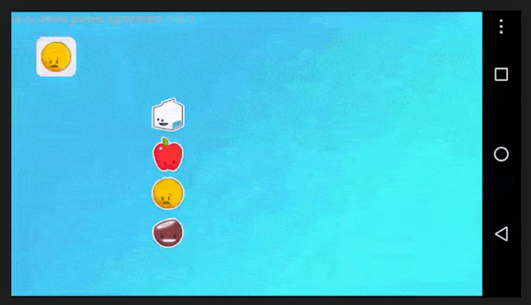
Cleaning the big picture

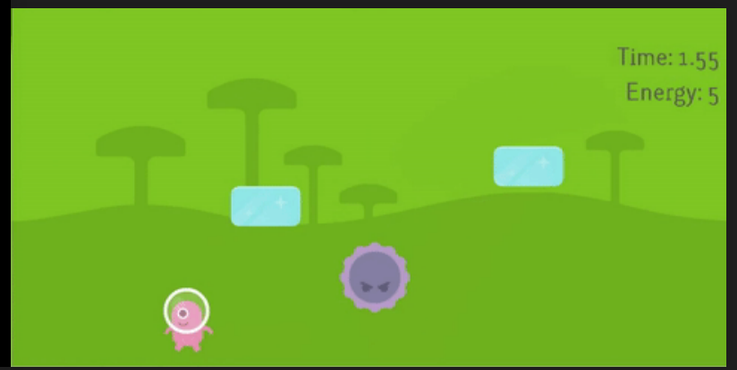
Creating maintainable mobile games in Haskell

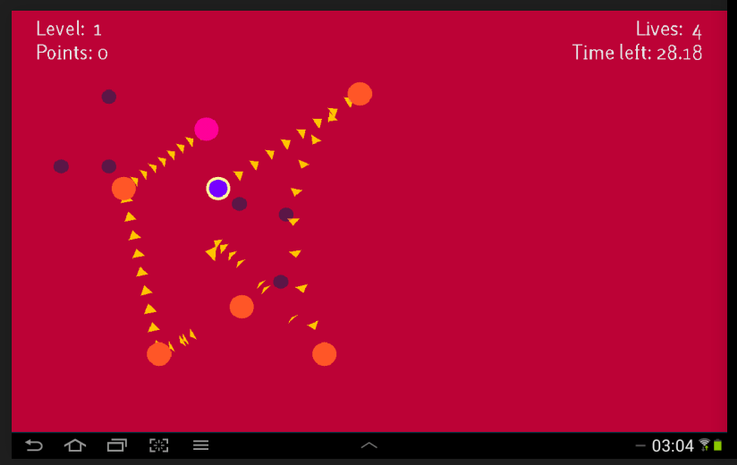
Christina Zeller

Keera Studios — keera@keera.co.uk



August - BOB 2019 (Berlin)





Focus of Cleaning

- definitions
- module as a whole
- across modules
- across libraries / applications

Focus of Cleaning

- **definitions**
- module as a whole
- across modules
- across libraries / applications

Compile and Run

```
{-# LANGUAGE FlexibleContexts #-}

f i = f' (\i -> i + 1)
  where
    f' f = sum . fmap f
```

```
test1 = print $ f [0..4]
```

Compile and Run

```haskell
{-# LANGUAGE FlexibleContexts #-}

f i = f' (\i -> i + 1)
  where
    f' f = sum . fmap f
```

```haskell
test1 = print $ f [0..4]
```

```
[...]
src/Main.hs:9:3: error:
    • No instance for (Show (t0 c0 -> c0))
        arising from a use of ‘print’
        (maybe you haven't applied a function to enough arguments?)
    • In a stmt of a 'do' block: print $ f [0 .. 4]
      In the expression:
        do { putStrLn "Start";
             putStrLn "test1";
             print $ f [0 .. 4];
             putStrLn "End" }
      In an equation for ‘main’:
          main
            = do { putStrLn "Start";
                   putStrLn "test1";
                   print $ f [0 .. 4];
                   .... }
[...]
cabal: Error: some packages failed to install:
example-0.1.0.0 failed during the building phase. The exception was:
ExitFailure 1
```

Compile and Run

```haskell
{-# LANGUAGE FlexibleContexts #-}

f i = f' (\i -> i + 1)
  where
    f' f = sum . fmap f
```

```haskell
test1 = print $ f [0..4]
```

```haskell
test2 = print $ f "abc" [0..4]
```

Compile and Run

```haskell
{-# LANGUAGE FlexibleContexts #-}

f i = f' (\i -> i + 1)
  where
    f' f = sum . fmap f
```

```haskell
test1 = print $ f [0..4]
```

```haskell
test2 = print $ f "abc" [0..4]
```

```
examples$ cabal install -ft2 && .cabal-sandbox/bin/example
[...]
15
[...]
```

GHC-Warnings

Cabal file

```
executable example
  main-is:             Main.hs
  hs-source-dirs:      src
  default-language:    Haskell2010
  build-depends:       base
  ghc-options:         -Wall
```

GHC-Warnings

Cabal file

```
executable example
  main-is:             Main.hs
  hs-source-dirs:      src
  default-language:    Haskell2010
  build-depends:       base
  ghc-options:         -Wall
```

Compiling?

