Direct Manipulation Programming in Sketch-n-Sketch: Three Years, Three Ideas

Ravi Chugh
Direct Manipulation Programming in Sketch-n-Sketch: Three Years, Three Ideas

Ravi Chugh
I ❤️ GUIs
Direct Manipulation in Sketch-n-Slash: Three Years, Three Ideas

Ravi Chug

The University of Chicago

I 💔 GUIs
I 💚 PLs

I 💔 GUIs
type LogoParams
  = TopLeft {x:Num, y:Num, size:Num}
  | Center {cx:Num, cy:Num, rad:Num}

logo : String -> LogoParams -> Svg
logo fill logoParams =
  let
  {x, y, size} =
    case logoParams of
    TopLeft data ->
      data
    Center {cx, cy, rad} ->
      {x=cx-rad, y=cy-rad, size=2*rad}
  in
  [ rect fill x y size size
    , line "white" 10 x y (x + size) (y + size)
    , line "white" 10 x (y + size) cx cy
  ]

main =
svg <- logo "gray" <- TopLeft {x=100, y=130, size=200}
Sketch-n-Sketch

Direct Code Manipulation

Direct Output Manipulation
Direct Code Manipulation

Bidirectional Programming
[PLDI 2016, OOPSLA 2018]

Output-Directed Programming
[UIST 2016, work-in-progress]

Structured Text Editing
[ICSE 2018, work-in-progress]

Direct Output Manipulation

Sketch-n-Sketch
Bidirectional Programming  
[PLDI 2016, OOPSLA 2018]

Output-Directed Programming  
[UIST 2016, work-in-progress]

Structured Text Editing  
[ICSE 2018, work-in-progress]
Edit Program
Edit Program; Run
Edit Program; Run; View Output
( Edit Program; Run; View Output )+
(Edit Program; Run; View Output; Edit Output) + Synthesize Program Repair
Bidirectional Programming

\[ f(x) \Rightarrow y \quad \text{"get"} \]

\[ f(x^I) \Leftrightarrow y^I \quad \text{"put"} \]
1. Function $f$ in DSL or restricted style (i.e. point-free)

```
f x ⇒ y
```

```
f x' ⇝ "get"
```

```
"put"
```

```
f x'1 ⇐ y'1
```

Bidirectional Programming
1. Function f in DSL or restricted style (i.e. point-free)

2. Update "data" \(x\) but not "code" \(f\)
1. Function $f$ in DSL or restricted style (i.e. point-free)

2. Update "data" ($x$) but not "code" ($f$)

3. Restricted structural changes between $y$ and $y'$
\[ f(x) \Rightarrow y \]
\[ f(x^l) \Leftarrow y^l \]
Bidirectional Evaluation

\[ f(x) \Rightarrow y \]

\[ f(x') \Leftrightarrow y' \]
Bidirectional Evaluation

All expressions (code + data) can be run backwards and updated.
Bidirectional Evaluation

All expressions (code + data) can be run backwards and updated.

"Small" structural changes to values + user-defined lenses for customization.
Bidirectional Evaluation

\[ e \Rightarrow v \]

\[ e^l \leftrightarrow v^l \]
Bidirectional Evaluation

Elm-like PL + HTML GUI

Demo
Bidirectional Evaluation
Future Work

More explanation / interaction when ambiguity

Synthesis of larger structural repairs

Distributed editing for web development

Other source languages and target domains
Bidirectional Programming
[PLDI 2016, OOPSLA 2018]

Mikaël Mayer
Wed 3:30pm @ OOPSLA
Bidirectional Programming
[PLDI 2016, OOPSLA 2018]

Output-Directed Programming
[UIST 2016, work-in-progress]

Structured Text Editing
[ICSE 2018, work-in-progress]
Draw
Relate
Group
Demo
Programming in 2015

let rect1 = ...
let line2 = ...
let line3 = ...
let rect1_x = ...
let rect1_y = ...
let rect1 = ...
let line2 = ...
let line3 = ...

Programming in 2015

Sketch-n-Sketch
Programming in 2015

```
let group x y size =
  let rect1 = ...  
  let line2 = ...  
  let line3 = ... 

  group 0 0 999999
```
Programming in 2015

let group x y size =

  let rect1 = ...
  let line2 = ...
  let line3 = ...

  group 0 0 999999
Programming in 2015

let group x y size =

  let rect1 = ...
  let line2 = ...
  let line3 = ...

group 0 0 111111
"Raw" Shape

(x1,y1)

(x2,y2)

(x3,y3)
Draw
Relate
Group
Tweak

(x1, y1)

(x1+2, y1+8)

(x1+10, y1+10)

"Anchored" Shape
Draw
Relate
Group
Tweak

"Bounded" Shape

(x1, y1)

(x1 + 0.20*(x2-x1), y1 + 0.80*(y2-y1))

(x2, y2)
Representation affects grouping operations.

