Tools

Brahm Prakash Mishra and John Sarracino

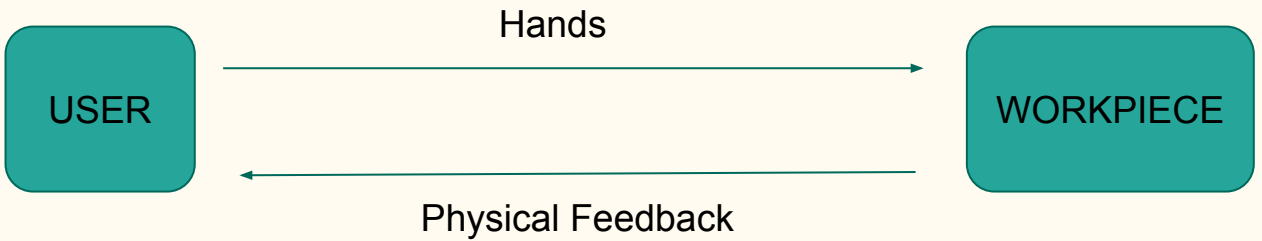
Traditional Fabrication

Cons

- Slow - every step requires conception + execution time
- Industry grade machinery
- First and final - no undo

Pros

- User attention on workpiece
- Interactive editing



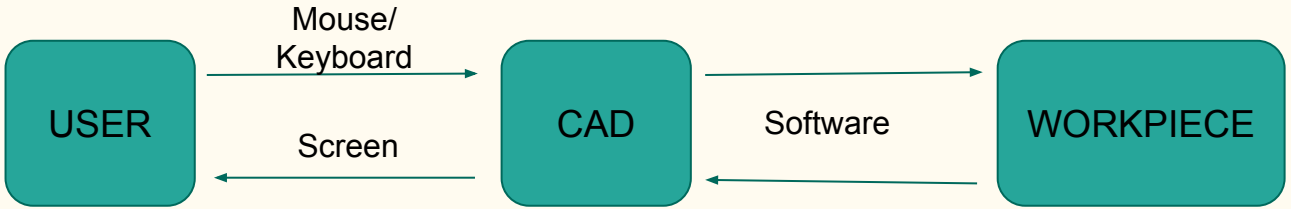
Rapid Prototyping

Cons

- Removes user from workpiece
- Prevents interactive editing

Pros

- Fast iterations
- Precision
- Allows trial and error



Existing Tools for Personal Fabrication

- 3D Printers, Laser Cutters, Ideal for rapid prototyping
- User input via CAD
- New lab setup in MAE

