Boxer and the Tradition of Materialised Programming

Antranig Basman

A paradoxical tradition

Whilst Boxer spearheaded the tradition, most members joined it unknowingly of their predecessors

What characterises the tradition?

We'll use as a working definition the one from our paper Software and How it Lives On (Basman, Church, Klokmose & Clark, 2016) and say that a programming environment is materialised to the extent that it is free of divergence

> *divergence - a discrepancy between its bookkeeping, runtime state and the state with respect to which it can be externally authored*

In practice, this definition is insufficient, but good enough to draw a boundary around the systems of interest. Has easily checkable consequences for the system's memory model in that it's free of usually ubiquitous constructs such as the stack and the heap (see Sitaker, 2016)

Boxer itself

Was designed explicitly with this ergonomic and engineering aim - A Principled Design for an Integrated Computational Environment - ACM HCI (diSessa, 1985) explains that the purpose of its naive realism was explicitly to enable a simple surrogate model whereby users could step through the trajectory of an execution by visualising successive configurations of the system itself.

> *Even those designers who are willing to divert resources like the display screen from highly tuned functionality to understandability have almost universally opted for user interfaces which act as buffers or facades to hide system complexities from the user rather than to search for a simplicity which could be shown*

Chris Hancock's Flogo II

Described in 2003 PhD Thesis Real-Time Programming and the Big Ideas of Computational Literacy

The only member of the tradition created with knowledge of its predecessor

Inspired by Boxer's design but addressing criticisms (p.29-30) of its failure to support real-time applications and asynchrony

Materialised Recursion in Flogo II - Boxtower

```
LEFT: {-200} - 200
BOTTOM: {-200} - 200
SIZE: {400} 400
TOP: {200} BOTTOM + SIZE
RIGHT: {200} LEFT + SIZE
MIDDLE: {0} LEFT + SIZE div 2
RECT(TOP + 4,LEFT - 4,BOTTOM - 4,RIGHT + 4,'gray)
CX: {-42} FAX(0)
CY: {182} FAY(0)
CIRCLE(CX,CY,5,'gray,50)
WHEN CX < MIDDLE
    BOXTOWER(BOTTOM,LEFT,SIZE div 2)
        TOP: {0} BOTTOM + SIZE
        RIGHT: {0} LEFT + SIZE
        FILL: {RED    } on resume choose('red,'blue,'green,'yellow)
        RECT(TOP,LEFT,BOTTOM,RIGHT,'gray,PCT,FILL)
        WHEN CY > TOP and SIZE > 8
            MIDDLE: {-100} (LEFT + RIGHT) div 2
            WHEN CX < MIDDLE
                BOXTOWER(TOP,LEFT,MIDDLE - LEFT)
            OTHERWISE
                BOXTOWER(TOP,MIDDLE,RIGHT - MIDDLE)
                    TOP: {100} BOTTOM + SIZE
                    RIGHT: {0} LEFT + SIZE
                    FILL: {YELLOW} on resume choose('red,'blue,'green,'yellow)
                    RECT(TOP,LEFT,BOTTOM,RIGHT,'gray,PCT,FILL)
                    WHEN CY > TOP and SIZE > 8
                        MIDDLE: {-50} (LEFT + RIGHT) div 2
                        WHEN CX < MIDDLE
                            BOXTOWER(TOP,LEFT,MIDDLE - LEFT)
                        OTHERWISE
                            BOXTOWER(TOP,MIDDLE,RIGHT - MIDDLE)
                    START_TIME: {317683} on resume tenths_now()
                    AGE: {4  } tenths_now() - START_TIME
                    PCT: {28  } min(100,AGE % (SIZE div 7))
        START_TIME: {317683} on resume tenths_now()
        AGE: {4  } tenths_now() - START_TIME
        PCT: {14  } min(100,AGE % (SIZE div 7))
OTHERWISE
    BOXTOWER(BOTTOM,MIDDLE,SIZE div 2)
```

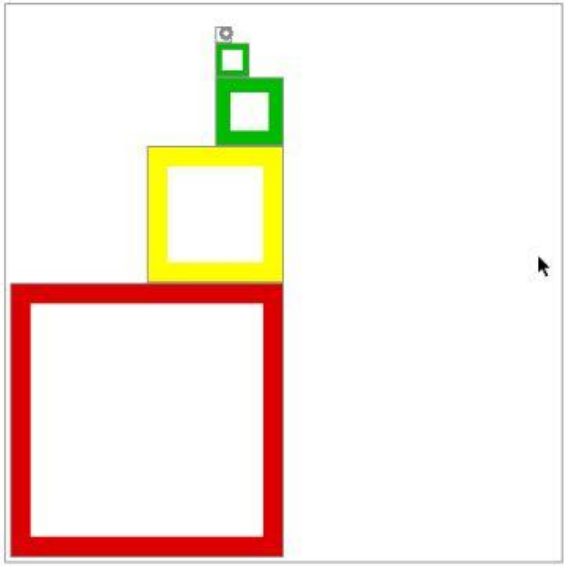

Fig. 9. Snapshot of live program execution of BOXTOWER program. Note live nested display of recursive calls.

