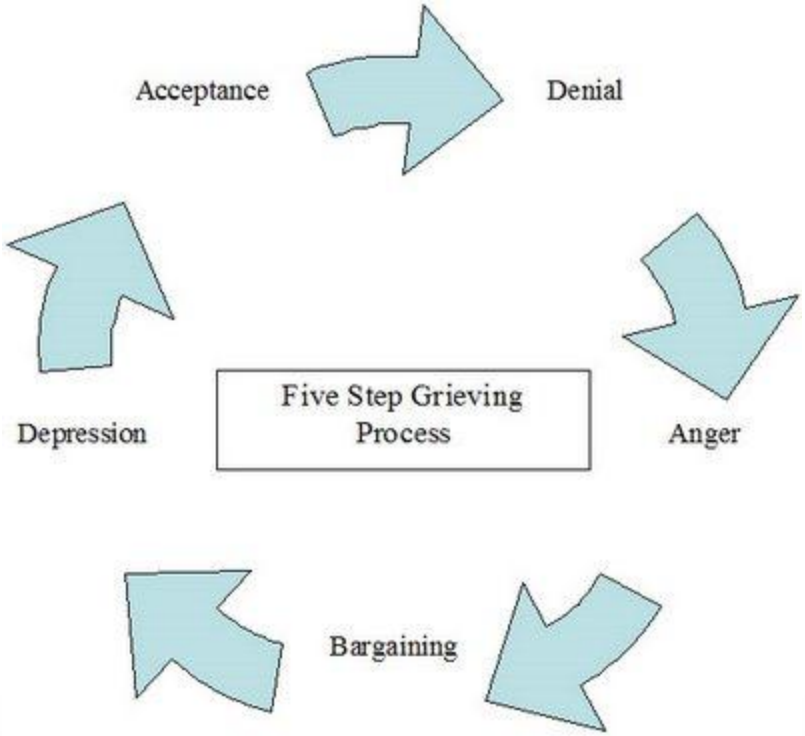
# Elm

## Making the Web Functional

Evan Czaplicki

# Why Elm?

- Inspired by the Kübler-Ross model



# Mission

- Make GUI programming more pleasant
  - Reduce the time and headache to get from idea to reality
  - Make people ask, "How was it not this way before?"

- Make programming more accessible
  - No installation required / interactive compiler online
  - Quick visual feedback
  - Examples!
  - Easy path from novice to expert

# How?

- More pleasant: Functional GUIs
  - Generally enforces safe programming practices at the language level
  - Plays nice with concurrency (see my thesis)
  - Beauty / Elegance
  - More on this later!

- Accessibility: Functional GUIs for everyone!
  - Target the web – the most widely supported GUI platform
  - Be open source – on github
  - Create great resources and examples
  - Free compiler online

"But aren't GUIs imperative?"

# Becoming Functional

"No matter what language you work in, programming in a functional style provides benefits.

You should do it whenever it is convenient, and you should think hard about the decision when it isn't convenient."