GHC-Warnings

Cabal file

```
executable example
  main-is:             Main.hs
  hs-source-dirs:      src
  default-language:    Haskell2010
  build-depends:       base
  ghc-options:         -Wall
```

Compiling?

```
examples$ cabal install -ft2
Resolving dependencies...
Notice: installing into a sandbox located at
/home/examples/.cabal-sandbox
Configuring example-0.1.0.0...
Building example-0.1.0.0...
Installed example-0.1.0.0
```

```
examples$ touch src/* && cabal install -ft2 -j1
```

```
examples$ touch src/* && cabal install -ft2 -j1

[...]
Preprocessing executable 'example' for example-0.1.0.0...
[1 of 1] Compiling Main ( src/Main.hs, dist/dist-sandbox-64af14a3/build/example/example-tmp/Main.o )

src/Main.hs:13:19: warning: [-Wtype-defaults]
    • Defaulting the following constraints to type ‘Integer’
        (Enum a0)
          arising from the arithmetic sequence ‘0 .. 4’
          at src/Main.hs:13:19-24
        (Num a0) arising from a use of ‘f’ at src/Main.hs:13:11-24
        (Show a0) arising from a use of ‘print’ at src/Main.hs:13:3-24
    • In the second argument of ‘f’, namely ‘[0 .. 4]’
      In the second argument of ‘($)’, namely ‘f "abc" [0 .. 4]’
      In a stmt of a 'do' block: print $ f "abc" [0 .. 4]

src/Main.hs:21:1: warning: [-Wmissing-signatures]
    Top-level binding with no type signature:
      f :: forall t c (t1 :: * -> *).
           (Foldable t1, Num c, Functor t1) =>
           t -> t1 c -> c

src/Main.hs:21:3: warning: [-Wunused-matches]
    Defined but not used: ‘i’

src/Main.hs:21:12: warning: [-Wname-shadowing]
    This binding for ‘i’ shadows the existing binding
      bound at src/Main.hs:21:3
[...]
Installed example-0.1.0.0
```

Focus	Methods
Definitions	style guides, alignment, documentation, variable/function naming, signatures, level of abstraction / splitting functions, ghc-warnings, hlint

Focus of Cleaning

- definitions
- **module as a whole**
- across modules
- across libraries / applications

Explicit Imports

```haskell
{-# LANGUAGE FlexibleContexts #-}

import Magic
import MagicCarpet
import MAGO

f i = f' fly
  where
    f' f = magoMagic . magic f
```

Match the modules with the imports!

```
-- This module handles the rendering of the game state.
module AppParts.Game.Finished.DeviceOutput where
```

```
-- This module updates the game state.
module AppParts.Game.Finished.Logic where
```

Match the modules with the imports!

```haskell
-- This module handles the rendering of the game state.
module AppParts.Game.Finished.DeviceOutput where
```

```haskell
-- This module updates the game state.
module AppParts.Game.Finished.Logic where
```

```haskell
-- External imports
import App.Preferences          (Preferences)
import FRP.Yampa                (Event, SF)
import Playground               (AppPartOut, ExternalAction, clickTimeUpdate)

-- Internal imports
import AppParts.Game.Constants (finishedGameDelay, idFinishedG)
import AppParts.Game.State     (GameState)
import Resource.Manager        (RenderEnv)
import UserInput               (Controller)
```

Match the modules with the imports!

```haskell
-- This module handles the rendering of the game state.
module AppParts.Game.Finished.DeviceOutput where
```

```haskell
-- This module updates the game state.
module AppParts.Game.Finished.Logic where
```

```haskell
-- External imports
import App.Preferences          (Preferences)
import FRP.Yampa                (Event, SF)
import Playground               (AppPartOut, ExternalAction, clickTimeUpdate)

-- Internal imports
import AppParts.Game.Constants (finishedGameDelay, idFinishedG)
import AppParts.Game.State     (GameState)
import Resource.Manager        (RenderEnv)
import UserInput               (Controller)
```

```haskell
-- External imports
import App.Preferences     (Preferences)
import Game.VisualElem     (VisualElem)
import Graphics.UI.Collage (Collage)
import Playground          (displayWithBGColor)

-- Internal imports
import AppParts.Game.State (GameState)
import Resource.Manager    (RenderEnv, ResourceId (IdColorBg, IdSFinished))
import SAGE.Extra          (bgClgVEfromBgId)
```

Is there something odd here?

