
Greifbare Programmierung

Friederike Otto
ottof@informatik.uni-muenchen.de

Universität München
Amalienstrasse 17, 80333, Deutschland

Betreuer: Paul Holleis

Zusammenfassung

Auch wenn wir täglich von immer mehr digitaler Technik umgeben sind, bleibt es doch nur wenigen Menschen vorbehalten selbst Programme zu schreiben. Würde der Akt des Programmierens aus der abstrakten Ebene der Desktop Implementierung entfernt, und in die reale Welt integriert werden, so könnte Programmierung für jeden möglich werden. Gerade Kindern bieten solche Systeme neue Lernmöglichkeiten um ihr logisches Denken zu fördern und spielend mit den Möglichkeiten der digitalen Technik vertraut zu werden. Die Anfänge der Greifbaren Programmierung gehen zurück auf die 1960er und bis heute wird an entsprechenden Systemen gearbeitet. Diese Arbeit gibt einen Überblick über die Ansätze die bisher entwickelt wurden und Systeme die tatsächlich den Weg in die Schulen gefunden haben.

**Tell me - I forget
Show me - I remember
Involve me - I will understand**

(Konfuzius)

**1. Greifbare Programme, warum und für wen?
Ein Einblick in die Logo – Philosophie.**

Programme greifbar machen, im Sinne von physisch greifbar, aber auch, und das versucht dieser Forschungszeitung zu erreichen, gedanklich greifbar, also den Prozess zur Manipulation computergesteuerter Geräte zu entmystifizieren und somit der breiten Öffentlichkeit zugänglich zu machen. So soll es zum Ziel werden den Akt des Programmierens aus dem abstrakten Umfeld von Monitor und Tastatur herauszunehmen und in das direkte Umfeld des Nutzers zu transferieren.

Den Grundstein für diese Entwicklung legte Seymour Papert [23] bereits in den 1960' er Jahren mit Gründung der Logo Philosophie [19].

In seiner Theorie des Konstruktivismus stellt er den ungleich höheren Lernerfolg dar, der sich durch „Learning by making“ einstellt. Hier sollen Kinder selbstständig ihre eigenen Ideen umsetzen können, durch den gesamten Prozess der Edit- Compile- Debug Schleife ihr Wissen festigen und jeden Fehler als Möglichkeit erkennen Zusammenhänge besser zu verstehen, denn:

“Life is not about knowing the right answer -(...)- it is about getting things to work!”
(Seymour Papert) [19]

Kinder als Zielgruppe sind nahe liegend, lernen sie schon im Kleinkindalter Wissen zu prüfen und zu festigen, indem sie Situationen aus dem Alltag nachstellen (Puppenspielen, Playmobil, Legobausteine etc.). So ist der Schritt nicht mehr weit Computer in das reale Umfeld der Kinder zu integrieren um Zusammenhänge greifbar zu machen und somit völlig neue Lernmöglichkeiten zu schaffen [20]. Doch diese Möglichkeiten verbaut man ihnen, wenn sich Computerinteraktion auf das abstrakte Level der Desktop Programmierung begrenzt.

Solche Grenzen zu sprengen ist Ziel des Forschungszeitunges der Greifbaren Programmierung. Diese Arbeit wird einige Entwicklungen aus diesem Bereich darstellen und durch Beispiele belegen, dass Kinder durchaus in der Lage sind kleine Programme zu implementieren, wenn man ihnen das richtige Werkzeug zur Hand gibt. So sollen die Kinder nicht nur den Dingen das Denken beibringen, sondern durch sie auch selbst zum Denken angeregt werden und neue Denkweisen entwickeln.

2. Bisherige Entwicklungen im Bereich der Greifbaren Programmierung

2.1 Floor Turtle – Geometrie zum Anfassen

Im Zuge seiner Konstruktivismus- Theorie versuchte Seymour Papert (Abb.1) auch die Lernmöglichkeiten an Schulen zu verbessern. So wollte er den Kindern kein abstraktes, und damit schwer zugängliches geometrisches Wissen lehren, sondern sie Geometrie erfahren lassen. Denn wie in allen Lebensbereichen wird auch die Mathematik verständlicher, wenn man sie greifbar erleben kann und so war die Turtle Geometrie geboren.



Abb.1 [6]

Die erste Floor Turtle[1] wurde Anfang der 70er am M.I.T. entwickelt, ein mechanischer Roboter, der via Kabel mit einem Computer verbunden war. Der Roboter, der bewusst auf einen einfachen Befehlssatz (Vor, Zurück, Links, Rechts) beschränkt war, konnte die am Computer eingegebenen Befehle (in der Programmiersprache Logo) verarbeiten und direkt ausführen.