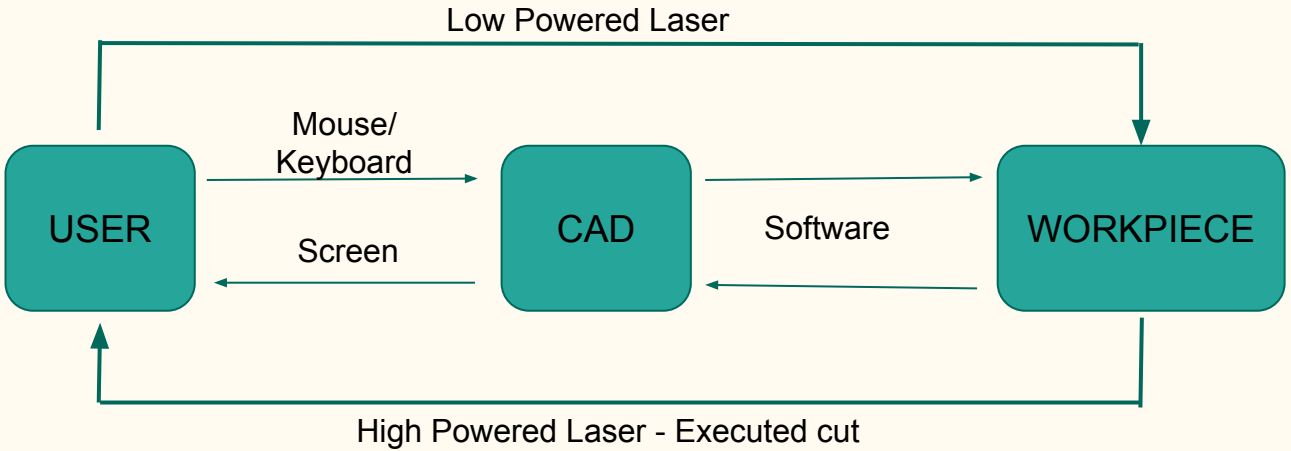
Constructable

Pros

- Retains desirable features of CAD
- Enables Interactive Editing
- Improves design process by keeping user focus on workpiece

Cons

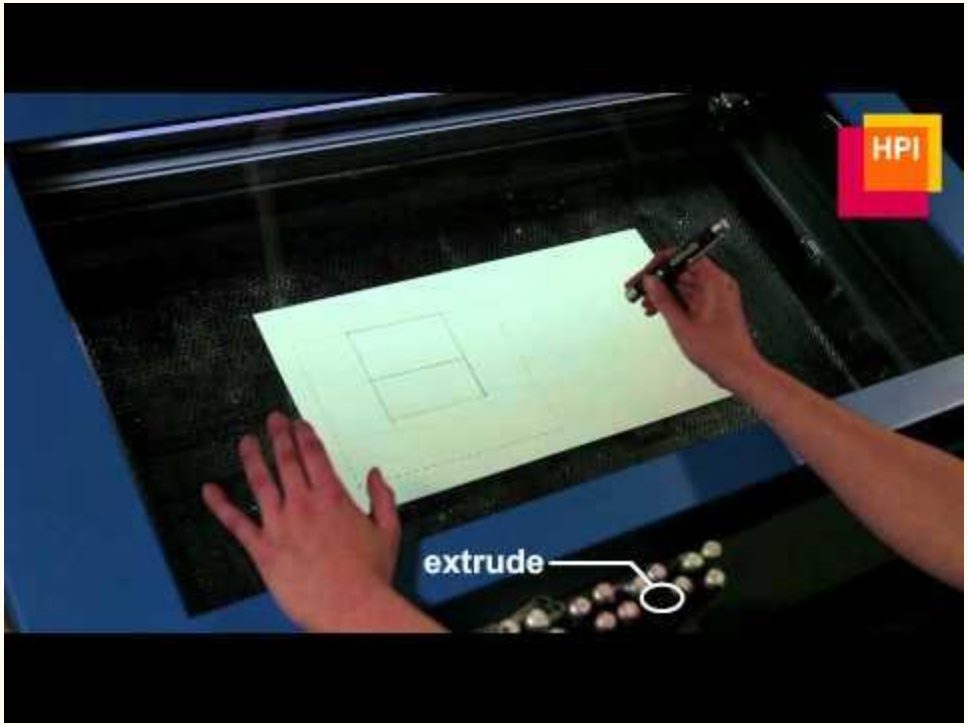
- **Analyse possible cons of constructable, as a tool**



Constructable: Interactive Construction

- Interactive editing has value for artists and designers
- Proposed system adds these benefits to technical projects
- Simultaneously satisficing requirements such as precision
- And improving prototyping speed
- Interactive fabrication $\Longrightarrow$ Interactive construction

Demo



Mechanism

- Different kinds of low powered, proxy lasers for input
- Each with its own set of constraints
- Camera reads the laser movement and processes
- Laser pointer movement retraced using a high powered laser

Impactful features

- Discussion: What was novel about the technique in the paper?

- Good interaction design with the proxy lasers, using physical objects as reference.
- Ariel points out that cognitive load of user is reduced due to uniformity across lasers.
- Immediately moved on to point out lack of user-evaluation data. - Faulty generalization.