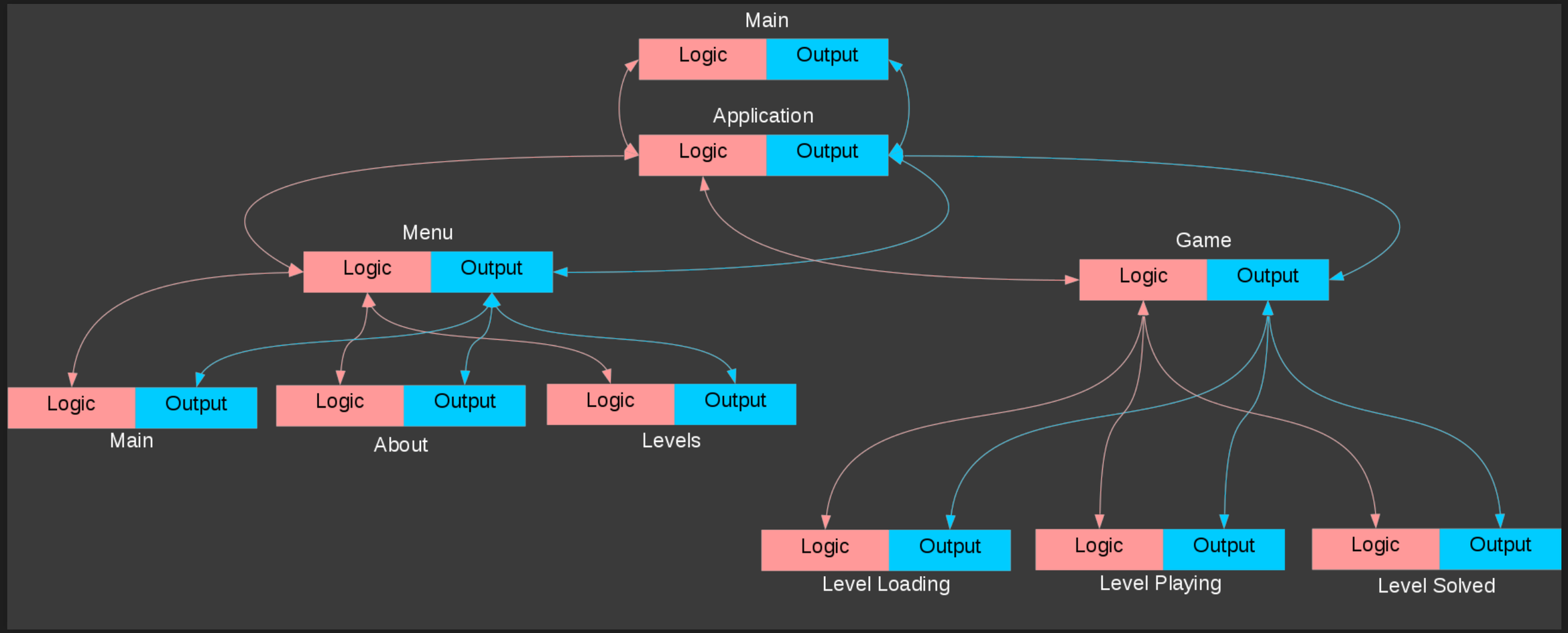
```haskell
-- This module handles the rendering of the game state.
module AppParts.Game.Finished.DeviceOutput where

-- External imports
import App.Preferences     (Preferences)
import Game.VisualElem     (VisualElem)
import Graphics.UI.Collage (Collage)
import Playground          (displayWithBGColor)

-- Internal imports
import AppParts.Game.State (GameState)
import Resource.Manager    (RenderEnv, ResourceId (IdColorBg, IdSFinished))
import SAGE.Extra          (bgClgVEfromBgId)
```