Durch die direkte Reaktion der Floor Turtle (Abb. 2,3) in ihrer realen Welt und der damit verbundenen Identifikationsmöglichkeit der Kinder konnten diese schnell einfache Algorithmen zum Aufbau geometrischer Formen entwickeln. Indem sie selbst „Schildkröte spielten“ wurde es leicht nachzuvollziehen, dass sich beispielsweise ein Quadrat aus dem einfachen Algorithmus – Wiederhole viermal: gehe nach vorn und drehe dich um 90 Grad – zusammensetzt, und schon war ihr erstes Logo Programm geschrieben:

[3]

```
TO SQUARE :  
    REPEAT 4 [FORWARD: 50  
              RIGHT 90 ]  
END
```

So ließ sich der Roboter mittels der prozeduralen Programmiersprache Logo steuern, das heißt, dass die einzelnen Befehlssätze sequentiell von der virtuellen Maschine in der Floor Turtle abgearbeitet wurden.

Doch war es den Erfindern nicht wichtig den Kindern Programmierkenntnisse zu vermitteln, vielmehr war es nun möglich abstrakte Mathematik und Physik spielend in die Umwelt der Kinder zu integrieren („Toys to think with“).

Die Entwicklung der Floor Turtle:



Abb.2 [25]



Abb.3 [24]

Mit dem Siegeszug des PCs in den 80er Jahren fand auch die Floor Turtle ihren Weg in die Schulen. Obwohl bis heute vereinzelt eingesetzt, konnte sie sich auf Grund des hohen Preises, ihrer Unzuverlässigkeit und der mangelnden Möglichkeit komplexere geometrische Formen zu erschaffen, nicht weiter durchsetzen.

2.2 Screen Turtle – Ein Schritt zurück ist ein Schritt nach vorn Die Turtle auf dem Monitor

In den 80er Jahren rückte das M.I.T. von ihrer Idee der Greifbaren Programme ein Stück weit ab, und entschied sich auf Grund zahlreicher Vorteile dafür die Turtle auf den Bildschirm zu bringen [2].

Obwohl es sich jetzt nicht mehr um einen fassbaren Gegenstand handelt, soll an dieser Stelle doch kurz auf dieses Kapitel in der Entwicklungsgeschichte Greifbarer Programme eingegangen werden.

Ein Blick auf die Vorteile der Screen Turtle erklärt den ungewöhnlichen Schritt zurück auf den Monitor. So wurde der ursprüngliche Befehlssatz der Floor Turtle (ca.20) übernommen und auf über 100 Instruktionen erweitert [4], womit das implementieren komplexerer Formen möglich wurde (Abb.4). Die Analogie der Schildkröte wurde ebenfalls beibehalten und so werden Befehle immer in Blickrichtung der Turtle ausgeführt, wie der virtuelle Stift auf dem Monitor weiterhin genannt wurde.

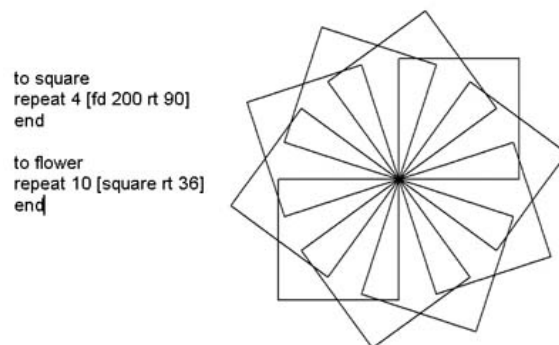


Abb.4 [26]

Neben der Entwicklungsumgebung veränderte sich auch die zugehörige Programmiersprache Logo, sie wurde toleranter gegenüber den Namen der Primitiven (Forward = For = FD = Ahead), sowie in ihrer Syntax. Die Primitive, denen Parameter mitgegeben werden mussten (z.B. Forward 100) erhielten als Einheit Turtle Steps, die je nach Auflösung des Bildschirms variierten und denen meist die kleinste, am Monitor darstellbare Distanz zu Grunde lag. Des Weiteren gehörten das Einführen von Vektoren (relative Abstände), sowie verschiedene Farben und Füllmöglichkeiten zu den Neuerungen.

Ganz offensichtlich war die Screen Turtle nutzerfreundlicher und praktikabler als ihr Vorgänger, doch der ursprüngliche Gedanke der Greifbaren Programmierung war

verloren gegangen, als sie aus der Alltagsumgebung der Kinder herausgenommen wurde.

Nicht nur die Identifikationsmöglichkeit, also das Nachspielen der Turtle war verloren gegangen, auch wurde die Screen Turtle wieder ein abstraktes Objekt für das nicht klar war, was zum Beispiel links oder rechts bedeutet (aus Sicht der Kinder, oder aus Sicht der Turtle), da sich die Interaktion auf einen zweidimensionalen, Bildschirm beschränkte.

Doch die Erfahrungen die aus diesem Projekt hervorgingen und der hohe Lernerfolg der bei den Kindern zu beobachten war, mobilisierte weitere Forschungsgruppen die Vorteile beider Turtles zu vereinigen.

