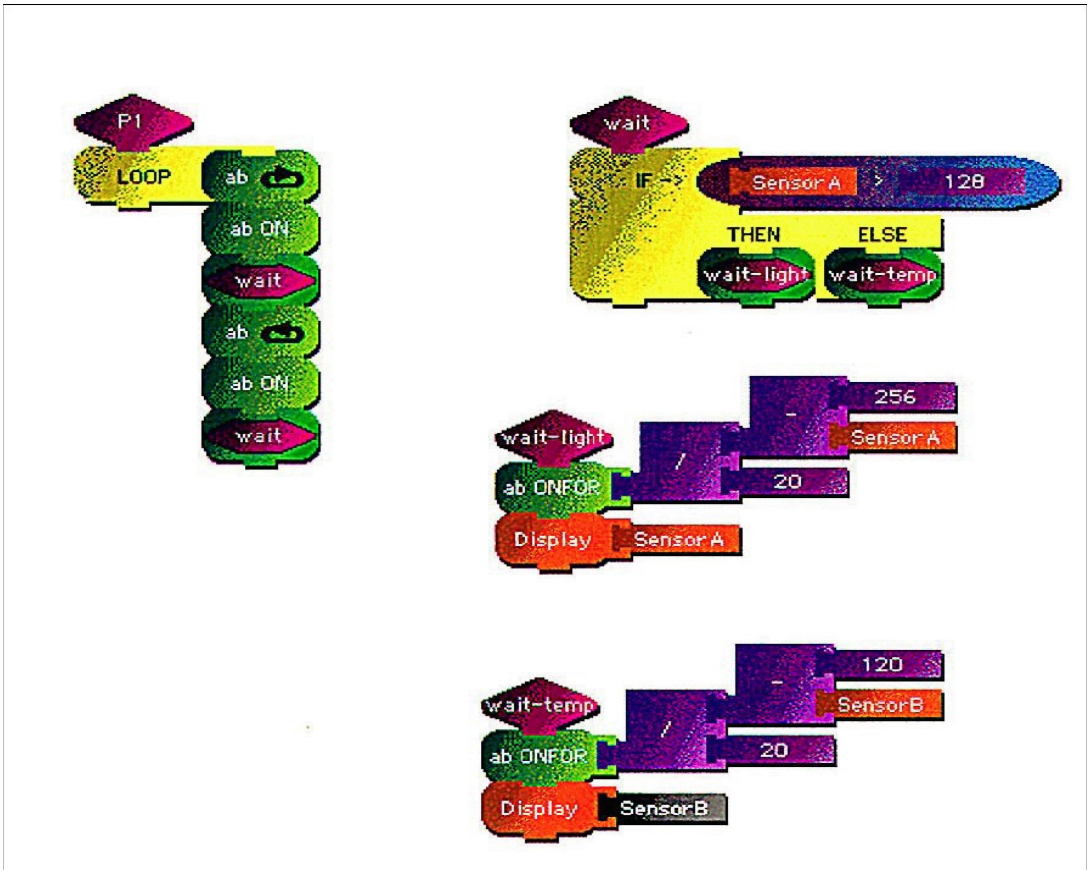

Abb. 2: LogoBlocks: Virtuelle Blöcke

LogoBlocks ist eine visuelle Programmiersprache, die auf Logo basiert. [4] Visuelle Programmiersprachen unterscheiden sich von klassischen wie C++ dadurch, dass sie das Programmieren in mehreren Dimensionen unterstützen, im Gegensatz zu nur einer Dimension einer Textdatei. Ein prominentes Beispiel dafür sind Teile des Visual Studio Pakets, und andere RAD Umgebungen. Das Design der LogoBlocks Umgebung entspricht Puzzlestücken, die man zusammenstecken, und somit Codeblöcke erstellen kann. Am Anfang haben die Autoren diese Puzzles noch auf dem Bildschirm angeordnet, später haben sie metallene Würfel von etwa 20 cm Kantenlänge physisch zusammengesteckt, wie man das in der Abbildung 1 sieht. Der Aufbau der Sprache selber sollte jedem, der sich schon einmal mit imperativen Programmierung beschäftigt hat, ziemlich schnell ersichtlich sein. Im obigen Beispiel bewegt sich ein Motor schneller oder langsamer, abhängig vom Input der Sensoren.