Flogo II's motivation for materialisation

This creates a steady frame for program execution, and, consequently, a meaningful dynamic display.

And yet, procedural code remains less live than declarative code. Much procedural code spends most of its time in an inert state, waiting to be executed. When it does run, its execution flashes by as part of a thread of execution that may jump around in the program. Flogo II's steady trace of procedural code is helpful for sorting out what the procedural code is doing or has done.

Subtext

A family of systems designed by Jonathan Edwards (2005-present), without knowing reference to Boxer's tradition

Most condensed motivation in Subtext: Uncovering the Simplicity of Programming appealing to Don Norman's "Gulfs of Execution and Evaluation."

Parallels Boxer's "Copy and Execute" with a "Theory of Copying" as its mechanism for elaboration/execution.

"Copying is isormophic" notion mirror's Boxer's semantic for copying/executing material containing internal/external links.

Subtext (2005 treatment)

> *Calling, referring, instantiating, sharing, refining, modularizing, and versioning all become forms of copying. The ultimate usability feature is coherence.*

Echoes Boxer's notion of detuned primitives and diffused functionality

> *Subtext could be described as functional prototypes.*

A distant echo of "copy and execute" but actually a rather different model. Subtext works to preserve provenance of raw materials for execution, and features much milder polymorphism at the executing site

Fluid's Infusion Framework

Our own framework/language (2009-present) originally motivated to create highly accessible, deeply adaptable web content.

Led to retrospective discovery of Open Authorial Principle (Basman, Lewis, Clark, 2018)

> *The design should allow the **effect** of any expression by one author to be replaced by an **additional expression** by a further author*

A model of additive authorship in networks of authors, echoing Boxer's LaDDER model (Layered Distributed Development of Educational Resources) (Boxer Profile: Component Computing, diSessa, 2001)

Inspiration from the Web

The materialised structures of the web following from principles such as REST, and the DOM's addressible structure exposed as CSS selectors, eventually motivated a fully materialised design, free of divergence (2018 paper)

- The expressions of multiple collaborating authors cannot be adequately negotiated without a stable coordinate system addressing the entire design, including those parts currently in execution

- The experience of negotiating over styling decisions via a browser's CSS inspector with a network of authors is a great model for negotiating application structure in a LaDDER-like network

Integration

Not in the sense of Boxer's "integrated" (self-contained) functionality - instead refers to integration with facilities outside the system

The Mythical Matched Modules (Kell, 2009) introduces the notion of an Integration Domain - expresses how values appearing at different coordinates are correlated, and how the contents of memory can be explained - rather than necessarily designating execution

- Complementary both to Boxer's strategy for eliminating divergence as well as Subtext's

A “Boxer in a Box” limits its utility

Chris Hancock’s 4th criticism (2003 thesis) - that Boxer represents a parallel world - a hermetic application

A “Boxer on the web” (as with Bruce Sherin’s implementation) would remedy this to a good extent, but not completely

Henri’s desire for Boxer is that it could be used to orchestrate other applications

- This is an aspiration but ultimately would require these applications to be re-expressed or wrapped

- Since there is currently no medium for this integration, these capabilities tend not to be exposed, but if there were, they might be

- Jeremy notes that all of the integration facilities on his computer are incommensurable and unusable

Externalisation

Crude, experimental system, the GPII Nexus (2018) exposed a coordinatised executing design over web protocols. Only weakly divergence-free.

Underlying theory developed in Tracing a Paradigm for Externalization: Avatars and the GPII Nexus (Clark, Basman, 2017)

Could we make Boxer's boxes entirely transparent - to view from outside the system?

- possibly not at the same time as preserving its execution semantics as a core facility, because of their exceptional demands on polymorphism

- if we didn't, could execution semantics be represented at some other level of the system design?

Boxer Futures

- Boxer on the web (as with Bruce Sherin's BoxerJS)

- Asynchronous Boxer (as with Chris Hancock's Flogo II)

- Externalised Boxer (with a transparent, malleable file format)

- Collaborative Boxer (with the file format synchronised to a distributed group, as with Webstrates)

And then - can we extract some kind of common substrate from "Boxer variants" that can support Mark Guzdial's ecology of languages, where "Classic Boxer" appears as just one personality of the language?

Sustaining extremely long-term, sporadically funded, highly distributed and ambitious research programmes?

- Over to Luke