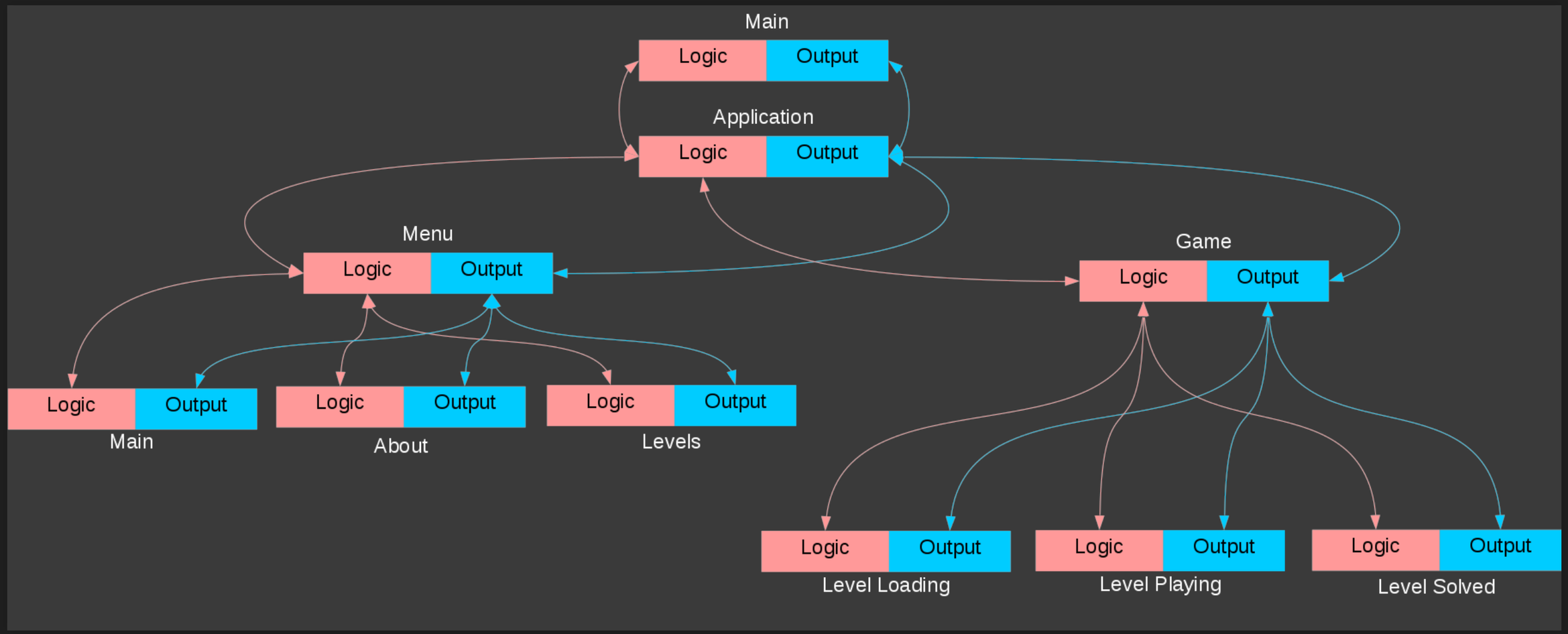
```haskell
-- This module updates the game state.
module AppParts.Game.Finished.Logic where

-- External imports
import App.Preferences          (Preferences)
import FRP.Yampa                (Event, SF)
import Playground               (AppPartOut, ExternalAction, clickTimeUpdate)

-- Internal imports
import AppParts.Game.Constants (finishedGameDelay, idFinishedG)
import AppParts.Game.State     (GameState)
import Resource.Manager        (RenderEnv)
import UserInput               (Controller)
```

Focus	Methods
Definitions	style guides, alignment, documentation, variable/function naming, signatures, level of abstraction / splitting functions, ghc-warnings, hlint
Whole Module	explicit imports, renaming, restructuring, haddock documentation

Focus of Cleaning

- definitions
- module as a whole
- **across modules**
- across libraries / applications


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



In your projects, where do you have

- a similar structure?
- repeated structures?

AppParts.Game.Loading.DeviceOutput

Safe Haskell None
Language Haskell2010

The device output handles the rendering of the level loading state.

Contents
- Rendering of the level loading state
  - Visual
- Game specific elements
  - Visual

Rendering of the level loading state

```
render :: (Preferences, GameState) -> RenderEnv -> IO ()
```

Visual

```
display :: (Preferences, GameState) -> RenderEnv -> IO ()
```

Game specific elements

Visual

```
completeVisual :: (Preferences, GameState) -> RenderEnv -> IO (ResourceId, Collage (VisualElem ResourceId) (Int, Int))
```

Combination of background color and a collage with everything that should be rendered.

PRE: The game state must be in 'GameLoading i' status.

AppParts.Game.Finished.DeviceOutput

Safe Haskell None
Language Haskell2010

The device output handles the rendering of the finished game state.