Eine Schwierigkeit, der man oft im tangible programming Bereich begegnet, ist die fixe Form der zu manipulierenden Objekte. Auch in LogoBlocks ist es schwierig manche Programme physisch zusammenzustecken ohne einige Blöcke nur um Abstand zu halten benutzen zu müssen. In der Bildschirmvariante davon wird das dadurch gelöst, dass man die

virtuellen Blöcke strecken kann, um ihre Grösse anzupassen. Möchte man ähnliches mit den physischen Klötzen versuchen, stellt man eine substanzielle Herausforderung an die Designer.

2.2 AutoHAN: An Architecture for Programming the Home

Im diesen Abschnitt stelle ich das AutoHAN Projekt von Blackwell und Hague vor. [5] Dieses erhebt den Anspruch, ein betriebsbereites Heimnetzwerkkonzept mit einer vielseitigen tangible programming Schnittstelle zu sein. Auf diese werde ich mich im Folgenden konzentrieren, und die sonstige zur Verfügung gestellte Infrastruktur zum grössten Teil aussen vor lassen.

2.2.1 Kognitive Anforderungen an die Lösung

In diesem Projekt haben die Autoren Gewicht darauf gelegt, theoretische Richtlinien für ihren Lösungsansatz aufzustellen. Insbesondere wollten sie damit die oben erwähnte Herausforderung der sehr heterogenen Benutzerbasis angehen. Sie stellten die Frage, wie viel Komplexität man dem durchschnittlichen Benutzer zutrauen kann. In vorangegangenen Studien haben sie herausgefunden, dass auch Menschen, die nur ungern Computer benutzen, vergleichbar oft Abstraktionen zur Organisation ihres Alltags verwenden, wie erfahrene Computerbenutzer. [6] Sie verwendeten etwa Ordner, Ablagen und Terminplaner, waren aber nur widerstrebend bereit, ihre Zeit in das virtuelle EDV Äquivalent zu investieren. Die Autoren haben zwei Schlüsse für ihre Lösung daraus gezogen. Einerseits haben sie einen Schwerpunkt auf direkte Manipulation bei der Programmierung gelegt, darum haben sie sich auch für eine tangible Schnittstelle entschieden. Auch andere Studien haben nämlich gezeigt, dass die Eigenschaft der direkten Manipulation hilfreich beim Erlernen einer neuen Programmiersprache ist. [7,8] Ausserdem wollten sie den Einstieg zu ihrem System möglichst einfach gestalten, indem sich der Benutzer graduell in die Komplexität der Programmiersprache einarbeiten kann. Deswegen kann man die „Media Cubes“, die die Tokens ihrer tangible programming Sprache darstellen, als eine sehr einfache Fernbedienung benutzen. Die Autoren sind davon ausgegangen, dass jeder normale Erwachsene fähig ist eine solche zu bedienen. Im nächsten Abschnitt werden wir uns diese Media Cubes genauer anschauen.

2.2.2 Media Cubes

Media Cubes sind buchstäblich Holzwürfel von etwa 10 cm Kantenlänge. Zuerst einmal verfügen sie über einen Knopf um die Benutzereingabe entgegenzunehmen. Zur Rückmeldung an den Benutzer haben sie eine einzelne LED und einen piezoelektrischen Wandler, um ein Klicken zu erzeugen eingebaut. Weiter benutzen sie eine Reihe von Infrarotsensoren und Sendern, um mit der AutoHAN Infrastruktur zu kommunizieren. Schliesslich sind an vier Seiten des Würfels Kurzstreckenantennen angebracht. Mit diesen können die Media Cubes Geräte erkennen, die ebenfalls mit solchen Antennen ausgestattet sind, Haushaltsgeräte etwa, und insbesondere andere Media Cubes.

Abb. 3: Media Cubes Prototypen

Media Cubes als Fernbedienungen

Als Einstieg werden die Benutzer ermutigt Media Cubes als eine Art „Ein-Knopf Fernbedienung“ zu betrachten. Dies funktioniert folgendermassen: Wenn ein Cube eine vordefinierte „Play-Pause“ Funktion repräsentiert, können wir ihn nahe an beispielsweise einen CD-Player halten, und dann den Knopf des Würfels drücken. Anhand des Funksignals von dem mit einer Kurzstreckenantenne ausgestatteten CD-Players identifiziert der Cube diesen, und teilt der AutoHAN Infrastruktur über den Infrarotsender mit, das er mit dem Gerät assoziert worden ist. Dieser Vorgang wird abgeschlossen durch das Klicken des piezoelektrischen Wandlers zur Rückmeldung an den Programmierer. Das nächste Mal, wenn der Benutzer den Knopf an diesem Cube drückt, sendet dieser ein Signal an die Infrastruktur, welche entsprechend der definierten Rolle des Würfels das „Play-Pause“ Kommando an das assozierte Gerät weiterleitet. Wenn sich der Endbenutzer an diese einfache Funktionalität gewöhnt hat, wird er eher bereit sein, sich mit den komplexeren Funktionen, die diese Infrastruktur bietet, zu beschäftigen.

