Rethinking Programming

“Environment”

Technical and Social Environment Design toward Convivial Computing

Convivial Computing Salon, May 5, 4-5pm London (Zoom)

Jun Kato\(^*1\), Keisuke Shimakage\(^*2\)

\(^*1\) National Institute of Advanced Industrial Science and Technology (AIST)

\(^*2\) OTON GLASS, Inc.
Rethinking programming “environment”

BACKGROUND

Programming environment design for development of programs that run in the real world

TOWARD CONVIVIAL COMPUTING

• Technical environment design for collaborations
• Social environment design for more inclusion

CONCLUSION
A bit of self introduction...

Phybots
A toolkit (API and runtime debugger) for making "robotic things"
ACM DIS 2012

DejaVu
IDE extensions for developing Interactive camera-based programs
ACM UIST 2012

Songle Sync
APIs for controlling various devices in synchronization with music
ACM Multimedia 2018
Rethinking Programming "Environments"
Rethinking Programming "Environments"
In our demonstration experiment, over 110 devices were connected to the Songle Sync platform.
Integrated Development Environment (IDE)

- Code editor
- Programming language
- Toolkit (API)
- Domain-specific language
- Visual programming
- Programming language
- IDE extensions
- Debugger

Rethinking Programming "Environments"
A bit of self introduction...

Phybots
A toolkit (API and runtime debugger) for making “robotic things”
ACM DIS 2012

DejaVu
IDE extensions for developing interactive camera-based programs
ACM UIST 2012

Songle Sync
APIs for controlling various devices in synchronization with music
ACM Multimedia 2018

PX for programs running in the real-world
**f3.js: A Parametric Design Tool for Physical Computing Devices for Both Interaction Designers and End-users**

f3.js is a tool for designing physical computing devices. It allows users to create designs using JavaScript and then generate code for specific hardware platforms. The platform supports both interaction designers and end-users.

[https://f3js.org](https://f3js.org)

*Jun Kato, Masataka Goto*
Personal fabrication made easy

3D printers

Photo taken by Atsushi Tadokoro (CC BY 2.0)
https://www.flickr.com/photos/tadokoro/5138646645

Laser cutters

Sensor and actuator modules

How about programming and device assembling?
Different tools and expertise needed for software and hardware design

- Programmers need to imagine hardware while writing code
- "new Button()" does not infer any hardware layout
f3.js

Module repository for hardware metrics

APIs for enclosure layout

Live programming editor for microcontroller firmware and enclosure layout

IDE for creating laser-cut interfaces and microcontroller programs from single code base
Provide the source code of a microcontroller or tiny computer in JavaScript. Node.js-based computers are supported. Require f3.js package and use its API to design the device enclosure.

```javascript
var WebSocket = require('ws');
var serverAddr = 'ws://192.168.10.100:8080/entry'; // Server URL
var ws = new WebSocket(serverAddr);
var upmBuzzer = require('upm_buzzer');

var numBuzzers = 3; // Number of buzzers [1,5]
var volume = 57; // Volume [0,100]
var inputs = [3, 5, 6, 9];
var buzzers = [];
var r = 80;

var f3js = require('f3js');
var c = f3js.createContainer();
c.x = 50;
c.y = 50;

var c1 = 43,
c2 = 36,
x = 70,
y = 140,
width = 70;
var a = c.createPath();
var ps = a.extrude(60);
ps[0].y = 50;
ps[0].x = 270;
f3js.add(ps[0]);

var ox = [ 65, 145, 65, -15 ]
, dy = [ -15, 65, 145, 65 ];
function addPetol(layout, module) {
    var cc = f3s.createContainer();
    // Code to add Petol
}
```
Integrated Development Environment (IDE)

Programing environments are designed for programmers
Rethinking programming “environment”

BACKGROUND
Programming experience design for development of programs that run in the real world

TOWARD CONVIVIAL COMPUTING
• Technical environment design for collaborations
• Social environment design for more inclusion

CONCLUSION
Programming environments are (usually) different from runtime environments

- **Programmers** develop programs and publish them
- **Users** install “apps” and use them
- Once published, the programs cannot be edited
Some environments allow “remix”

Scratch – Imagine, Program, Share. https://scratch.mit.edu/

“See inside”
to read and edit source code

“Remixes”
to play with edited programs
Some environments allow “remix”

Still, programming environments are designed for programmers

“Remixes” to play with edited programs
What if we design it for “both”

- Instead of compiled programs, source code is shared
- Furthermore, users benefit from a respective UI design
- Users can **safely edit the programs without breaking them**
Live Tuning:
Users can get a customized variation of the device.
User-Generated Variables:
Users can propose parameters for the device spec
A multi-layered UI design approach

- Base layer for programmers
- Another layer for users


Meta-design framework

- Programmers as meta-designers
- Users as designers
- Programming environments become socio-technical systems

Fischer et al. Revisiting and broadening the meta-design framework for end-user development. 2017.
Rethinking programming “environment”

BACKGROUND

Programming experience design for development of programs that run in the real world

TOWARD CONVIVIAL COMPUTING

• Technical environment design for collaborations
• Social environment design for more inclusion

CONCLUSION
A programming environment is usually designed to consist of (merely) computational artifact
A merely technical approach is “scalable” but sometimes in short of “social inclusion”

- Computational support X is available for many people but for some people with characteristics Y
- For instance, f3.js can be used by a variety of people but is not designed for people with visual impairment
The above studies emphasise how rapid prototyping tools hold great potential for producing individualised, and affordable, AT. However, despite use of the DIY acronym, people with disabilities have been framed as primarily users or consumers, rather than producers, of DIY-ATs. For example, Hook et al. defined DIY-AT generally as ‘the development of AT by non-professionals’ [29:598]—referring to parents, friends and care-givers. Buehler et al. [14] noted that it is rarely the people with disabilities themselves who create and share DIY-AT designs online. While DIY is not an unfamiliar concept in AT, it seems to be affiliated people with a technical background who are part of the extended care network around a person with a disability. There are
A smart glasses built with a 3D printer and Raspberry Pi to help the father of one of the authors (Keisuke) who acquired dyslexia.
OTON GLASS as a toolkit
People with visual impairments teamed up with evangelists to develop smart glasses
A Paralympian with visual impairment and his professional supporter proposed to add the voice chat feature and implemented it with help of the evangelists.
An evangelist as part of a programming environment

- Social inclusion is a hybrid of technical and social implementations
- Programming environment design can be community design
- Programming as communication between diverse kinds of people
Summary of our social environment design

Programming by a community of people

• A programming environment consists of not only computational artifacts but also a community of people to collaborate

• “People are message-passing objects” [Salon 2020 Day 1]

Programming as communication

• Unlike tailor-made model, “the programmer” with visual impairment produces ideas and decides what to build

• The programming activity inherently involves communication and enables empowerment
Rethinking programming “environment”

BACKGROUND
Programming experience design for development of programs that run in the real world

TOWARD CONVIVIAL COMPUTING
• Technical environment design for collaborations
• Social environment design for more inclusion

CONCLUSION
Programming Experience (PX) research for convivial computing

Programming environments are usually designed exclusively for programmers but should be more inclusive!

- They can be shared among programmers and users
- They can be hybrid of people and computational artifacts