Contents

Synopsis

Rendering of the finished game state

```
render :: (Preferences, GameState) -> RenderEnv -> IO ()
```

Visual

```
display :: (Preferences, GameState) -> RenderEnv -> IO ()
```

Game specific elements

Visual

```
completeVisual :: (Preferences, GameState) -> RenderEnv -> IO (ResourceId, Collage (VisualElem ResourceId) (Int, Int))
```

Combination of background color and a collage with everything that should be rendered.

AppParts.Menus.Main.DeviceOutput

Safe Haskell None
Language Haskell2010

Synopsis

Contents
- Rendering of the main menu
  - Visual
- Game specific elements
  - Visual

The device output handles the rendering of the main menu.

Rendering of the main menu

```
render :: (Preferences, MenuState) -> RenderEnv -> IO ()
```

Visual

```
display :: (Preferences, MenuState) -> RenderEnv -> IO ()
```

Game specific elements

Visual

```
completeVisual :: (Preferences, MenuState) -> RenderEnv -> IO (ResourceId, Collage (VisualElem ResourceId) WidgetPos)
```

Combination of background color and a collage with everything that should be rendered.

```
completeWClg :: (Preferences, MenuState) -> RenderEnv -> Collage (Widget (VisualElem ResourceId)) WidgetPos
```

A collage with all widgets.

Note: Make sure that this really contains all widgets and call this function from the logic.

AppParts.Menus.Levels.DeviceOutput

Safe Haskell None
Language Haskell2010

The device output handles the rendering of the levels menu.

Contents
- Rendering of the level menu
  - Visual
- Game specific elements
  - Visual
- Auxiliary functions
  - Sizes
  - Pages and their Thumbnails

Rendering of the level menu

```
render :: (Preferences, MenuState) -> RenderEnv -> IO ()
```

Visual

```
display :: (Preferences, MenuState) -> RenderEnv -> IO ()
```

Game specific elements

Visual

```
completeVisual :: (Preferences, MenuState) -> RenderEnv -> IO (ResourceId, Collage (VisualElem ResourceId) (Int, Int))
```

Combination of background color and a collage with everything that should be rendered.

```
completeWClg :: (Preferences, MenuState) -> RenderEnv -> Collage (Widget (VisualElem ResourceId)) WidgetPos
```

A collage with all widgets.

Note: Make sure that this really contains all widgets and call this function from the logic.

PRE: The menu page needs to be a valid number. That is, a page that contains at least one level.

Auxiliary functions

Sizes

```
thumbBSize :: Num a => (a, a)
```

The size of the thumbnail board (without the header).

AppParts.Game.Loading.Logic

Safe Haskell None
Language Haskell2010

Synopsis

This module defines the game as a big Signal Function that transforms a Signal carrying an Input `Controller` information into a Signal carrying `GameState`.

Adapted from sage haskanoid/GamePlay.hs.

Documentation

loadLevel :: SF ((Controller, RenderEnv), (Preferences, GameState)) ((Preferences, GameState), Event AppPartOut, [ExternalAction]) #

Signal function that waits `levelLoadingDelay` seconds, then loads the level and request of updating the game status.

PRE: The game state must be in 'GameLoading i' status. PRE: `i` must be valid. That is, the level needs to exist in `levels`.

loadLevel' #

```
:: ((Controller, RenderEnv), (Preferences, GameState))
-> ((Controller, RenderEnv), (Preferences, GameState))
```

New game state where the level is loaded.

Generate a game state where the level is loaded and request an update of the game status.

PRE: The game state must be in 'GameLoading i' status. PRE: `i` must be valid. That is, the level needs to exist in `levels`.