2.4 Programmierbare Bausteine - Greifbare Programme erobern das Kinderzimmer

Bereits seit 1987 wurde am M.I.T. Media Laboratory an verschiedensten Entwürfen eines autonomen programmierbaren Bausteins gearbeitet

Bei ihren Entwicklungen, wie im gesamten Bereich der Greifbaren Programmierung sahen sie Kinder als Designer und Entwickler eigener Programme und wollten somit ihre Rolle nicht nur auf das manipulieren vorgegebener Systeme beschränken. Doch mit ihrer Idee der programmierbaren Bausteine (P- Bricks) wollten sie noch ein Stück weitergehen als es die bisherigen Entwicklungen erlaubten und setzten sich folgende Entwurfsziele [5]:

- Variable Einsatzmöglichkeiten:

Die Bausteine sollten eine Vielzahl von Einsatzmöglichkeiten abdecken, beispielsweise das Steuern von Robotern, das Gestalten interaktiver Räume oder auch das Durchführen physikalischer Experimente.

- Verschiedenste I/O Modalitäten:

Der P-Brick sollte unterschiedlichste Einflüsse aus der realen Welt verarbeiten können, z.B. durch Berührungs-, Licht- und Temperatursensoren, sowie Infrarotempfänger. Auch die Interaktionsmöglichkeiten mit der Außenwelt wurden durch erweiterte Ausgabemöglichkeiten wie Motoren, Licht, Sound und Infrarot verbessert. So boten sich aus den Kombinationen der Ein- und Ausgabe verschiedenste Einsatzmöglichkeiten.

- Parallelität:

Gerade bei Kindern ist es wichtig die Möglichkeit bereitzustellen, mehrere Dinge gleichzeitig zu kontrollieren, sonst verlieren sie schnell die Lust an ihrem Spielzeug. Daher sollte der P-Brick mehrere Ein- und Ausgaben gleichzeitig verarbeiten können.

- Interaktivität:

Eine weitere Anforderung an die programmierbaren Bausteine war, das sie nicht nur autonom arbeiten, sondern auch untereinander kommunizieren können sollten, um so eine Interaktion mehrerer Maschinen zu ermöglichen.

Als Ergebnis wurde unter anderem der Model 120 (Abb. 5) Programmable Brick (1994) vorgestellt, basierend auf einem Motorola Prozessor mit 32 kbyte RAM.

Der Baustein konnte vier Ausgaben und acht Eingaben gleichzeitig kontrollieren, war mit Mikrophon, Infrarot und einer kleinen Anzeige ausgestattet. Programme in LogoBlock (Die zugehörige Programmiersprache [18]) konnten auf den Brick übertragen werden, der sie dann Mittels eines eigenen Logo Interpreters ausführen konnte.

So war der P-Brick frei von störenden Kabeln, mit denen beispielsweise noch die Floor Turtle zu kämpfen hatte, da der P-Brick mit einem eigenen Mikrocomputer ausgestattet war, der in einen handtellergroßen Legobaustein eingebettet wurde. Dadurch konnte er an viele verschiedene Lego Kreaturen angesteckt werden und den Konstruktionen der Kinder damit die gewünschten Eigenschaften geben (Abb. 6).



Abb.5 Model 120 [22]



Abb.6 Cricket- Dino [5]

Die Funktionsweise des P-Bricks wurde von dem Sponsor LEGO als RCX Lego Brick (Abb. 7) im Rahmen des Lego Mindstorms Construction Kits [8] auf den Markt gebracht. Mit dem enormen kommerziellen Erfolg hatte keiner gerechnet, so wurde in den USA das Lego Mindstorms Robotic Invention Kit allein in den ersten drei Monaten 80.000mal verkauft. Das Set bestand insgesamt aus 717 Teilen, mit LegoBlocks (der zugehörigen Programmiersprache), Motoren, Rädern, verschiedenen Sensoren und den RCX Brick.



Abb.7 RXC Baustein [22]

2.5 Crickets – die mini Bausteine

1996 entwickelten Martin, Silverman und Berg am Media Lab die neue Generation der programmierbaren Bausteine, die Crickets.

Angelehnt an der Technologie der Thinking Tags [9], die zum 10 jährigen Geburtstag des Lab entwickelt wurden. Die Tags konnten als Namensschilder getragen werden, über Infrarot miteinander kommunizieren und Daten austauschen, gemeinsame oder fehlende Interessen wurden über grüne und rote LEDs angezeigt.

Schließlich stellte sich heraus, dass sich die Kern Hardware der Thinking Tags auch für die programmierbaren Bausteine anbot. Waren diese bisweilen noch so groß wie eine Milchtüte, konnte durch die Tag Technologie die Größe der CPU drastisch verringert werden.

Durch diese Technologie wurde es nun möglich die P- Bricks auf die Größe einer 9 Volt Batterie zu reduzieren, die Crickets [7]. Diese neue Generation der Programmierbaren Bausteine war wesentlich leichter, konnte Motoren kontrollieren, Sensor Daten verarbeiten und via Infrarot mit anderen Crickets und technischen

Geräten kommunizieren. Der erste Cricket der im März 1996 am M.I.T. vorgestellt wurde hatte folgende Eigenschaften:

Infrarot Schnittstelle zur Inter-Cricket Kommunikation, sowie dem Herunterladen von Programmen, jeweils zwei Anschlüsse für Ein- und Ausgabe, einen Schalter um das Cricket Programm zu starten und zu stoppen, drei LEDs und eine 9V Batterie zur Energieversorgung.

