**littleBits Synth Kit**
as a physically-embodied, domain specific, functional programming language

James Noble & Timothy Jones
Victoria University of Wellington
New Zealand

“no programming”

**Cognitive Dimensions**

- **Medium**: circuit elements, “bits”
- **Activities**: exploratory design, modification, incremental construction, transcription?
- **Visibility**: general structure high, element settings low
- **Diffuseness**: compact, small size
- **Viscosity**: low viscosity, easy to change
- **Secondary Notation**: labels, colours, position

“no programming”
Cognitive Dimensions

- **Hidden Dependencies:**
  - intermodule dependencies *explicit*,
  - parameter dependencies *hidden*
- **Role Expressiveness:** geeks not folkies...
- **Premature Commitment:** easy to change
- **Progressive Evaluation:** delay to reorganise,
  - parameter changes instant
- **Provisionality:** can move disconnected modules
- **Abstraction:** none

---

But is it Haskell?

```haskell
• data ClipState = O | P
data LB (s :: ClipState) a
  instance Functor (LB s)
  • clip :: LB s a → LB O b → LB s b
  • instance Monad (LB O) where (>>>) = clip

• oscillator :: Knob → Knob → LB O ()
• filter :: Knob → Knob → LB s a → LB O a
• keyboard :: Key → LB O a → LB O (LB P a)
```

```haskell
• example :: LB O ()
exemple = do
  trigger ← keyboard D wire
  oscillator 20 40
  filter 80 50 trigger
  speaker

```

---

But is it Haskell?

```haskell
```

---

But is it Haskell?

```haskell
```

---