AppParts.Game.Finished.Logic

Safe Haskell None
Language Haskell2010

Synopsis

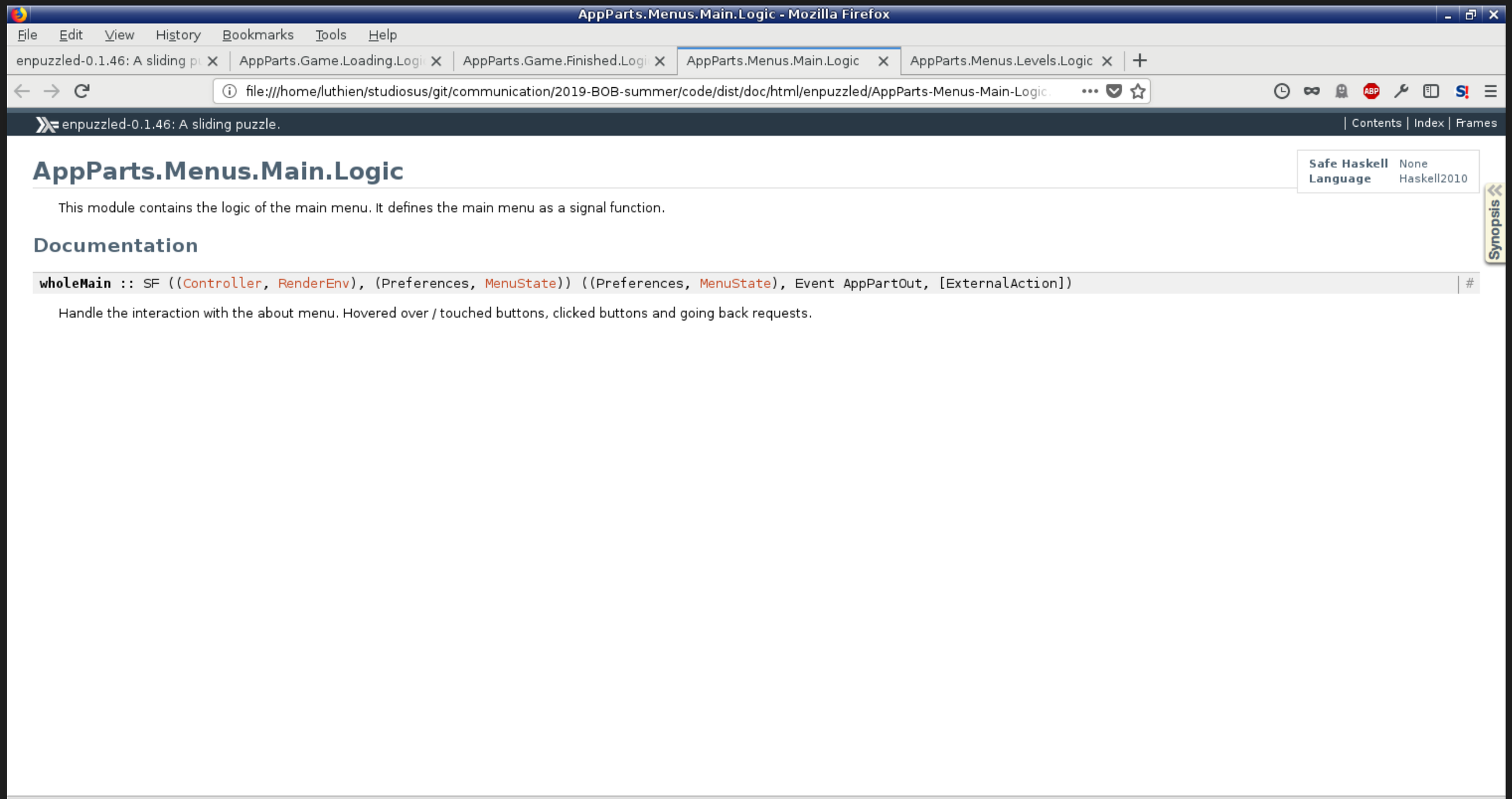
This module defines the game as a big Signal Function that transforms a Signal carrying an Input `Controller` information into a Signal carrying `GameState`.