– John Carmack

founder of id Software

Wolfenstein 3D, Doom, and Quake

Functional Programming in C++

# Anatomy of a GUI

**Computations**

Purely Functional

High-level

Compositional

Clear semantics

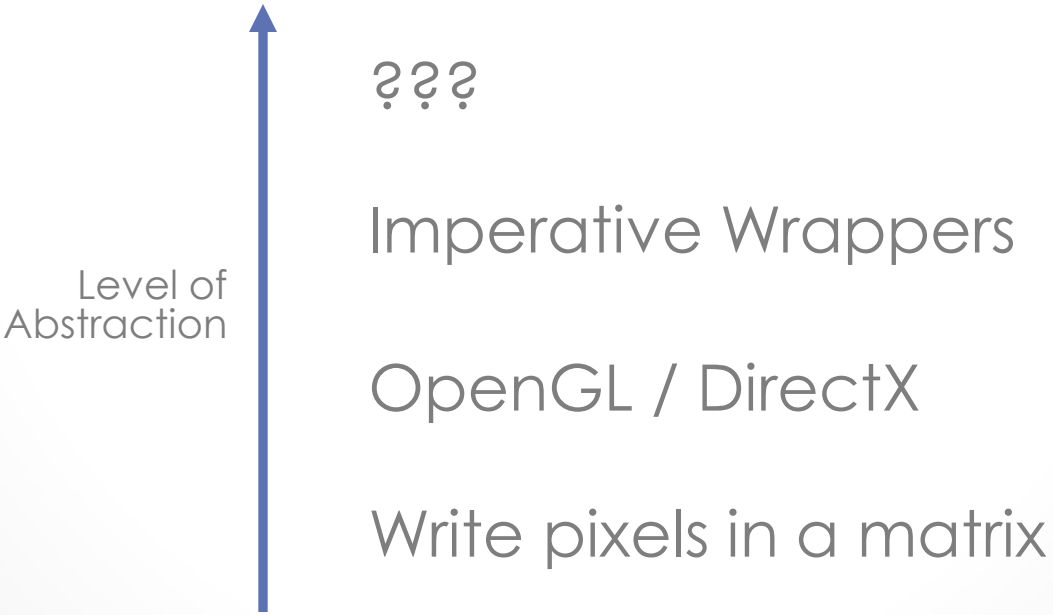
**Graphics**

**Reactions**

How do we do this in a functional way?

# Graphics

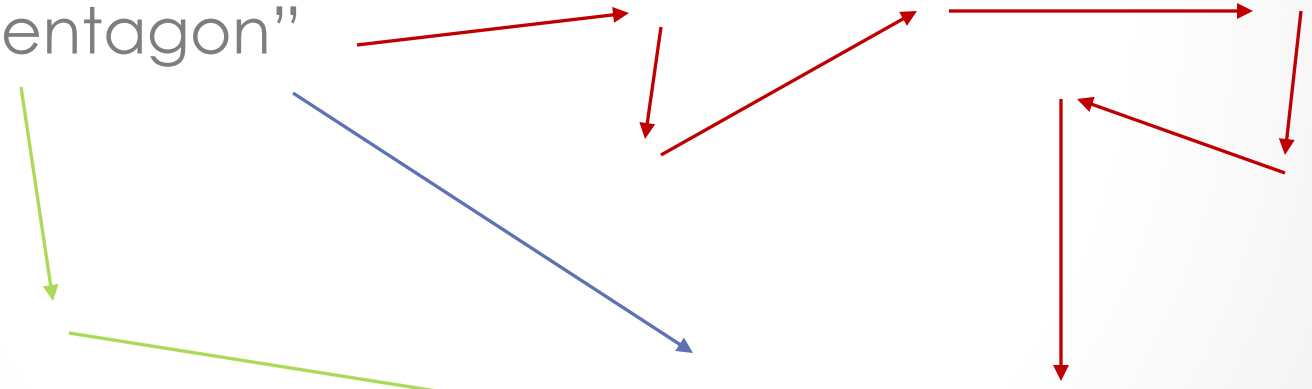
- Historically a very imperative undertaking.
  - Common and accepted ≠ Inherently Correct
- Moving towards higher-level abstractions.



# Graphics

- What is "more abstraction" for graphics?
- How many steps are there between:

"I *want* a pentagon"



"I *have* a pentagon"