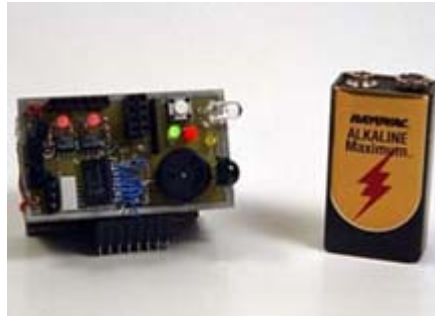


Abb.8 Classic Cricket [10]

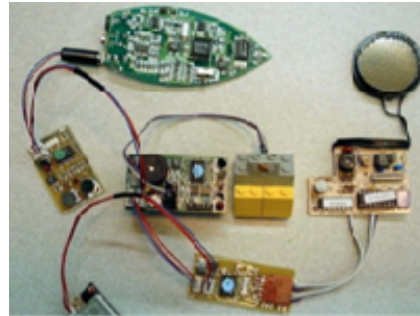


Abb.9 Cricket über einen Bus Gerät verbunden mit anderen Geräten [12]

Bei dem obigen Cricket handelt es sich um den Classic Cricket (Abb.8), um das System aber so klein wie möglich zu halten, begann man Funktionalität auszulagern und in andere Crickets zu integrieren, so entstanden weitere Cricket Arten, wie:

- Display Cricket für Sensor Input Anzeigen
- MIDI Cricket für Sound Input Verarbeitung
- Scientific Cricket für die Analog- Digital Wandlung sonstiger Sensor Eingabe

Um eine Vielzahl an Geräten in das System integrieren zu können, wurde ein eigenes Cricket Bus System entwickelt. Bis zu diesem Zeitpunkt hatte man für jede Funktion einen eigenen Cricket entwickelt, doch mit der Arbeit an dem Bus wurde klar, dass dies gar nicht nötig war. Sie begannen die Schaltkreise zu bündeln und ein eigenes Busgerät zu entwickeln, das mit den einzelnen Geräten (Sensoren, Motoren...) direkt kommunizieren konnte und über ein Protokoll die Informationen an den Cricket weiterleitete. Damit wurden weitere Cricket Versionen überflüssig, da es dem „Blue Dot“ Cricket Bus Gerät (Abb.9) möglich war mit fast jedem anderem Gerät zu kommunizieren.

Programmiert wurden die Crickets in einem Logo Dialekt, der direkt via Infrarot geladen werden konnte.

Aufgrund der geringen Größe der neuen Bausteine entstand ein vielfältiges Einsatzfeld für die Crickets, wie z.B. Roboterprogrammierung, Bewegungserfassung oder interaktive Räume [21]. Da es den Crickets auch möglich war, untereinander zu kommunizieren wurde die Simulation vom Zusammenleben mehrerer Maschinen möglich.

2.6 Das „Beyond Black Boxes“ Projekt - Greifbare Programme von Kindern entwickelt

Im Zuge der Cricket Entwicklung startete das M.I.T. das Beyond Black Boxes Projekt [13,11]. Das Projekt ging über zwei Jahre, den Augenmerk gerichtet auf die Evaluierung und Weiterentwicklung technischer Geräte die Kindern und Jugendlichen erlauben sollten ihre eigenen wissenschaftlichen Instrumente zu bauen. Die geringe Größe der Crickets ermöglichte eine Vielzahl an neuen Konstruktionen, man konnte sie beispielsweise in jeden Gebrauchsgegenstand integrieren oder am Körper tragen. Der geringe Preis (unter 30\$) erlaubte es zudem mehrere Crickets in ein System einzubauen. Im Folgenden werden einige Beispiele, die aus dem Beyond Black Boxes Projekt hervorgegangen sind präsentiert:

Das Vogelhaus:

Jenny, 11 Jahre entschied sich in dem Workshop eine Maschine zum Vogelfüttern zu entwickeln, dabei ging es ihr hauptsächlich darum eine Möglichkeit zu finden die Tiere zu beobachten und zu analysieren, auch wenn sie nicht daheim war. Das Grundkonstrukt war schnell gebaut, doch Jenny wollte ja etwas über die Vögel erfahren.

So rüstete sie ihr Vogelhaus mit einem Touchsensor aus der prüfen konnte ob ein Vogel gelandet war. Diesen Sensor verband Jenny mit einem Cricket den sie programmierte die Vögel zu zählen und der einen Lego Mechanismus in Gang setzte welcher eine Kamera auslöste, und somit ein Foto von dem Vogel machte (Abb.10). Nun war es Jenny möglich herauszufinden welche Vögel in ihrer Umgebung lebten. Um zu verhindern, dass derselbe Vogel mehrmals hintereinander fotografiert wurde baute sie in ihr Cricket Programm zusätzlich eine wait- Routine ein die dies verhinderte.



