Diagrams
A Functional EDSL for Vector Graphics

Ryan Yates    Brent Yorgey

FARM
Vancouver, BC, Canada
5 September 2015
What is diagrams?

- Domain-specific language for vector graphics
What is diagrams?

- Domain-specific language for vector graphics
- Embedded in Haskell
What is diagrams?

- Domain-specific language for vector graphics
- Embedded in Haskell
- 7+ years of development
What is diagrams?

- Domain-specific language for vector graphics
- Embedded in Haskell
- 7+ years of development
- Large, active, creative community
Why an EDSL?

- Powerful, programmable alternative to Illustrator, Inkscape, PGF/TikZ
Why an EDSL?

- Powerful, programmable alternative to Illustrator, Inkscape, PGF/TikZ
- Tools influence the creative process
cabal install diagrams

http://projects.haskell.org/diagrams
(second Google result!)
diagrams

---

**diagrams** is a **powerful, flexible, declarative** domain-specific language for creating vector graphics, using the **Haskell programming language**.

---

### Get started

- Read the quick start tutorial or the user manual.

### Get excited

- Check out the gallery for example images and code.

### Get connected

- Drop by the #diagrams IRC channel with questions, or post them to the mailing list.

---

#### News

- **May 27, 2015**
  - Jeff spoke about diagrams at the New York Haskell users' group. Diagrams: Composition, Envelopes and Alignment. Part 1 is about composition and envelopes, and part 2 talks about animated GIFs and the design of the arrow API. The slides are also available.

- **April 19, 2015**
  - Diagrams 1.3 released! Look for a blog post soon with a rundown of new features. In the meantime, see the migration guide for help porting existing diagrams code to 1.3.

- **November 24, 2013**
  - Brent gave a talk on diagrams at the New York Haskell user's group. Part 1 presents a basic introduction to the library, and part 2 talks about mathematical abstraction and DSL design. The slides are also available.
If you want to place two diagrams next to each other using the local origin of the second diagram, you can use something like `beside = flip beside important`, that is, use a vector in the opposite direction and give the diagrams in the other order. Since placing diagrams next to one another horizontally and vertically is quite common, special combinators are provided for convenience. `beside` and `above` are specializations of `beside` which juxtapose diagrams in the x- and y-directions, respectively.

![Diagram](http://projects.haskell.org/diagrams/doc/manual.png)

**Juxtaposing without composing**

Sometimes, one may wish to position a diagram next to another diagram without actually composing them. This can be accomplished with the `beside` function. In particular, `beside v d1` returns a modified version of `d1` which has been translated to be next to `d2` in the direction of `v`. (In fact, `beside` itself is implemented as a call to `juxtapose` followed by a call to `cone`.)

![Diagram](http://projects.haskell.org/diagrams/doc/manual.png)
Demo: visualizing binary trees
Tree visualizations
More examples
3D Trees

seeds/seeds.html

http://www.cs.rochester.edu/u/ryates/art/seeds/
Weaving a Torus

http://mathr.co.uk/blog/2013-04-05_weaving_a_torus.html
Parking in Westminster

Cretan maze

http://www.corentindupont.info/blog/posts/2014-02-17-Cretan-Maze.html
Puzzles

Stencil diagrams

https://readerunner.wordpress.com/2014/04/29/red-black-neighbourhood-stencil-diagrams/
Happy diagramming!