# Functional Graphics

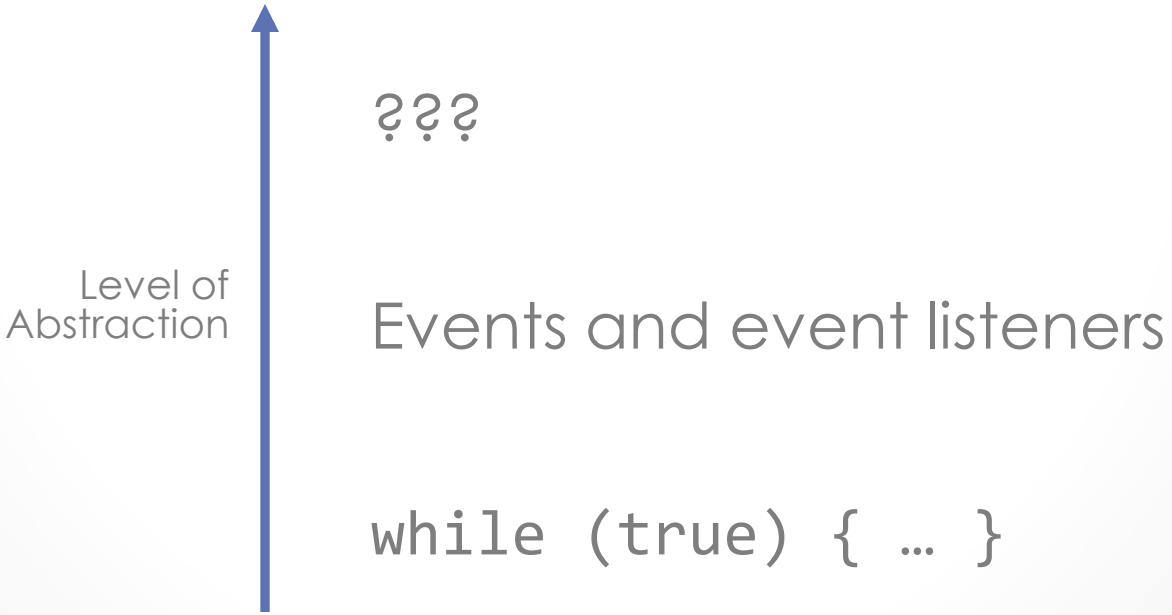
Understanding the `Element` and `Form` types.

# Anatomy of a GUI

| **Computations** | **Graphics** | **Reactions** |
| --- | --- | --- |
| Purely Functional | Purely Functional | |
| High-level | High-level | |
| Compositional | Compositional | |
| Clear semantics | Clear semantics | |

# Reactions

- Historically a very imperative undertaking.
  - Common and accepted ≠ Inherently Correct
- Moving towards higher-level abstractions.



# Functional Reactions

- Normally values are immutable:

```
(3,4) :: (Int,Int)
coordinate :: (Int,Int)
```

- Introduce time-varying values:

```
Mouse.position :: Signal (Int,Int)
```

- A signal is like a stream of events

# Signals

- An animation has type `(Signal Element)`
  - It is an `Element` that changes over time!

- How do we use signals?

```
lift :: (a → b) → Signal a → Signal b
```

```
main = asText (3,4)
main = lift asText Mouse.position
```

- Everything updates automatically!

# Functional Reactions

Creating a GUI with Elm

mouse position

# Anatomy of a GUI

| **Computations** | **Graphics** | **Reactions** |
| --- | --- | --- |
| Purely Functional | Purely Functional | Purely Functional |
| High-level | High-level | High-level |
| Compositional | Compositional | Compositional |
| Clear semantics | Clear semantics | Clear semantics |

# Games in Elm

- Imperative game programming is *too* flexible
  - Change anything, anytime
  - Not a huge deal if your code is disorganized
- Elm *requires* good internal structure
  - No global mutable state, no flipping pixels, no destructive updates
- Example: Pong and a design/code walkthrough.

# Structuring Games

- Every Elm game must have three parts:
  - model the game
  - update the game
  - view the game

- It may be helpful to think of it as a functional variation of Model-View-Controller.

# Mission Revisited

- Make GUI programming more pleasant
- Make programming more accessible

I hope I have convinced you that Elm does both!

# How to help!

- Try out Elm for yourself
- Join the mailing list
- Share your experiences, code, libraries, etc.

# More Information

- *Related Work* and *References* in my thesis.
- The very well-written original paper on FRP (1997).
- Arrowized FRP, a very clever variation of FRP.
- Functional Reactive Programming for Haskell
- Papers on Concurrent ML.
- Try out Elm at elm-lang.org!

# Questions

• • •

Possibly answers too!