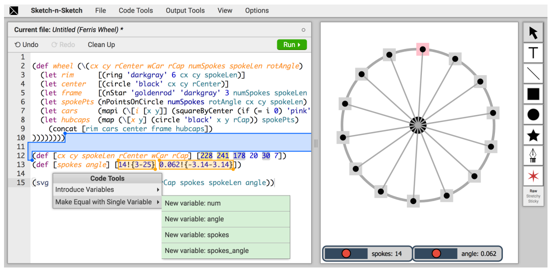
Ravi Chugh — Direct Manipulation Programming Systems   Research Statement (August 2017)

Programming is a powerful and increasingly important skill in many disciplines, but the workflow that programming languages demand—coding in a text editor, running the program, viewing the output, and then repeating—both (a) limits the creativity and pace of expert programmers and (b) shuts out millions more users that, though not programmers, nonetheless create a variety of complex digital objects using graphical user interface (GUI) based software. On the other hand, GUIs are intuitive and interactive, but they often require repetitive tasks to be performed manually, again, limiting the pace of meaningful work. This tension between programming languages and GUI systems is ubiquitous, appearing in domains such as word processing (*e.g.* LaTeX vs. Microsoft Word), data manipulation and visualization (*e.g.* D3 or Python vs. Microsoft Excel), web development (*e.g.* JavaScript vs. Adobe Dreamweaver), and graphic design (*e.g.* Processing vs. Adobe Illustrator).



```

(def wheel (\(cx cy rCenter wCar rCap numSpokes spokeLen rotAngle)
  (let rim      [(ring 'darkgray' 6 cx cy spokeLen)]
  (let center   [(circle 'black' cx cy rCenter)]
  (let frame    [(nStar 'goldenrod' 'darkgray' 3 numSpokes spokeLen
  (let spokePts (nPointsOnCircle numSpokes rotAngle cx cy spokeLen)
  (let cars     (mapi (\[i [x y]] (squareByCenter (if (= i 0) 'pink'
  (let hubcaps  (map (\[x y] (circle 'black' x y rCap)) spokePts)
    (concat [rim cars center frame hubcaps])
))))))))

(def [cx cy spokeLen rCenter wCar rCap] [228 241 178 20 30 7])
(def [spokes angle] [14!{3-25} 0.062!{-3.14-3.14}])

(svg                               Cap spokes spokeLen angle))
```

`http://ravichugh.github.io/sketch-n-sketch/`

The primary goal of my research is to develop new interactive capabilities for writing programs, to bridge the gap between programming languages and graphical user interfaces—a combination I call *direct manipulation programming systems*—in order to make expert programmers more productive and to make programming more accessible to novices. A motto for this work is to enable "Programming with Less Keyboard, More Mouse." In pursuit of this goal, my research comprises work at the boundaries of **programming languages**, **software engineering**, and **human-computer interaction**.

Direct Manipulation of Code and Output in SKETCH-N-SKETCH. So far, my research group and I have been investigating direct manipulation programming techniques in an open-source, browser-based system, called SKETCH-N-SKETCH, for creating Scalable Vector Graphics (SVG)—such graphics are an important building block for many other application domains. In addition to supporting traditional, text-based programming, SKETCH-N-SKETCH allows the code (*i.e.* source text) and output of a program to be directly manipulated with the mouse, using the left and right halves, respectively, of the editor (shown above).

- (Prototyping Phase) Early in the design process, it is natural to experiment with shapes directly in the canvas. Therefore, SKETCH-N-SKETCH allows the user to draw new shapes directly in the canvas (as in a traditional drawing editor like Illustrator), while the system automatically generates high-level, readable code that, when executed, produces the same shape as drawn in the canvas. This serves as starter code that can be used for subsequent and more complex tasks. [1]

- (Repair Phase) As the design matures, the user can declare new intended relationships among the output by clicking shapes and their attributes with the mouse—*e.g.* to equate the colors of different shapes, to relate their positions, or to group shapes—and SKETCH-N-SKETCH semi-automatically updates the program to satisfy the declared relationships. If the user's intent is ambiguous, SKETCH-N-SKETCH provides an interactive menu of options from which to choose. [1] [2]

- (Refactoring Phase) Our approach embraces unrestricted, text-based programming for tasks that are difficult for, or beyond the reach of, purely direct manipulation-based automation. To reduce the amount of tedious and error-prone text-editing, SKETCH-N-SKETCH provides features for directly manipulating the program text (using the mouse) to perform common code transformations. [3]

Together, these features allow the user to create complex, reusable programs to generate certain classes of designs, using fewer text-edits than in traditional programming languages and fewer mouse-edits than in traditional direct manipulation editors.

Direct Manipulation Programming for Data and Pedagogy. Spreadsheets are used by millions of people—students, educators, office workers, scientists, among many others—to clean, manipulate, and visualize data sets large and small. Because of the ubiquity of this domain, our next area of focus will be spreadsheet programming. To augment the techniques developed in SKETCH-N-SKETCH, we will develop program synthesis algorithms and user interfaces for manipulating and visualizing data. We imagine a system in which the user can freely mix between direct manipulation of data in tabular form (as in Excel), direct manipulation of visualizations of the data (as in Illustrator), and the text-based programs that manipulate them. Compared to many promising improvements to programming models for experts (*e.g.* D3), end user programming (*e.g.* by Flash Fill), and in between (*e.g.* Data Wrangler), my goal with direct manipulation programming is for programs in a general-purpose language to be at the forefront of the user experience. This way, experts will harness the full expressive power of programming, library writers will provide customizations for specific groups of end users, and novices will encounter a gentler learning curve.

I plan to deploy the direct manipulation programming systems that result from our research to motivate and teach programming to students with a variety of interests, both inside and outside a computer science curriculum. Ultimately, my goal is to lay the foundations for enabling *every* direct manipulation system to be a direct manipulation programming system.

[1] Hempel, Chugh. **Semi-Automated SVG Programming via Direct Manipulation.** *User Interface Software and Technology (UIST)*, 2016.
[2] Chugh, Hempel, Spradlin, Albers. **Programming and Direct Manipulation, Together at Last.** *Programming Language Design and Implementation (PLDI)*, 2016.
[3] Hempel, Lubin, Lu, Chugh. **Deuce: A Lightweight User Interface for Structured Editing.** Draft, August 2017.