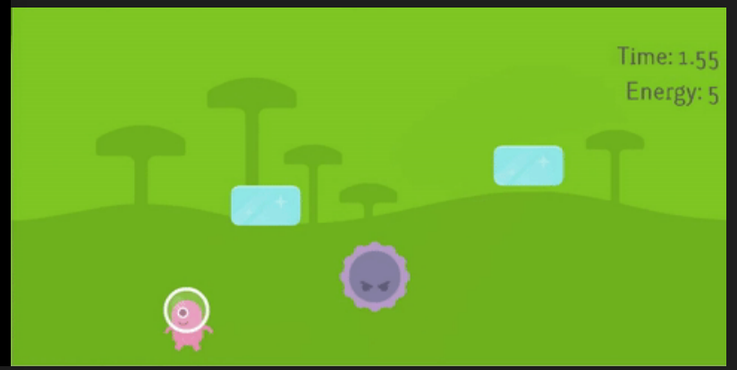
Mobile Game Programming in Haskell

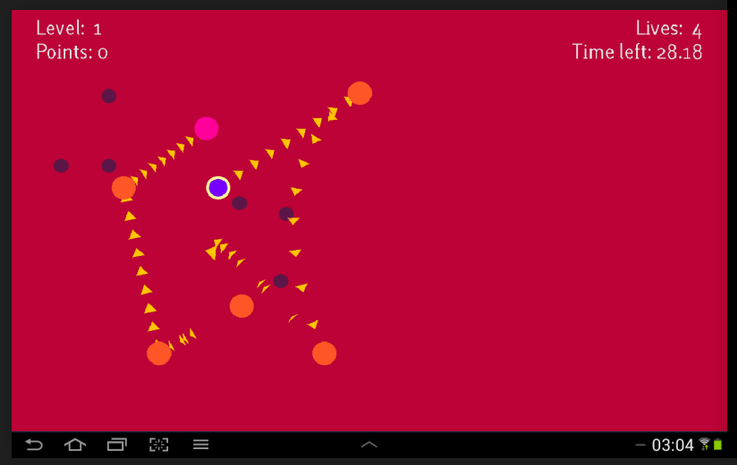
Christina Zeller and Ivan Perez

Keera Studios — keera@keera.co.uk

August - FARM 2019 (Berlin)