Media Cubes als abstrakte Programmierbausteine

Cube Abstraktionen erstellt man, in dem man zwei oder mehr davon nebeneinander anordnet, und die AutoHAN Infrastruktur anweist, diese Konfiguration als ein Skript zu speichern. Die Cubes stellen zu diesem Zweck ihre Nachbarsbeziehungen selber mit Hilfe der Kurzstreckenantennen fest und leiten dann diese Information an AutoHAN weiter. Damit können wir eine grosse Anzahl von ein- und zweidimensionalen(und somit visuellen) Programmiersprachen nachbilden. So wäre es ein leichtes, die Cubes entsprechend zu beschriften um das LogoBlocks System zu realisieren. Die Autoren können sich dementsprechend auch ganz verschiedene Herangehensweisen an die Programmierung der von AutoHAN unterstützten Wohnumgebungen vorstellen. Als einzige Einschränkung fordern sie, dass ein kognitiv plausibler Weg vom Benutzen des einzelnen Cube als Fernbedienung zu Funktionen der höheren Stufe führt. Zwei Paradigmen haben sie allerdings genauer erforscht, und diese werden wir uns jetzt kurz anschauen

2.2.3 Programmierparadigmen in der AutoHAN Umgebung

Da AlgoBlocks eine relativ direkte Umsetzung von einem bekannten Programmierstil ist, hat es auch ähnliche kognitive Anforderungen an den Programmierer wie das Vorbild. Blackwell und Hague wollten deswegen mit neuen Herangehensweisen an das Problem experimentieren. Aus diesen Bemühungen sind zwei Lösungsansätze entsprungen. Der erste basiert darauf, die Cubes natürliche Kategorien repräsentieren zu lassen in einem System der ontologischen Abstraktionen. Der andere bedient sich der These, dass sich Abstraktionen natürlich durch die Sprache ausdrucken lassen. Wir werden uns im Folgenden beide anschauen.

Das ontologische Paradigma

Die Cubes repräsentieren jeweils einen Typus einer einfachen Ontologie, die Seiten des Würfels hingegen die möglichen Interaktionen mit diesem Typ. Solche Würfel sind in der Abbildung 4 dargestellt. Mit einem Beispiel möchte ich das System illustrieren.

Anhand von Erwartungen eines durchschnittlichen Benutzers haben die Autoren versucht,

sinnvolle Kategorien von Cubes zu spezifizieren. Ein „Event Cube“ etwa repräsentiert verschiedene Ereignisse, die von Sensoren, Handlungen der Bewohner oder einer automatisierten Aktion, wie dem Klingeln des Weckers, ausgelöst werden können. Mögliche Interaktionen mit einem solchen Cube sind etwa „An“ und „Aus“. Seine Seiten werden entsprechend beschriftet. Ein „Channel Cube“ wiederum kann für verschiedene Medien stehen, etwa Fernsehen, Radio, oder die private Videosammlung. Die einzelnen Seiten des Würfels können dabei den Inhalt des Mediums repräsentieren, abhängig vom assoziierten Gerät als Quelle oder Senke. So könnte man einen Event Cube mit der Türklingel assozieren, und neben die „An“ Seite davon einen Channel Cube stellen, der die Kamera an der Haustür repräsentiert. Neben diesem wiederum könnte man einen weiteren Channel Cube stellen, der für das Videoarchiv steht. Entsprechend wird ein solches Skript AutoHAN veranlassen beim Klingeln den Gast aufzunehmen und die Aufnahme im Videoarchiv abzulegen.