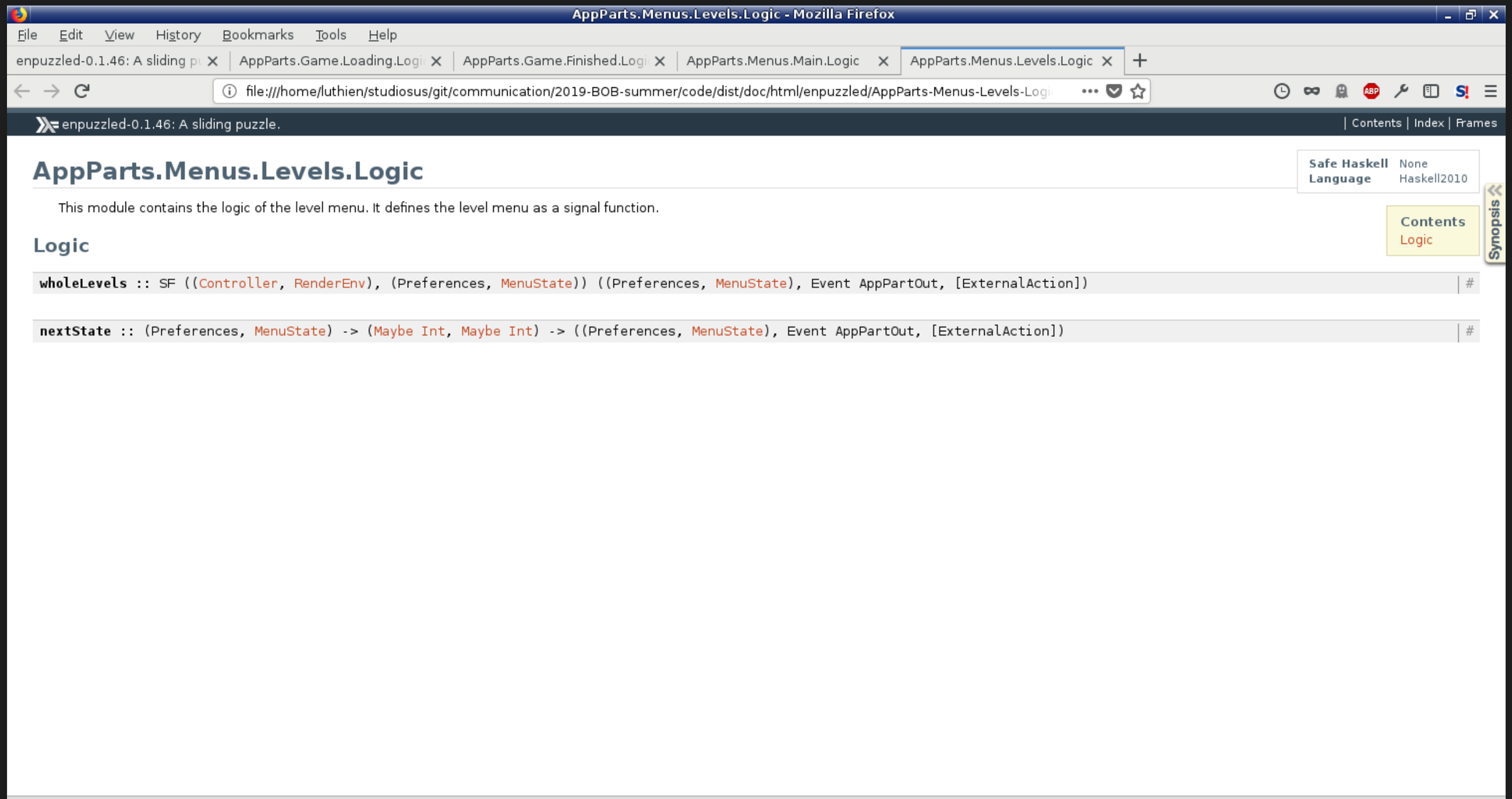
Adapted from sage haskanoid/GamePlay.hs.

Documentation

```
finishedGame :: SF ((Controller, RenderEnv), (Preferences, GameState)) ((Preferences, GameState), Event AppPartOut, [ExternalAction])
```

Signal function for requesting to changing the game state, after waiting `finishingGameDelay` seconds.





```
vi -d src/AppParts/Game/Loading/DeviceOutput.hs src/AppParts/Game/Finishing/DeviceOutput.hs
```

```
vi -d src/AppParts/Game/Loading/DeviceOutput.hs src/AppParts/Game/Finishing/DeviceOutput.hs
```

Terminal - DeviceOutput.hs (~/studiosus/git/communication/2019-BOB-summer/code/src/AppParts/Game/Loading) (1 of 2) - VIM

File Edit View Terminal Tabs Help

```
{-# LANGUAGE PatternGuards        #-}
{-# LANGUAGE TypeFamilies         #-}
{-# LANGUAGE TypeSynonymInstances #-}
-- | The device output handles the rendering of the level loading state.
module AppParts.Game.Loading.DeviceOutput where

-- External imports
import App.Preferences     (Preferences)
import Game.VisualElem     (VisualElem (VisualText))
import Graphics.UI.Align   (Align (Align), HAlign (HCenter), VAlign (VCenter))
import Graphics.UI.Collage (Collage (CollageItem), collageMapM)
import Playground          (displayWithBGColor)
import Playground.SDL      (dAlignToAbsPos')

-- Internal imports
import AppParts.Game.Levels (bgColor, levelInfo, levelName, levels, mBgImage)
import AppParts.Game.State  (GameInfo (gameStatus), GameState (gameInfo),
                             GameStatus (GameLoading))
import Resource.Manager     (RenderEnv, ResourceId (IdColorFont, IdFont))
import SAGE.Extra           (bgClgVEfromMaybeBgId)

-- * Rendering of the level loading state

render :: (Preferences, GameState) -> RenderEnv -> IO ()
render = display

-- ** Visual

display :: (Preferences, GameState) -> RenderEnv -> IO ()
display ctxt env = do
  cV <- completeVisual ctxt env
  displayWithBGColor cV env True

-- * Game specific elements

-- ** Visual

-- | Combination of background color and a collage with everything that should
-- be rendered.
--
-- PRE: The game state must be in 'GameLoading i' status.
completeVisual :: (Preferences, GameState) -> RenderEnv
               -> IO (ResourceId, Collage (VisualElem ResourceId) (Int, Int))
completeVisual (_prefs, gs) env = do
  clg' <- collageMapM (dAlignToAbsPos' env) clg
  return (bgColor lvlSpec, mappend bgClg clg')
  where
    bgClg = bgClgVEfromMaybeBgId (mBgImage lvlSpec)

    clg = CollageItem (VisualText IdFont IdColorFont txt) ((0,0), Align HCenter VCenter)
    txt = "LOADING LEVEL " ++ levelName (levelInfo lvlSpec)
```