Magic Cookies

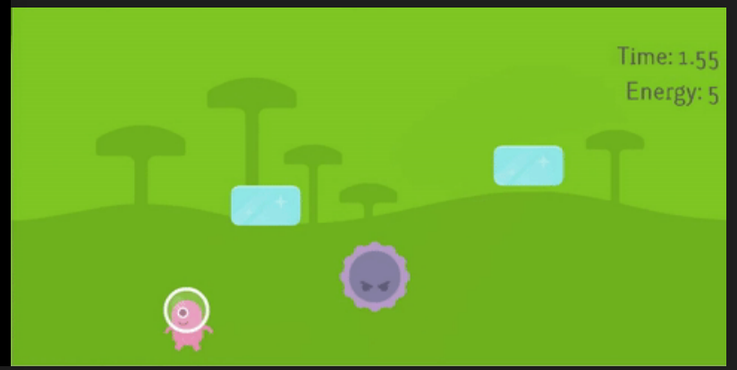

Time: 1.55
Energy: 5

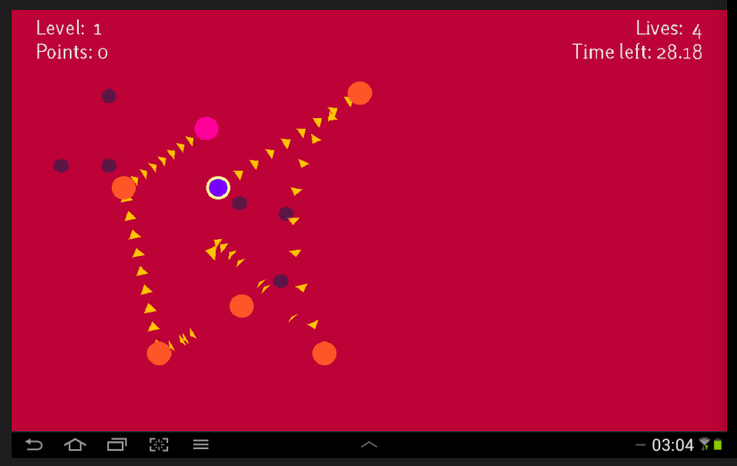

Level: 1
Points: 0
Lives: 4
Time left: 28.18
03:04


LEVEL 0
0 / 5

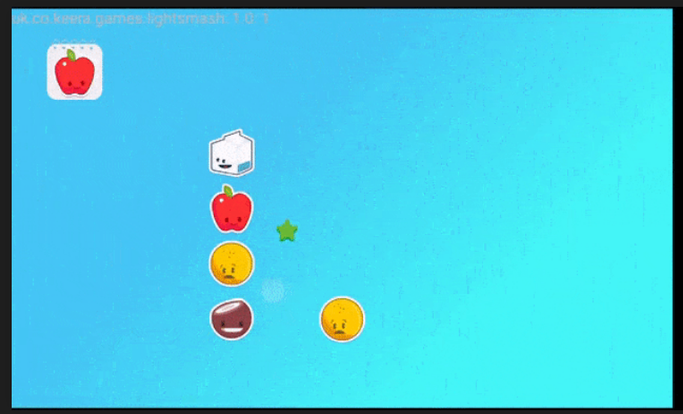

HASHANOID
BROUGHT TO YOU BY
HEERA STUDIOS
TAP THE SCREEN TO PLAY

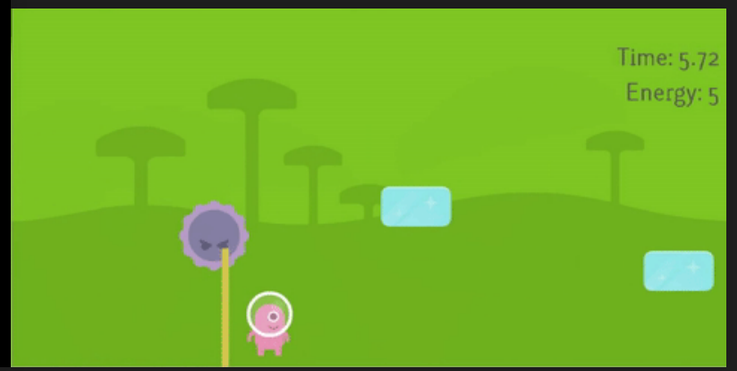
IO - touch and stick!

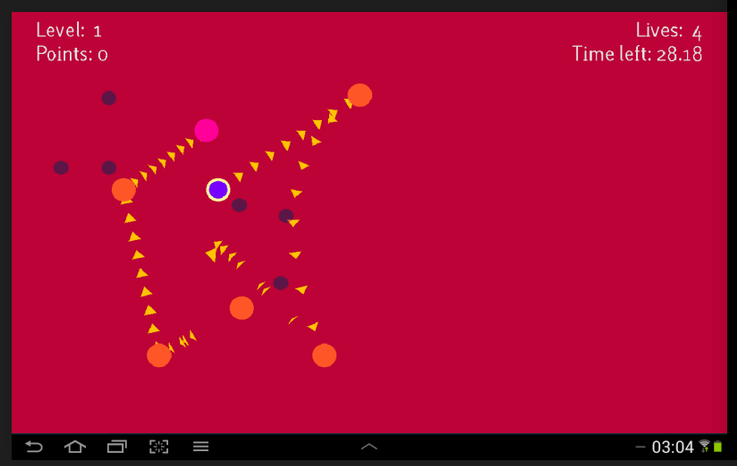




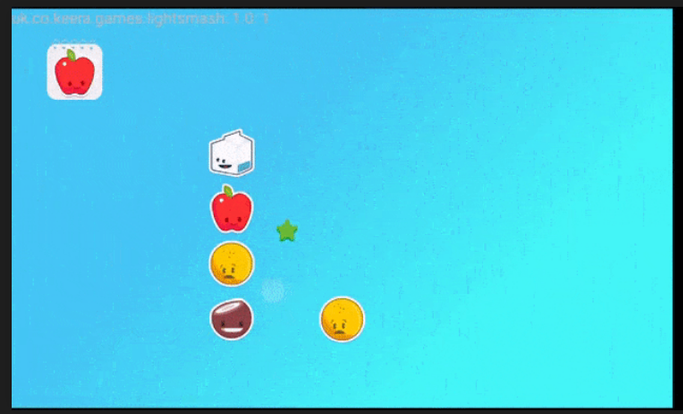
Compositional Structure?

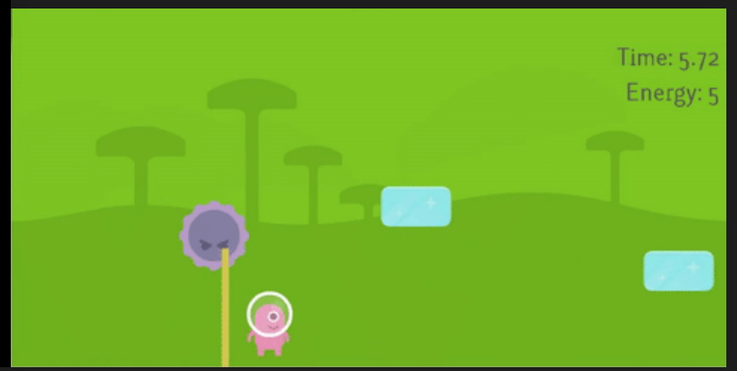


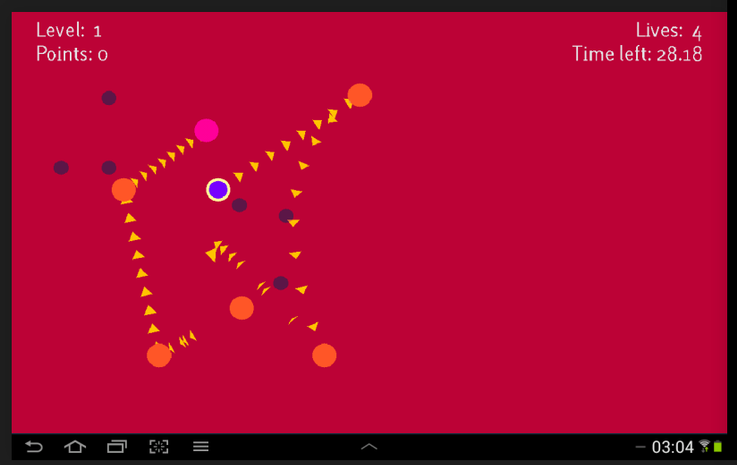




What kind of Input / Backend / Platform?