Abb.10 Jennys Vogelhaus [11]

Dancing Creatures:

Durch die Möglichkeit der Crickets miteinander zu kommunizieren, konnten auch Maschinen gebaut werden, die sich „unterhalten“ konnten. Wie die Dancing Creatures (Abb.11), die sich gegenseitig neue Tanzschritte beibrachten, indem sie über ihre Infrarot Verbindung die Programme auf andere Crickets übertragen konnten. (siehe auch 3.3 Dance Craze Buggies)

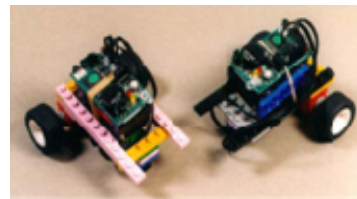


Abb.12 Dance Craze Buggies [11]

Das Nagelstudio:

Elise (11) entschied sich dafür ihr eigenes Nagelstudio (Abb. 12) zu entwerfen.

Die Funktionalität bestand aus folgendem:

- Eine Vorrichtung in die man die Hände legen konnte
- Sensoren die registrierten ob eine Hand da war
- Ein Motor der Wattebälle bewegte, die die Nägel polierten
- Eine rotierende Feder die die Nägel trocknen sollte
- Hintergrundmusik, solange die Nägel gemacht wurden
- Ein Cricket, der registrierte, wenn eine Hand den Sensor berührte und dann die programmierte Routine zum polieren der Nägel anstieß



Abb.12 Das Cricket Nagelstudio [12]

Durch die hohe Flexibilität des Crickets war ein breites Spektrum an Einsatzmöglichkeiten gegeben, das den Kindern die Möglichkeit gab ihre eigenen Ideen frei zu entwickeln und damit aus der Rolle des reinen Manipulators eines Systems heraus zu brechen. Die kleinen programmierbaren Bausteine konnten nach Belieben in nahezu alle Alltagsgegenstände integriert werden. So wurde mit den Crickets eine Komponente entwickelt, welche die Greifbare Programmierung ein großes Stück nach vorne brachte, da sie durch den hohen Grad der Manipulationsfreiheit des Nutzers die Beziehung zwischen ihm und dem Gerät intensivierte.

Das Beyond Black Boxes (BBB) Projekt hat gezeigt das Kinder aktiv wissenschaftlich arbeiten können mit Hilfe von programmierbaren Bausteinen. Auch wenn die Teilnehmer den technischen Teil nur bedingt verstehen konnten, so haben sie doch schnell gelernt mit diesem kleinen Gerät den statischen Objekten in der realen Welt Verhalten beizubringen.

So bringt der Touchsensor im Vogelhaus alleine noch nichts, aber mit Hilfe des Sensors in Verbindung mit dem Cricket konnte Jenny ein Programm schreiben, das dem Boden Verhalten zuordnete.

Während das BBB Team immer mehr und kleinere intelligente Devices baute, stellte sich schnell heraus wie wichtig die Kommunikation der Geräte untereinander ist damit sie nicht zu technischen Inseln verkommen. So wird als neues Designziel festgelegt, das die Crickets mit möglichst vielen Geräten aus dem Alltag der Kinder (Kameras, Gameboy, Stereoanlage u.v.m.) agieren sollen, damit sie als Teil eines Gesamtsystems gesehen werden, und nicht als isoliertes Spezialgerät.

3. Tangible Programming Brick System – Programmieren nach Baukastenprinzip

3.1. Idee des Systems

Ende der Neunziger entwickelte Timothy McNerney in Anlehnung an die Cricket Architektur das Tangible Programming Brick System[14].

Idee des Systems war einen ganzen Satz Legosteine ähnlicher programmierbarer Bausteine zu realisieren, mit denen die Kinder herkömmlich spielen können und dabei eigenständig Programme entwickeln. Auch McNerney wollte greifbare direkte Programmierung in das reale Umfeld der Kinder bringen:

„My primary motivation is to make programming an activity that is accessible to the hands and minds of younger children by making it more direct and less abstract.” [14]

Während die Crickets über einen Bus mit anderen Komponenten kommunizierten, waren die Tangible Programming Bricks (TPB) mit einer Steckverbindung ausgestattet (Abb.13), wobei das Programm sequentiell von unten nach oben abgearbeitet wurde.

Eine weitere Neuerung fand sich in McNerneys Steckkarten Prinzip (Abb.14), das es ermöglichte den TPB Parameter mitzugeben.



Abb.13 TPB Stapel [14]



Abb.14 TPB mit Steckkarten [14]

Neben der Intention abstrakte Programmierung auf ein kinderfreundliches Level zu bringen sah McNerney auch den großen Vorteil seines Systems darin, dass es mehreren Personen gleichzeitig möglich war an einem Programm zu arbeiten, was nicht nur die Qualität des Programms verbesserte, sondern auch den Lerneffekt innerhalb der Gruppe.

„I imagined a new generation of young programmers building programs together with their hands instead of one person typing on a keyboard and others looking over their shoulder.” (McNerney) [14]

McNerneys Ziel war es durch die TPB das Programmieren zu einer Handlung zu machen die man anfassen kann, und durch diese Wandlung der Programmierung zu einem Spielzeug, den Kindern fundamentale Konzepte des digitalen Zeitalters in einer spielerischen Umgebung näher zu bringen.