- Dicto Simpliciter

Critique - Deconstructing the constructable

- Benefit of interactive editing for artists (and designers) - creative process inspired by partially complete workpiece. Does this really extend to technical projects? (Absence of user data) *-Ex Falso Quodlibet*
- Points out two defects of interactive fabrication - Slow editing, precision. Claims constructable has ability to create functional mechanical devices (i.e, satisfices precision) while maintaining immediateness of traditional interactive editing devices. Slow editing - unresolved for most part.

- No quantitative measures of ‘precision’! *- Reification*
- Zhou - *The precision the system can provide is limited. For example, circle tool always produces a perfect circle but the diameter and location remain freehand. This indicates that the precision is only to the level of shape and it is impossible to precisely determine the parameter of the shape. This might cause problem even when making prototypes that require precise fit*

Concerns with suboptimal designs

- Why exclusion of ANY form of display?
- Speed of prototyping - takes severe hit when user makes a mistake, VERY common while prototyping

Concerns with feasibility

- What about the 3rd dimension that CAD offers? - Angelique
  - 3D printers are able to construct objects with depth definitions. The proposed machine can only do 2D fabrications, since no mechanism

Verdict

Would you be looking to use Constructable over existing CAD software in conjunction with a 3D Printer/Laser cutter

What audience do you think the tool is targeted at?

Past/Present/Future of UI Software Tools

- Reflective survey/insight paper by Brad Myers, 1999 (almost 20 years old)
- **UI builder**: a software library and/or graphical toolkit for authoring UI view

Goals

1) Discuss the proposed criteria for evaluating UI builders:

2) Discuss current/future criteria for UI builders:

Context

- Researched in response to personal computing -- state-of-the-art solution was “**write low-level code**”
- Ubiquitous + context-aware computing were predicted but not yet present

Discussion: If you were researching UIs in the 1990s, **what would you focus on**? What other **domains** does this problem **resemble**?

Goals

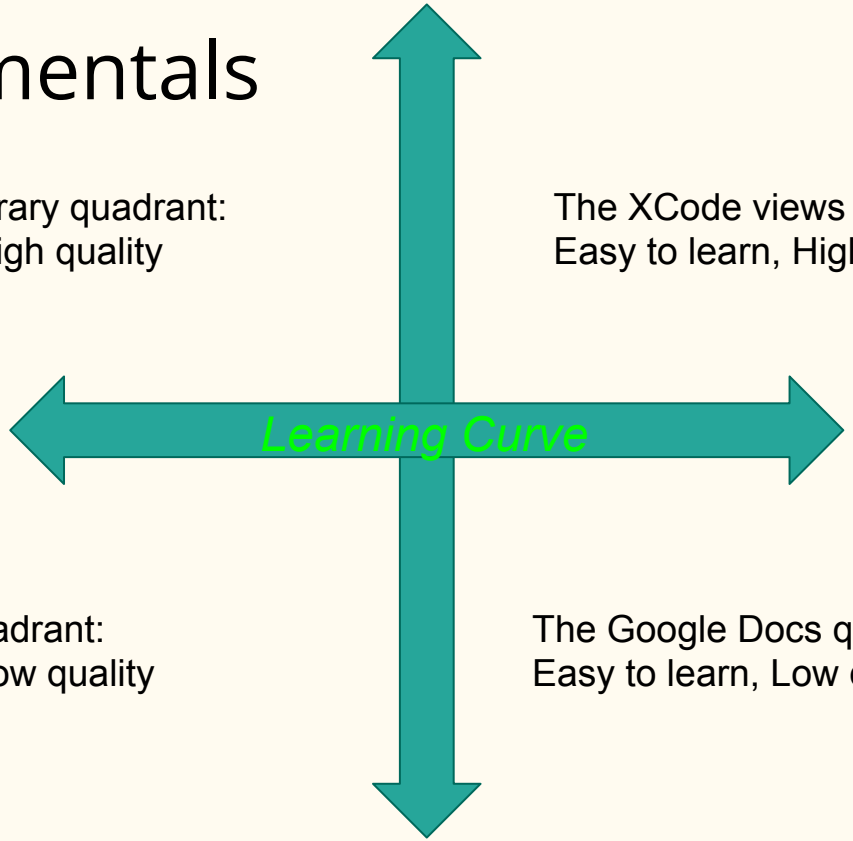
1) Discuss the proposed dimensions for evaluating UI builders:
   a) The fundamentals: **learning curve** vs. **quality** of output UI
   b) **Predictability**: is the UI builder's result predictable from its input?