Purity and Separation of Logic from IO

Compositional Application Architecture

Abstraction from Backend and Platform

Purity and Separation of Logic from IO

Compositional Application Architecture

Abstraction from Backend and Platform

```haskell
paintBall :: Resources -> Surface -> Object -> IO ()
paintBall resources screen object = do
  let  (x, y)    = objectPos object
       ballSize  = 20 -- constant
       img       = ballImg resources
  SDL.blitSurface img Nothing screen
                  (Just (SDL.Rect x y ballSize ballSize))
```

Backend: SDL vs HTML?

```
paintBall :: Resources -> Surface -> Object -> IO ()
paintBall resources screen object = do
  let  (x, y)    = objectPos object
       ballSize  = 20 -- constant
       img       = ballImg resources
  SDL.blitSurface img Nothing screen
                  (Just (SDL.Rect x y ballSize ballSize))
```

Backend: SDL vs HTML?

```
paintBall :: Resources -> Surface -> Object -> IO ()
paintBall resources screen object = do
  let  (x, y)    = objectPos object
       ballSize  = 20 -- constant
       img       = ballImg resources
  SDL.blitSurface img Nothing screen
                  (Just (SDL.Rect x y ballSize ballSize))
```

Transformations: Move to the left?

```
paintBall :: Resources -> Surface -> Object -> IO ()
paintBall resources screen object = do
  let  (x, y)    = objectPos object
       ballSize  = 20 -- constant
       img       = ballImg resources
  SDL.blitSurface img Nothing screen
                  (Just (SDL.Rect x y ballSize ballSize))
```

Transformations: Move to the left?

```
paintBall :: (Int, Int) -> Resources -> Surface -> Object -> IO ()
paintBall (dx, dy) resources screen object = do
  let  (x, y)    = objectPos object
       ballSize  = 20 -- constant
       img       = ballImg resources
  SDL.blitSurface img Nothing screen
                  (Just (SDL.Rect (x + dx) (y + dy) ballSize ballSize))
```

Hidden Structure: Was the ball touched?

```
paintBall :: Resources -> Surface -> Object -> IO ()
paintBall resources screen object = do
  let  (x, y)    = objectPos object
       ballSize  = 20 -- constant
       img       = ballImg resources
  SDL.blitSurface img Nothing screen
                  (Just (SDL.Rect x y ballSize ballSize))
```

Hidden Structure: Was the ball touched?

```
paintBall :: Resources -> Surface -> Object -> IO ()
paintBall resources screen object = do
  let  (x, y)     = objectPos object
       ballSize   = 20 -- constant
       img        = ballImg resources
  SDL.blitSurface img Nothing screen
                  (Just (SDL.Rect x y ballSize ballSize))
```

```haskell
paintBall :: Resources -> Surface -> Object -> IO ()
paintBall resources screen object = do
  let  (x, y)    = objectPos object
       ballSize  = 20 -- constant
       img       = ballImg resources
  SDL.blitSurface img Nothing screen
                  (Just (SDL.Rect x y ballSize ballSize))
```

- Backend
- Transformations
- Hidden Structure

```haskell
ballImg :: ImageSpec
ballImg = ("game/ball.png", Nothing)
```

```haskell
ballImg :: ImageSpec
ballImg = ("game/ball.png", Nothing)
```

```haskell
data ResourceId = IdBall
                | ... -- other resources

appResourceSpec :: ResourceSpec ResourceId
appResourceSpec = ResourceSpec
  { images =  [ (IdBall, ballImg) ]
  , ... -- fonts, colors, audio, etc.
  }
```

```
ballImg :: ImageSpec
ballImg = ("game/ball.png", Nothing)
```

```
data ResourceId = IdBall
                | ... -- other resources

appResourceSpec :: ResourceSpec ResourceId
appResourceSpec = ResourceSpec
  { images =  [ (IdBall, ballImg) ]
  , ... -- fonts, colors, audio, etc.
  }
```

```
-- package-backend-1 / package-backend-2 / package-backend-*
render :: RenderEnvironment -> ... ResourceId ... (Int, Int) ... -> IO ()
```

```
ballCollage :: Object -> Collage ResourceId (Int, Int)
ballCollage object = CollageItem IdBall (objectPos object)
```

```
ballCollage :: Object -> Collage ResourceId (Int, Int)
ballCollage object = CollageItem IdBall (objectPos object)
```

```
data VisualElem rid = VisualImage rid
                    | VisualFilledRectangle rid (Int, Int)
                    | VisualText rid rid String
```

```
ballCollage :: Object -> Collage ResourceId (Int, Int)
ballCollage object = CollageItem IdBall (objectPos object)
```

```
data VisualElem rid = VisualImage rid
                    | VisualFilledRectangle rid (Int, Int)
                    | VisualText rid rid String
```

```
collage :: Object -> (String, (Int, Int)) -> Collage (VisualElem ResourceId) (Int, Int)
collage object (msg, pos) = mconcat
  [ CollageItem (VisualImage IdBall) (objectPos object)
  , CollageItem (VisualText IdFont IdFontColor msg) pos
  ]
```

```
collage :: Object -> (String, (Int, Int)) -> Collage (VisualElem ResourceId) (Int, Int)
collage object (msg, pos) = mconcat
  [ CollageItem (VisualImage IdBall) (objectPos object)
  , CollageItem (VisualText IdFont IdFontColor msg) pos
  ]

-- package-backend-1 / package-backend-2 / package-backend-*
render :: RenderEnvironment -> ... Collage ... ResourceId ... (Int, Int) ... -> IO ()
```

- Backend
- Transformations
- Hidden Structure

```
collage :: Object -> (String, (Int, Int)) -> Collage (VisualElem ResourceId) (Int, Int)
collage object (msg, pos) = mconcat
  [ CollageItem (VisualImage IdBall) (objectPos object)
  , CollageItem (VisualText IdFont IdFontColor msg) pos
  ]

-- package-backend-1 / package-backend-2 / package-backend-*
render :: RenderEnvironment -> ... Collage ... ResourceId ... (Int, Int) ... -> IO ()
```