3.2 Die Entwicklung

Der folgende Abschnitt soll darstellen welche Entwurfsentscheidungen bei der Entwicklung des Tangible Programming Brick Systems zu treffen waren.

[15] McNerneys erster Prototyp (Abb. 15) basierte auf einem 4x2 Legostein. Er enthielt einen Mikrocomputer, eine LED und besaß die Möglichkeit mit seinen direkten Nachbarn nach oben oder unten zu kommunizieren. Doch bald stellte sich heraus, dass die angeforderte Funktionalität in dem kleinen Legostein nicht umsetzbar war, so besaß der Tangible Programming Brick noch keine ausgereifte Kommunikation und keine Eingabemöglichkeiten für Parameter.

Um das System ausbauen zu können stieg McNerney auf den größeren 6x2 Legostein um. Das 6x2 Model (Abb.16) bestand aus einem Mikrocomputer, weiblichen und männlichen Steckverbindungen auf Ober- und Unterseite des Bausteins, wobei die männlichen mit acht vergoldeten Kontakten ausgestattet waren für die Energieversorgung, die serielle Kommunikation und einer globalen run / stopp Signalweiterleitung.

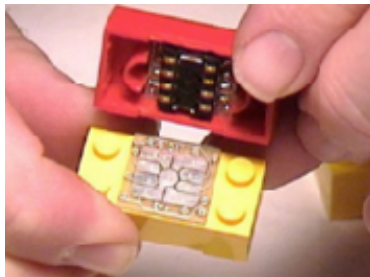


Abb.15 Der 4x2 TPB Prototyp [17]



Abb.16 Der 6x2 TPB [14]

Um den Einsatz der TPB nicht einzuschränken musste bedacht werden, dass die Bricks auch um 180 Grad verdreht (vorne auf hinten) gesteckt funktionsfähig sein sollten, so wurden die Wege für Kommunikation und Energieübertragung doppelt auf einen Baustein angebracht.

Als neue Manipulationsmöglichkeit greifbarer Programme führte McNerney drei Arten von Steckkarten ein, die auf die Vorderseite jedes TPB (Abb.17) passten [14]:

- IR/ Bus Card:
Diese Steckkarte enthielt jeweils einen infrarot Sender und Empfänger, sowie zwei Cricket Bus Konnektoren, durch die der Brick in Cricketsysteme eingebunden werden konnte. Die Infrarotverbindung war bestimmt für das Herunterladen von Programmen und der Kommunikation zu anderen Tangible Programming Brick Systemen.
- EEPROM Card:
Dient zur persistenten Speicherung von Informationen, sowie dem Bereitstellen von Parameterwerten.

- Display Card:
Diese Karte ist in der Lage jegliche alphanumerische Information für den Nutzer anzuzeigen, z.B. Parameternamen, Sensorwerte.

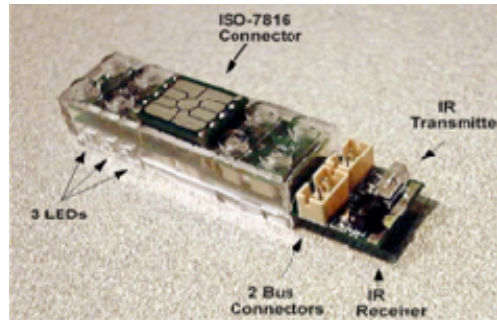


Abb.17
Ein TPB mit einer IR/ Bus Karte [14]

Die Karten erhöhten die Flexibilität der TPB und erlaubten es mehr Funktionalität in das System zu bringen ohne größeren Platz zu beanspruchen.

3.3 Beispiel für greifbare Programmierung mit TPB

Nach der Theorie soll nun ein praktisches Beispiel zum weiteren Verständnis dieses Systems beitragen:

Dance Craze Buggies:

1998 entwickelten Fred Martin und Rick Borovoy die Anwendung Dance Craze Buggies (Abb.18) für die Crickets, hier konnten Spielzeugautos Tanzschritte beigebracht und unter den Autos weitergegeben werden. Borovoy schrieb über die Idee:

„Imagine a child teaching her robotic toy a new dance step. Then, when she is playing with a friend, her toy can „teach“ this new dance to her friend`s toy. Later, her friend can modify this dance a little, and pass it to another friend. The creator of the dance can check the Net to see how far her dance has spread.“ [14]

McNerney sah seine Bricks als Möglichkeit die Idee der Dance Craze Buggies zu komplettieren, denn in Borovoys und Martins Szenario fehlte eine geeignete Möglichkeit den Autos neue Tanzschritte beizubringen, ohne dabei am PC programmieren zu müssen. Zur Lösung dieses Problems entwickelte McNerney ein „Lehrer Hilfsmittel“, genannt den Phaser, der mit einem Stapel TPB gefüttert werden konnte[14].

Jedem Brick wurde ein Tanzschritt zugeordnet (vor, zurück, kleiner Schritt vor...), und eine Parameter Karte, die die Anzahl an Wiederholungen festlegte. Dadurch wurde es den Kindern möglich ihre Spielzeugautos ganz ohne PC zu programmieren, indem sie ein entsprechendes Programm mit den Tangible Programming Bricks zusammensteckten und dann den Phaser damit fütterten. Wenn der Phaser aktiviert

wurde, wurde das Programm auf das entsprechende Auto geladen, es hatte den Tanz gelernt.