2) Discuss current/future criteria for UI builders:
   a) Automatic specialization: **one** app for **many** systems
   b) Modular interactivity: Toolkit support for different interaction modalities

The Fundamentals

The software library quadrant:
Hard to learn, High quality

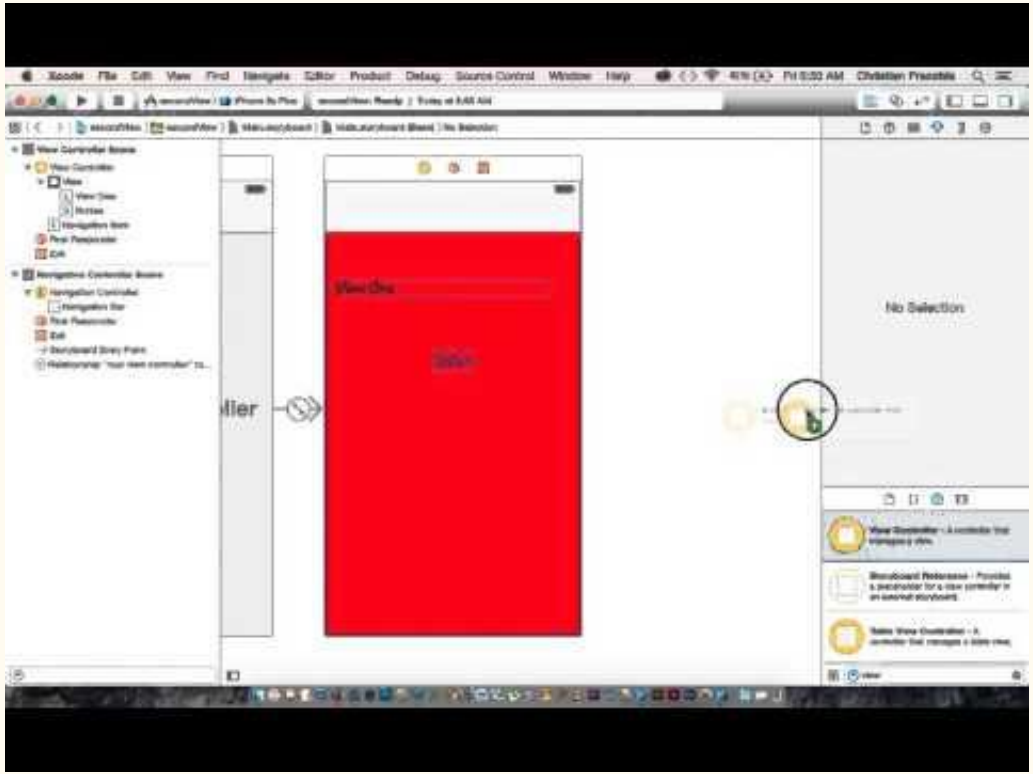
The XCode views quadrant:
Easy to learn, High quality