- Backend
- Transformations
- Hidden Structure

```
CollageItem (Widget wId (VisualImage resId) wSize) (Int, Int)
```

```
collage :: Object -> (String, (Int, Int)) -> Collage (VisualElem ResourceId) (Int, Int)
collage object (msg, pos) = mconcat
  [ CollageItem (VisualImage IdBall) (objectPos object)
  , CollageItem (VisualText IdFont IdFontColor msg) pos
  ]

-- package-backend-1 / package-backend-2 / package-backend-*
render :: RenderEnvironment -> ... Collage ... ResourceId ... (Int, Int) ... -> IO ()
```

- Backend
- Transformations
- Hidden Structure

```
CollageItem (Widget wId (VisualImage resId) wSize) (Int, Int)
```

Sizes of elements?

Purity and Separation of Logic from IO

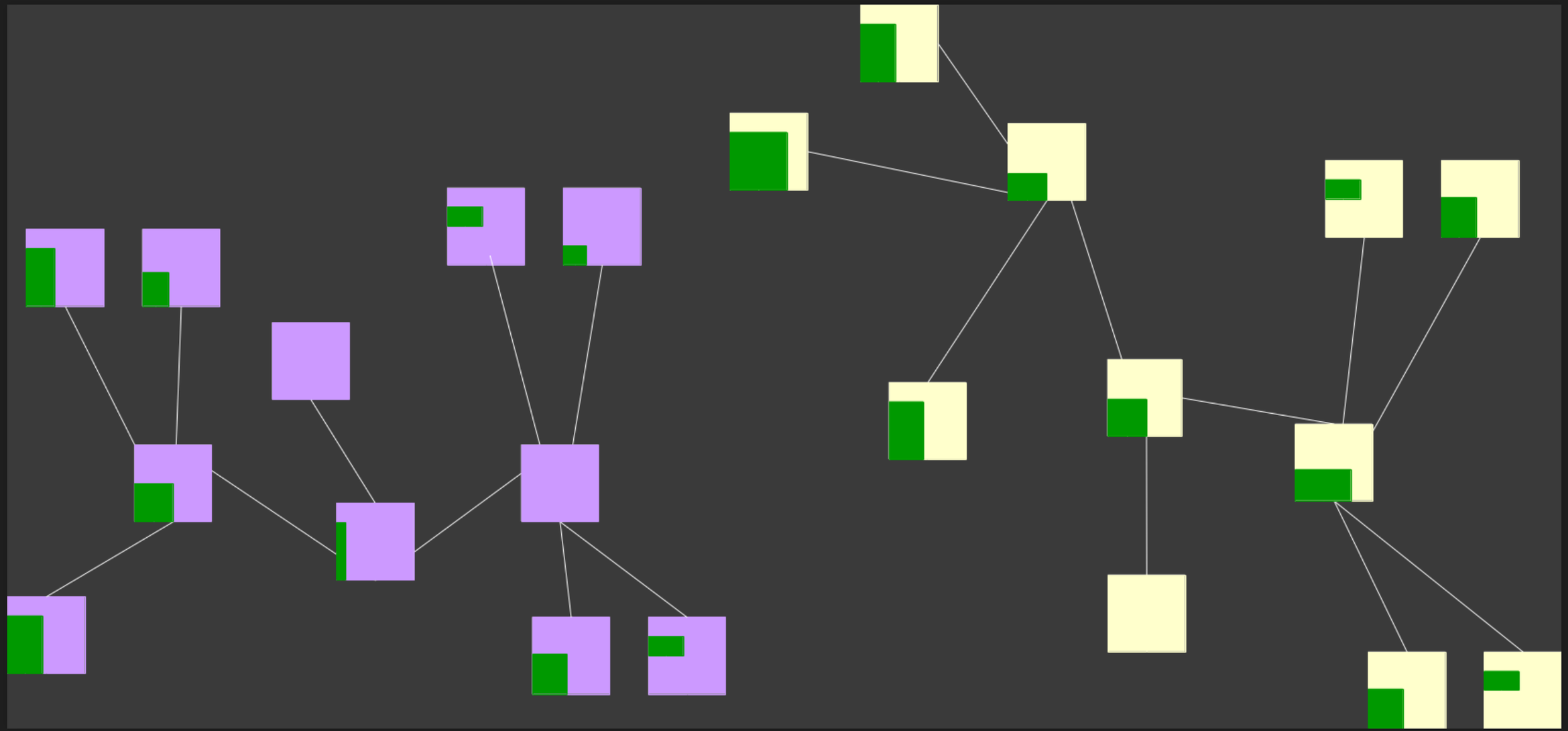
Compositional Application Architecture

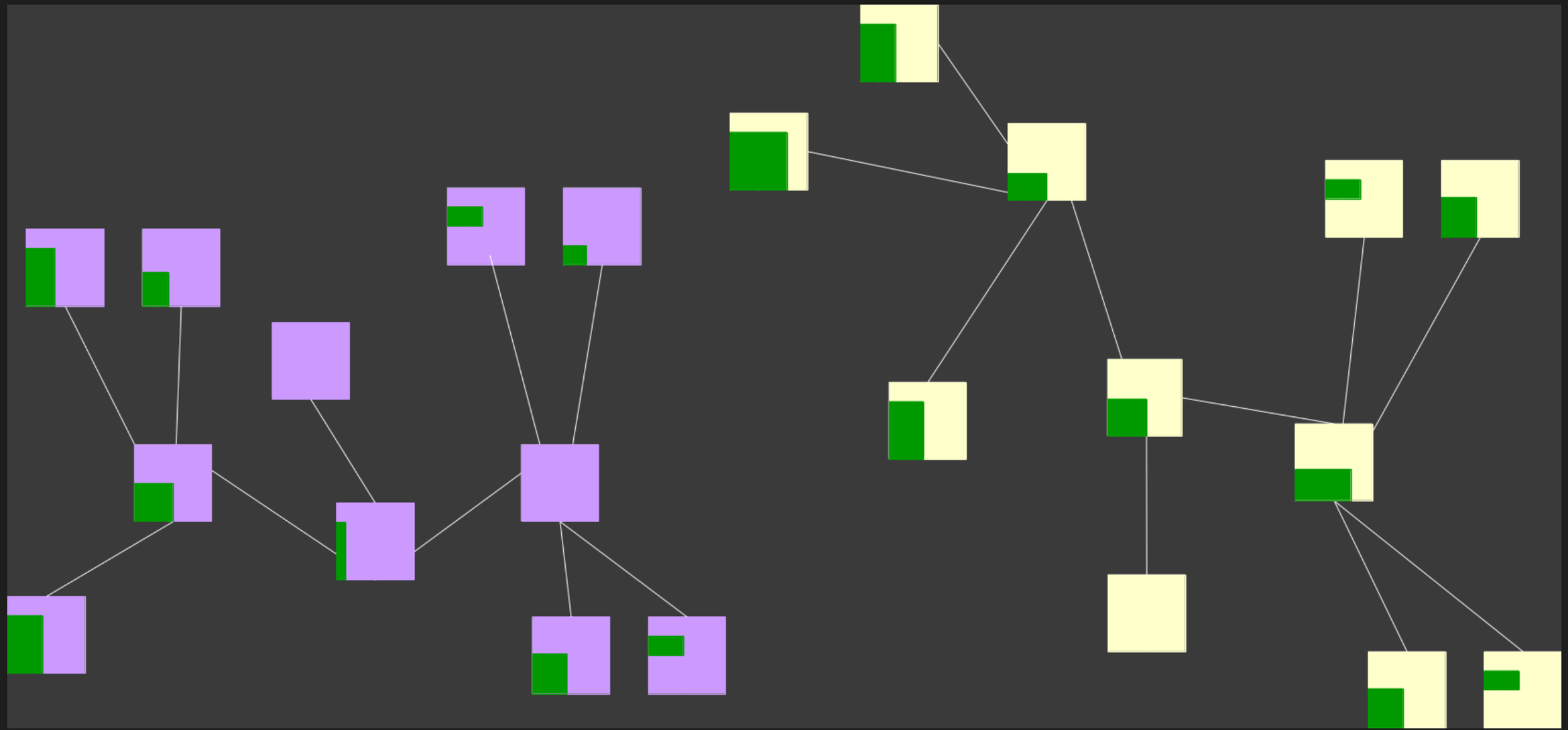