(def rawPoly
  (let [x1 y1] ...
    (let [x2 y2] ...
      (let [x3 y3] ...
        ... ))))

(def anchoredPoly
  (let [x1 y1] ...
    (let [dx2 dy2] ...
      (let [dx3 dy3] ...
        ... ))))

(def boundedPoly
  (let [x1 y1] ...
    (let [x2 y2] ...
      (let [x3 y3] ...
        (let [px3 py3] ...
          ... ))))

↓

↓

↓
"Anchored" Group

"Bounded" Group
"Anchored" Group

"Bounded" Group
"Anchored" Group

"Bounded" Group
"Anchored" Group

"Bounded" Group
Output-Directed Programming

Future Work

More control over generated code

More relationship / abstraction patterns

Visualization / manipulation of intermediates

DSLs for defining new transformations

General-purpose output-directed transforms
Output-Directed Programming

[UIST 2016, work-in-progress]
Bidirectional Programming
[PLDI 2016, OOPSLA 2018]

Output-Directed Programming
[UIST 2016, work-in-progress]

Structured Text Editing
[ICSE 2018, work-in-progress]
Demo

Structure Editors

Structured Text Editing

Show AST on Code

Refactoring Tools

Text Editors
(func arg)

Show AST on Code
Show AST on Code
Show AST on Code
Show AST on Code
func arg

Show AST on Code
Structured Text Editing versus "Traditional" Refactoring

21 users
Traditional ("Text-Select") Mode

Text Select

(def image1
    (let [width height]
        (let [x y] [100 100]
            (image "lightgrey"))

Right-Click Menu

Select Arguments

Defaults

Deuce ("Box-Select") Mode

Structure Select

(def image1
    (let [width height]
        (let [x y] [100 100]
            (image "lightgrey"))

Short Menu

Code Tools
- Move Definition
- Duplicate Definition

Defaults

- Move width and height

Move Definition

Requirements
- Select one or more variable definitions and one target position (i.e. whitespace) (Satisfied)

Code Updates
- Move width and height
1. Tutorial

2. **Head-to-Head** Tasks (2x each; once per mode)

3. **Mix-and-Match** Tasks (free to use both modes)

4. Exit Survey
Deuce more effective than Traditional?

Deuce doesn't help discoverability
Deuce more effective than Traditional?

Deuce may be faster once learned
Deuce preferred to Traditional?

Survey

- #Responses
- 

- #Participants

- #Responses

Almost everyone used Deuce more

Modest subjective preference for Deuce
**Mix-and-Match Tool Usage**

- Rename
- Make Equal with Single Variable
- Introduce Variable(s)
- Add Argument(s)
- Create Function from Arguments
- Move Definition(s)
- Inline Definition(s)
- Create Function by Merging Definitions
- Create Function from Definition

**Hypothesis:** Deuce better for multi-arg transforms
Deuce versus Traditional

Traditional may be better for learning

Deuce may be faster once learned

Deuce strongly preferred
Structured Text Editing

Future Work

UI concerns for larger programs

How to encourage structured transforms?

DSLs for defining new transformations

Real languages in real editors
Structured Text Editing
[ICSE 2018, work-in-progress]
Bidirectional Programming
[PLDI 2016, OOPSLA 2018]

Output-Directed Programming
[UIST 2016, work-in-progress]

Structured Text Editing
[ICSE 2018, work-in-progress]
Semantic Foundations for Hole-Driven Development

[Omar et al. POPL 2017, POPL 2019, wip]
Semantic Foundations for Hole-Driven Development

[Omar et al. POPL 2017, POPL 2019, wip]

Cyrus Omar
Tue 2:00pm @ LIVE
Thanks!

Nick Collins  Brian Hempel  Grace Lu  Justin Lubin  Mikaël Mayer

MITCH SNSF

Swiss National Science Foundation

Jacob Albers
Bidirectional Programming

Wed 3:30pm @ OOPSLA

Mikaël Mayer

Output-Directed Programming

Tue 4:00pm @ LIVE

Cyrus Omar

Structured Text Editing

Tue 2:00pm @ LIVE

Hazel