The Beamer quadrant:
Hard to learn, Low quality

The Google Docs quadrant:
Easy to learn, Low quality

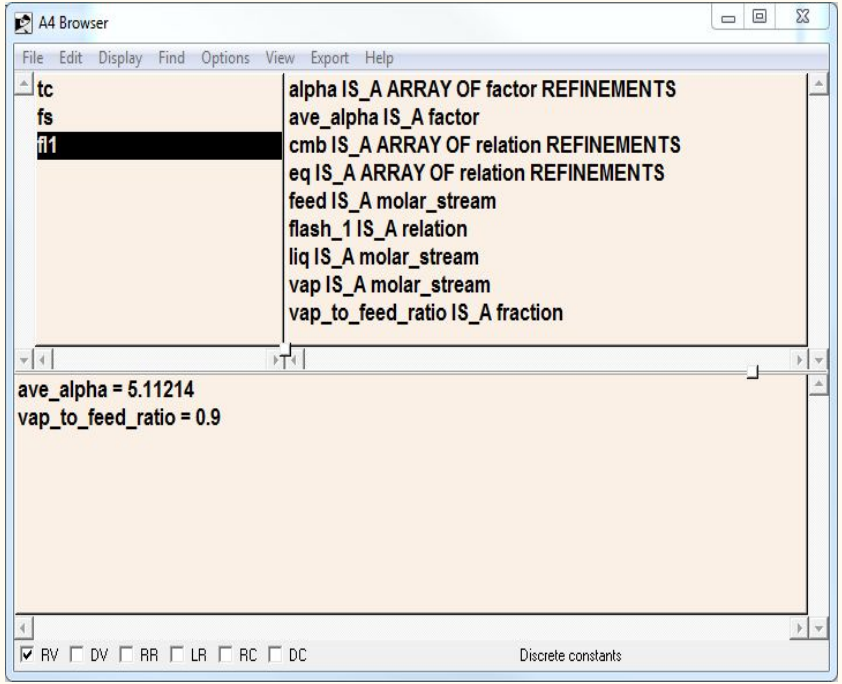
Easy to learn, High quality



Hard to learn, Low quality

```
proc save { } {
    set data [.text.t get 1.0 {end -1c}]
    set fileid [open $filename w]
    puts -nonewline $fileid $data
    close $fileid
}
```

TK



Predictability

- A good authoring tool is **predictable** -- the author should be able to predict the **change** an edit to the **input** will have on the **output**.
- There was a lot of research effort into high-level specifications for UIs:
  - Constraints: e.g. “dialog box X must be attached to gutter panel Y”
  - Formal Models: e.g. “Dialog box Z follows when Submenu item j in Menu item M is clicked”
  - Input/Output UI synthesis: e.g. “I need the system to take in a list of files and support view, copy, and delete.”
- Discussion question: Why might high-level specifications be unpopular?

Why is this relevant today?

- UbiComp, cloud, and contextual computing are all **here to stay**.

- Discussion question: What are some fundamental **issues** for a UI developer in the modern setting? Were UI builders **designed** to solve these problems?

- Current work: **domain-specific** UI builders (e.g. web applications, proof assistants)

Goals

1) Discuss the proposed dimensions for evaluating UI builders:
   a) The fundamentals: **learning curve** vs. **quality** of output UI
   b) **Generality**: does the UI builder solve a **relevant** problem?

2) Discuss current/future criteria for UI builders:
   a) Automatic specialization: **one** app for **many** systems
   b) Modular Interactivity: Toolkit support for different interaction modalities

Automatic Specialization

- Currently, people use **many different systems** to access the same **application** -- for example, an email client.
- It’s difficult and tedious to consistently author a UI for each system.
- High-level specs can help -- write one spec for each system, and one spec for the app.

Modular Interactivity

- Desktop Interactivity was relatively **simple** -- UI builders supported it with a set of **primitives**.
- Discussion question: In terms of Interactivity, how is UbiComp fundamentally different from desktops? How is contextual computing different from desktops?
- Discussion question: Can previous UI builder work on Interactivity handle UbiComp? Contextual computing? Why or why not?