Abstraction from Backend and Platform

Purity and Separation of Logic from IO

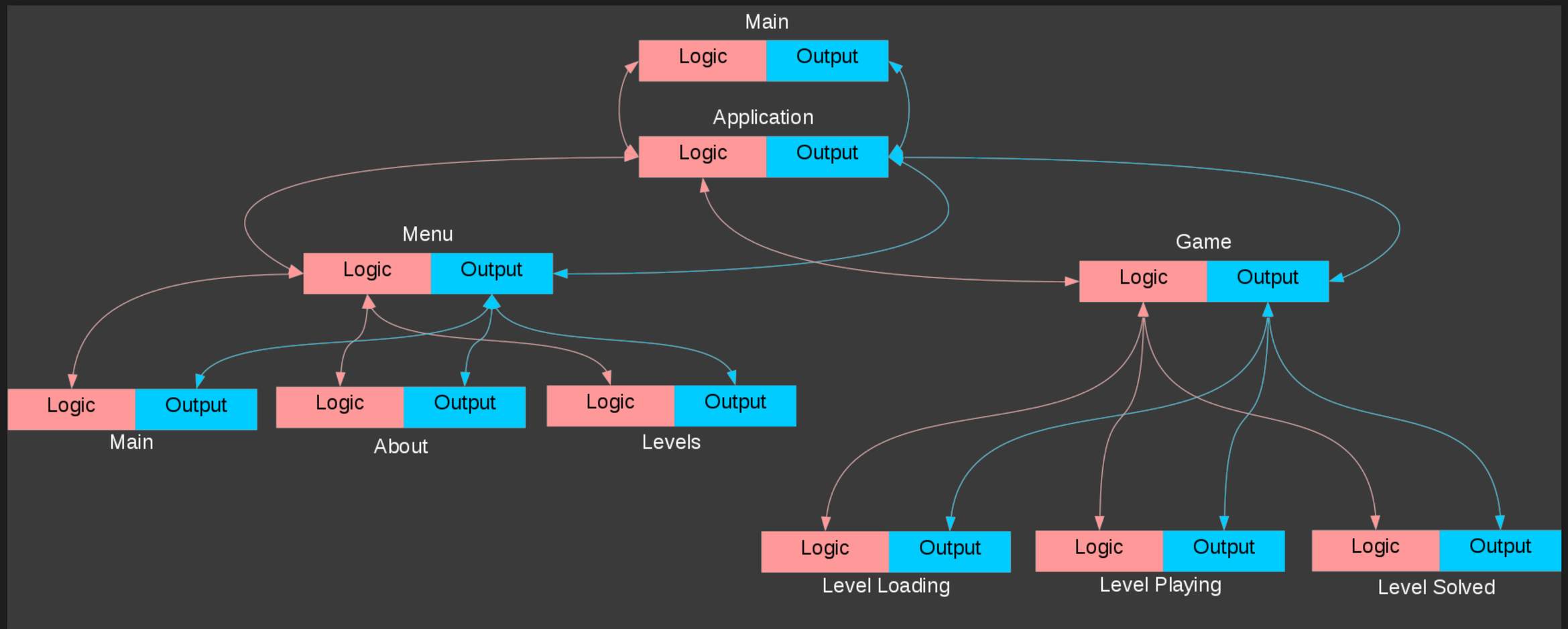
Compositional Application Architecture

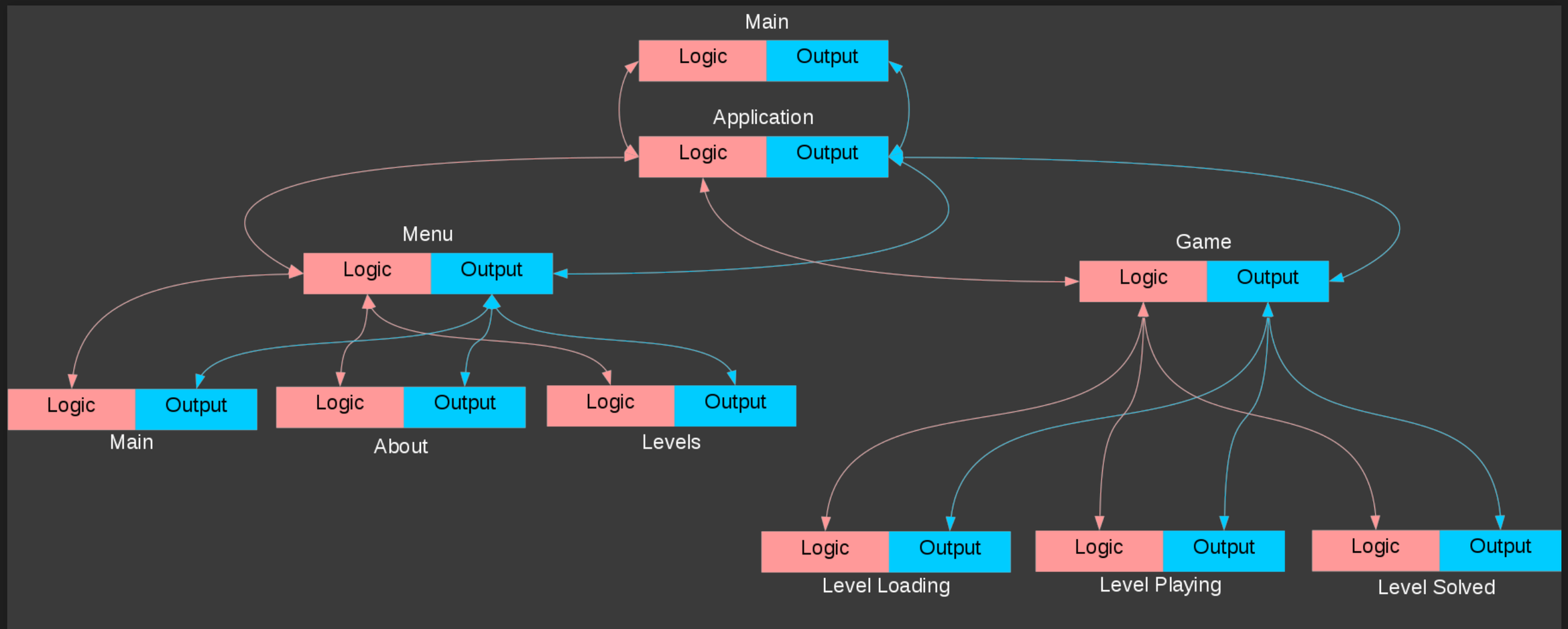
Abstraction from Backend and Platform



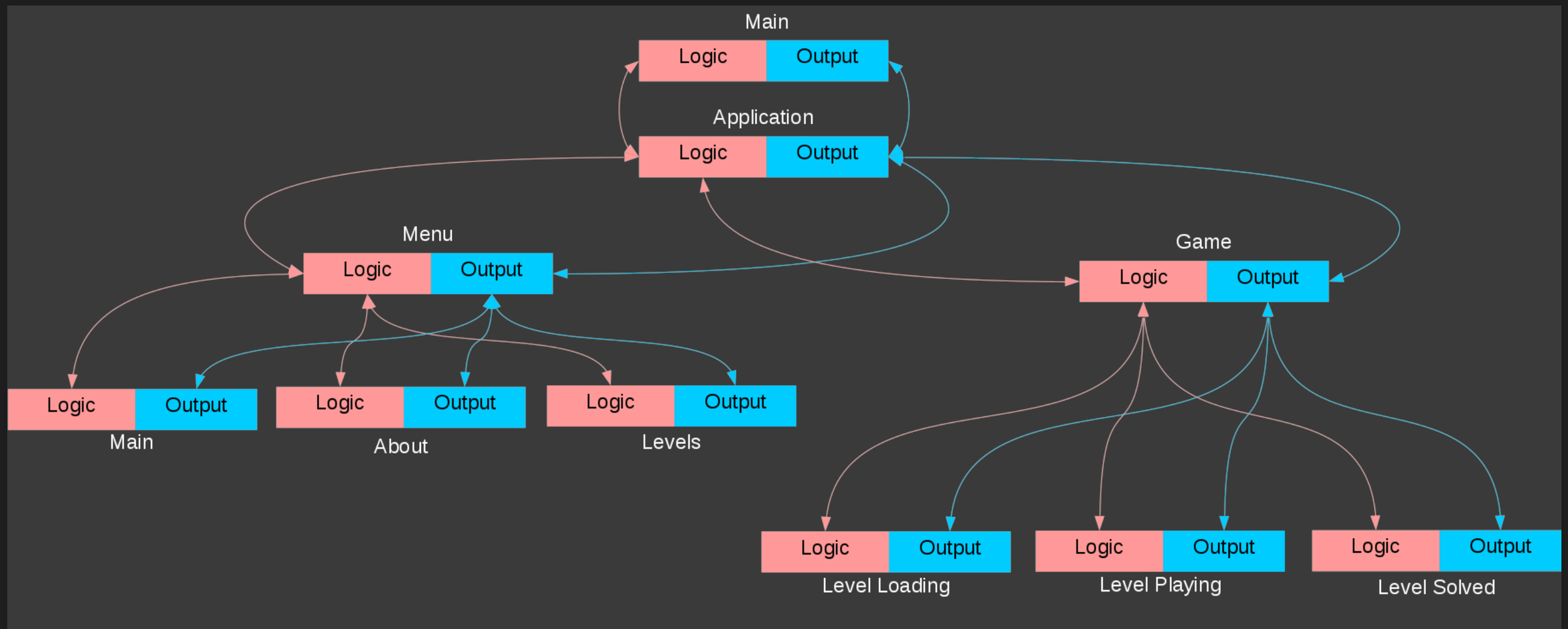


- Structure
- Shared code


Main
Logic
Output
Application
Logic
Output
Menu
Logic
Output
Game
Logic