Das linguistische Paradigma

Beim linguistischen Paradigma sind die Seiten der Media Cubes mit verschiedenen Wörtern beschriftet, die die jeweilige Funktion beschreiben. Eine mögliche Variante davon wäre etwa der „Clone“ Cube. Einen solchen kann man mit anderen Cubes, Geräten, oder ganzen Cube Konfigurationen assozieren, ein Konzept ähnlich dem der Zeigervariablen in klassischen Programmiersprachen. Eine Erweiterung davon ist der „List“ Cube, der mehrere solche Zuweisungen zusammenfassen kann. Eine weitere Ausprägung könnte der „Time“ Cube sein, der zeitliche Vorgänge abbildet. Ein mögliches Szenario zur Benutzung von diesen ist, einen Clone Cube für ein Skript stehen zu lassen, und anschliessend den Time Cube mit ihm zu assozieren. Dabei könnte das Skript die Alarmanlage scharf machen, die Rollläden herunterfahren und die Heizung abstellen. Wenn ich das nächste Mal aus dem Haus gehe, aktiviere ich den Time Cube, der die entsprechenden Schritte 10 Minuten später veranlasst.

2.2.4 Fazit AutoHAN und Media Cubes

Die AutoHAN Umgebung ist betriebsbereit, und die Media Cubes ermöglichen somit umfassende Forschung auf dem Gebiet der kognitiven Psychologie und Benutzerschnittstellen. Viele Fragen sind nach wie vor offen, eine wichtige mag sein, ob der Benutzer überhaupt bereit ist, sich mit diesen kleinen Würfeln auseinanderzusetzen. Es wäre auch gut zu wissen, welches Programmierparadigma zum Zweck der Programmierung der Wohnumgebungen geeignet ist. Diese Fragestellung ist natürlich eng verbunden mit der Frage nach der nötigen Mächtigkeit der Programmiersprache. Und schliesslich könnte sich die Architektur der Media Cubes als viel zu dynamisch erweisen. Wird der Benutzer die Übersicht darüber behalten können welche Cubes mit welchen Funktionen, Geräten und Skripten assoziert sind, sowohl während er programmiert, wie auch später bei der Benutzung?

3 Schlussfolgerungen

Es ist kaum anzuzweifeln, dass wir es in der Zukunft immer öfter mit smarten, vernetzten Geräten in unseren Haushalten zu tun haben werden. Die Anforderungen an eine Applikation, welche uns das Konfigurieren davon mindestens zu einem grossen Teil ersparen könnte, sind immens hoch. Darum sind wir mindestens in den nächsten Jahren auf die Programmierung

durch den Endbenutzer angewiesen.

Die diesbezügliche Forschung im Kontext der Wohnumgebungen ist aber noch relativ jung. Das tangible programming scheint mir dabei ein vielversprechender Ansatz zu sein. Es gibt genug Indizien, das solche Programmierschnittstellen die Bedienung der Systeme vereinfachen und den Lernprozess beschleunigen. Es bleibt, ausführliche Auswertungen mit einer grossen Benutzerpopulation vorzunehmen, um dies auch zu beweisen.

Literaturverzeichnis

[1] Ullmer, B., Ishii, H., and Glas, D., **mediaBlocks: Physical Containers, Transports, andControls for Online Media**, Proceedings of SIGGRAPH'98, ACM Press, 1998.

[2] O'Brien, J. et al. **At home with the technology**, ACM Trans. on Computer-Human Interaction, vol. 6 (3), pp. 282-308, 1999.

[3] Edwards, K. and Grinter, R. **At home with ubiquitous computing**, Proc. Ubicomp '01, pp. 256-272, Atlanta, Georgia: Springer-Verlag, 2001.

[4] Begel, A., **LogoBlocks: A Graphical Programming Language for Interacting with the World**, S.B. Thesis, MIT Department of Electrical Engineering and Computer Science, http://abegel.www.media.mit.edu/people/abegel/begelaup.pdf, 1996.

[5] Alan F. Blackwell, Rob Hague. **AutoHAN: An Architecture for Programming the Home.** Proceedings of the 2001 IEEE Symposia on Human-Centric Computing Languages and Environments, pp. 150-157, 2001

[6] A.F. Blackwell, T.R.G. Green & R.L. Hewson, **Product design to support user abstractions.** Paper submitted in June 2000 to special issue of ACM ToCHI on "The New Usability".

[7] A.F. Blackwell, **Pictorial representation and metaphor in visual language design.** Journal of Visual Languages and Computing, June 2001.

[8] A.F. Blackwell & T.R.G. Green, **Does metaphor increase visual language usability?** In Proc. IEEE Symp. On Visual Languages, pp. 246-253, 1999.