Diese Idee präsentierte McNerney auf der „Toys of Tomorrow“ Ausstellung 1999. Der Erfolg auf der Ausstellung hat die Idee der Greifbaren Programmierung untermauert, denn nun sind das Programm und das zugehörige Gerät nicht nur logisch verbunden, sondern auch visuell gekoppelt, was es Kindern wesentlich erleichtert das System zu verstehen, im Gegensatz zu einer unsichtbaren Software die vom Computer aus ein Gerät steuert.



Abb.18

Dance Craze Buggies, die über Crickets kommunizieren können, und über den TPB gefütterten Phaser programmiert werden [16]

Im Moment existieren bereits 50 verschieden Tangible Programming Bricks, allerdings wird wohl erst eine Kostenreduzierung der Bausteine dieses System zur Marktreife bringen.

3.4 Wie geht es weiter mit den Tangible Programming Bricks

Die erste Generation der TPB ist entwickelt, und hat ein großes Stück dazu beigetragen die Programmierung greifbar und damit verständlicher zu machen.

Doch auch dieses System steckt noch in den Kinderschuhen, hier noch ein paar Kritikpunkte die McNerney selbst aufgezeigt hat und in Zukunft bearbeiten will [14].

Die TPBs sind begrenzt auf einen linearen Verlauf:

Das jetzige System erlaubt es nur die Bausteine eindimensional zusammenzufügen. Die Implementierung eines Bausteins, der die Verbindung zu Nachbarn zwei- oder dreidimensional erlaubt wäre ein weiterer Entwicklungsschritt. Auch eine „Brücken“ Karte würde das Zusammenfügen in eine zweite Dimension erlauben. Diese Karte könnte als Verbindungsstück zwischen den Dimensionen agieren, womit komplexere Programmabläufe denkbar wären. Was allerdings auch zu einer schwereren Bedienbarkeit dieses Systems führen würde.

Kein visueller Unterschied der TPBs:

Formen können eine große semantische Hilfestellung bieten, gerade bei Kindern fällt das Differenzieren von Objekten leichter durch Formen als durch Labels.

Wie bei einem Puzzle wäre es auch denkbar, dass nur bestimmte Bricks zusammenpassen, was eine zusätzliche Implementationshilfe leisten würde.

Reduzierung der Kosten:

Wie bereits erwähnt ist das System im jetzigen Stadium zu teuer für eine Kommerzialisierung, auch wenn McNerney einen großen Kostenfaktor damit ausgeschlossen hat indem er nicht jedem Brick seine eigene Energieversorgung mitgab, so bleibt doch das Verbindungssystem und der Mikroprozessor ein erheblicher Kostenfaktor, wobei sich wahrscheinlich letzteres Problem mit der Zeit von selbst löst.

4. Materielles Programmieren, nur eine Spielerei?

Es gibt weit mehr Entwicklungen in dem Bereich der Greifbaren Programmierung, als ich an dieser Stelle vorstellen kann. Doch ich denke durch das Herausgreifen einiger repräsentativer Beispiele konnte gezeigt werden, dass sich viel bewegt in diesem Gebiet, angefangen vor 40 Jahren mit der noch etwas behäbigen Floor Turtle bis zum Ende des 19. Jahrhunderts, in dem es schon möglich ist, den gesamten Rechenprozess in einen Legostein zu packen. Man kann also gespannt sein wohin sich dieser Zweig der Informatik entwickelt.

Doch abgesehen von dem technischen Fortschritt der hier zu ersehen ist, wird auch immer mehr ein Umdenken in der Informatik erkennbar, hin zu mehr Nutzerfreundlichkeit. Das Beispiel der Greifbaren Programmierung zeigt, welche Wege man nehmen kann, um den Menschen den Umgang mit Bits und Bytes zu erleichtern. Manchmal muss man versuchen völlig neue Wege einzuschlagen und alte Denkweisen gänzlich hinter sich lassen, wie hier geschehen indem man nach neuen Interaktionsmöglichkeiten sucht, abseits der Desktop Programmierung. Auch sollte man dieses Thema interdisziplinär betrachten, so nutzt die Technologie alleine nicht viel, es muss auch tief greifend Überdacht werden, was Kinder lernen können und sollen.

Doch es wird auch Kritik laut, die davor warnt, Greifbare Programmierung schon so früh in das Leben unserer Kinder zu bringen wo wir doch jetzt schon immer abhängiger werden von Technologien die die meisten nicht verstehen. Können dann Kinder solche Zusammenhänge überhaupt begreifen, werden sie nicht schon zu früh überfordert? Andererseits ist es enorm wichtig, dass die künftigen Generationen mit dieser Technik vertraut werden, lernen sie zu kontrollieren und effektiv zu nutzen um unsere Zukunft gestalten zu können.