Output
Logic
Output
Main
Logic
Output
About
Logic
Output
Levels
Logic
Output
Level Loading
Logic
Output
Level Playing
Logic
Output
Level Solved



- global vs local Controller



- global vs local Controller
- global vs local Resource Manager

FAWN

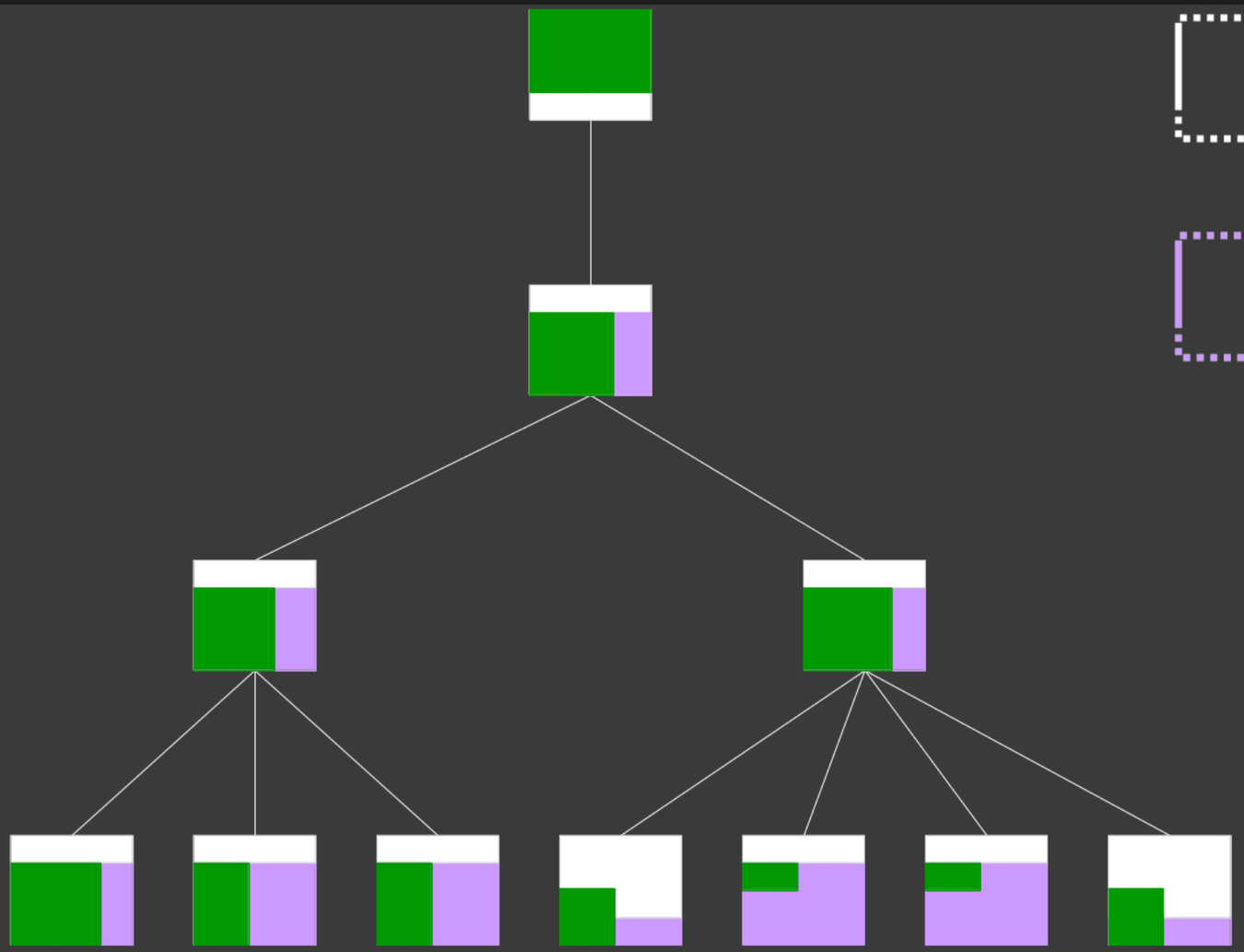
- fawn-android
- fawn-app-context
- fawn-app-debugging
- fawn-app-preferences
- fawn-app-settings
- fawn-ios

- fawn-render
- fawn-resourcemanager
- fawn-audio-sdl*
- fawn-render-sdl*
- fawn-resourcemanager-sdl*
- fawn-visual-sdl*

- fawn-app-mobilegame
- fawn-data-fps-counter
- fawn-menus
- fawn-physics-2d
- fawn-playground
- fawn-clock-html/sdl*
- fawn-input
- fawn-hardware-kinect-closest

- fawn-data-extra
- fawn-data-identity-list
- fawn-yampa-extra

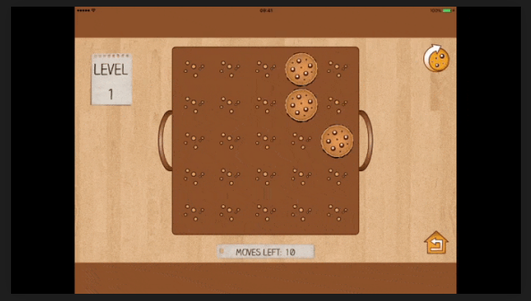
- fawn-...



Templates

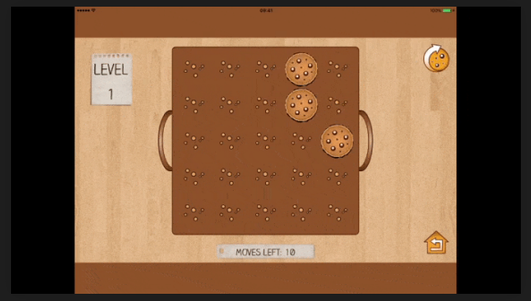
Individual Code

Using FAWN



- before over 2400 lines of code (LOC)
- after about 990 LOC
- core 400 LOC
- Alignment of visual elements

Using FAWN



- before over 2400 lines of code (LOC)
- after about 990 LOC
- core 400 LOC
- Alignment of visual elements

- over 2800 LOC
- core less than 1600 LOC
- not full use of FAWN
- Alignment of visual elements over 630 LOC

- Purity and Separation of Logic from IO
- Sizes of visual elements

- Purity and Separation of Logic from IO
    - Sizes of visual elements
- Compositional Application Architecture
    - global vs local Controller and Resource Manager
    - Alignment of visual elements

Mobile Game Programming in Haskell

Christina Zeller
Keera Studios Ltd
United Kingdom
christina.zeller@keera.co.uk

Ivan Perez
Keera Studios Ltd
United Kingdom
ivan.perez@keera.co.uk

Abstract

The use of pure functional languages for interactive applica-

1 Introduction

Functional programming has demonstrated, in multiple fields

- Purity and Separation of Logic from IO
  - Sizes of visual elements
- Compositional Application Architecture
  - global vs local Controller and Resource Manager
  - Alignment of visual elements
- Abstraction from Backend and Platform
- FAWN
- 6 and more awesome games

BACKUP

Abstraction from Backend and Platform

Input?	Backend?	Platform?
Keyboard, Kinect, Mouse, Touch, Wiimote, …	Browser, MacOS, Android, Windows, Linux, iOS, …	SDL1, SDL2, HTML

Input?	Backend?	Platform?
Keyboard, Kinect, Mouse, Touch, Wiimote, …	Browser, MacOS, Android, Windows, Linux, iOS, …	SDL1, SDL2, HTML

I don't care!

Select FAWN Package in Cabal File

```
import Game.Resource.Manager.SDL (loadImageSpec)

...

type ImageSpec = (FilePath, Maybe (Word8, Word8, Word8))
```

Select FAWN Package in Cabal File

```
import Game.Resource.Manager.SDL (loadImageSpec)

...

type ImageSpec = (FilePath, Maybe (Word8, Word8, Word8))
```

```
app.cabal:

if flag(sdl1)
   build-depends: fawn-resourcemanager-sdl1
if flag(sdl2)
   build-depends: fawn-resourcemanager-sdl2
```

Select with CPP in a FAWN Module

```
fawn.cabal

if flag(android)
   cpp-options: -DANDROID
if flag(ios)
   cpp-options: -DIOS
```

Select with CPP in a FAWN Module

```
fawn.cabal

if flag(android)
   cpp-options: -DANDROID
if flag(ios)
   cpp-options: -DIOS
```

```
-- FAWN-Module: Application Logging.
module App.Log where
#ifdef ANDROID
-- Do it the android way!
appLog :: ...
#idfef IOS
-- Do it the ios way!
appLog :: ...
#else
-- Do it the desktop way!
appLog :: ...
#endif
```

Apps do not care about backend, platform and input device.

Except for:

- advanced and non-standard input devices
- inclusion/exclusion of buttons
- name of the Haskell 'main' function

Apps do not care about backend, platform and input device.

Except for:

- advanced and non-standard input devices
- inclusion/exclusion of buttons
- name of the Haskell 'main' function

Open question:

When to use which approach (cabal/CPP)?

Further Reading

- Ivan Perez and Henrik Nilsson. 2017. Testing and Debugging Functional Reactive Programming. Proc. ACM Program. Lang. 1, ICFP, Article 2 (Sept. 2017), 27 pages.
- Ivan Perez, Manuel Bärenz, and Henrik Nilsson. 2016. Functional Reactive Programming, refactored. In Proceedings of the 9th International Symposium on Haskell (Haskell 2016). ACM, New York, NY, USA, 33–44.