Solange die Kinder spielerisch an das Thema herangeführt werden, und wie in vielen Beispielen gesehen mit großem Einsatz an Greifbaren Programmen arbeiten, denke ich, dass sie der künftigen Generationen die Möglichkeit bieten durch neue Lehr- und Lernmethoden, natürlich digitale Technik in ihren Alltag zu integrieren.

5. Quellenangabe

1. From Turtles to Tangible Programming Bricks: explorations in physical language Design, Timothy S. McNerney. Per. Ubiquit. Comput. (2004) 8: 326- 337; <http://portal.acm.org/citation.cfm?id=1023813.1023817>
2. Computer Science Logo Style, vol.1, Brian Harvey, Symbolic Computing 2/e Copyright (C) 1997 MIT; <http://www.cs.berkeley.edu/~bh/v1ch10/turtle.html>
3. <http://el.media.mit.edu/logo-foundation/logo/turtle.html>
4. http://www.staff.ucsm.ac.uk/rpotter/ict/logo-control/Screen_Turtle_Notes.pdf
5. Programmable Bricks: Toys to think with, M.Resnick, F.Martin, R.Sargant, B.Silverman, IBM Systems Journal, vol.35, nos.3&4, pp.443-452; <http://researchweb.watson.ibm.com/journal/sj/mit/sectionc/martin.html>
6. <http://web.media.mit.edu/~papert/> (Foto: Seymour Papert)
7. The Children´s Machines: Handheld and Wearable Computers Too, B.Mikhak, F.Martin, M.Resnick, R.Berg, B.Silverman, Proceedings of the International Symposium on Handheld Ubiquitous Computing, Karlsruhe, Germany; <http://www.wellesley.edu/Physics/Rberg/papers/HUC99.pdf>
8. Technologies for Lifelong Kindergarten, M.Resnick, Published in Educational Technology Research & Development, vol. 46, no. 4 (1998); <http://web.media.mit.edu/~mres/papers/lifelongk/lifelongk.pdf>
9. <http://web.mit.edu/6.933/www/Fall2000/LegoMindstorms.pdf>
10. <http://llk.media.mit.edu/projects/cricket/about/techtalk/reddot.html> (Foto: Cricket)
11. Beyond Black Boxes: Bringing Transparency and Aesthetics Back to Scientific Investigation, Mitchel Resnick, Journal of the Learning Sciences, vol.9, no1, pp.7-30; <http://llk.media.mit.edu/papers/archive/bbb/>
12. To Mindstorms and Beyond: Evolution of a Construction Kit for Magical Machines, B.Mikhak, F.Martin, M.Resnick, R.Berg, B.Silverman; Morgan Kaufman/ Academic Press, San Francisco, CA; <http://www.wellesley.edu/Physics/Rberg/papers/magical-machines.pdf>
13. <http://llk.media.mit.edu/projects/bbb/>
14. Tangible Programming Bricks: An approach to making programming accessible to everyone, Timothy Scott McNerney, S.M.Thesis <http://xenia.media.mit.edu/~mcnerney/mcnerney-sm-thesis.pdf>
15. Tangible Computation Bricks – Building-blocks for physical Microworlds, Timothy S. McNerney. <http://xenia.media.mit.edu/~mcnerney/tangible-comp-bricks-review2.2.pdf>
16. Meta-Cricket: A designer´s kit for making computational devices, F.Martin, M.Resnick, R.Berg, B.Silverman, IBM Systems Journal, vol.39, nos. 3&4, pp.795-815; <http://www.research.ibm.com/journal/sj/393/part2/martin.pdf>
17. <http://xenia.media.mit.edu/~mcnerney/lego/> (Foto: Der 4x2 TPB Prototyp)
18. LogoBlocks – a graphical programming language for interacting with the world, A.Begel, S.B. Thesis; <http://web.media.mit.edu/~abegel/begelaup.pdf>
19. Logo Philosophy and Implementation, Seymour Papert, Logo Computersystems Inc. 1999; <http://www.microworlds.com/company/philosophy.pdf>
20. Lego/Logo: Learning Through and About Design, M. Resnick, S. Ocko, Norwood , NJ: Ablex Publishing <http://llk.media.mit.edu/papers/archive/ll.html>

21. Physical Programming: Designing Tools for Children to Create Physical Interactive Environments, J. Montemayor, A. Druin, A. Farber, S. Simms, W. Churaman, A. Dàmour; <ftp://ftp.cs.umd.edu/pub/hcil/Reports-Abstracts-Bibliography/2001-21html/2001-21.pdf>
22. <http://el.media.mit.edu/logo-foundation/pubs/logoupdate/v7n1/v7n1-pbrick.html>
(Foto: Model 120 Programmable Brick, RXC Baustein)
23. www.papert.org. (Webseite: Seymour Papert, Gründer der Logo- Philosophie)
24. <http://www.st-bartholomews.leics.sch.uk/subjects/ict.html> (Foto: Floor Turtle)
25. <http://pocket.free.fr/html/vm2003/51.jpg> (Foto: Floor Turtle)
26. http://www.bridgendgfl.net/Teachers/digital_portfolio/Images/Year%204/U4Eb.jpg
(Foto: Screen Turtle)