src/AppParts/Game/Loading/DeviceOutput.hs 54,1 30%

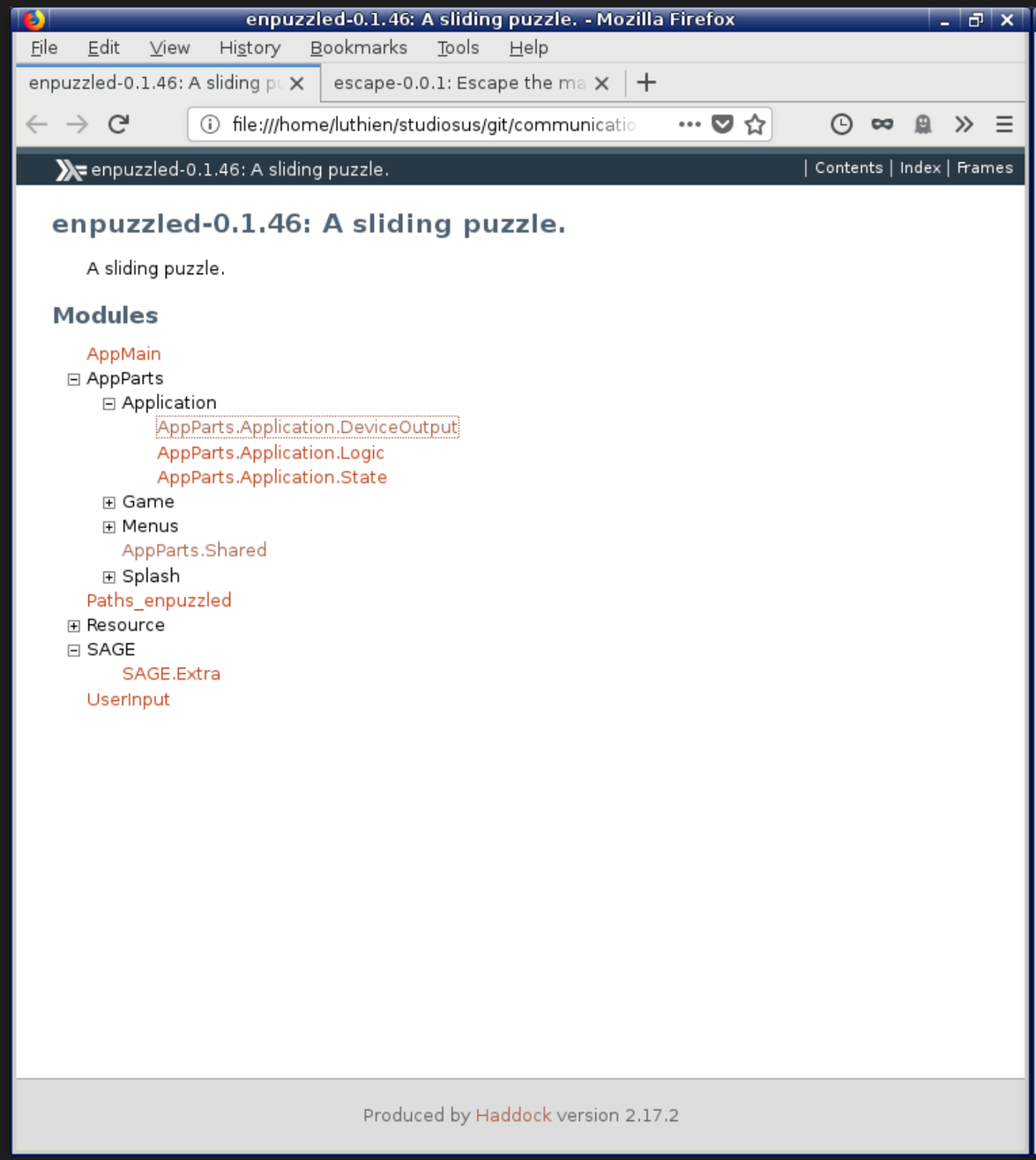
```
{-# LANGUAGE TypeFamilies         #-}
{-# LANGUAGE TypeSynonymInstances #-}
-- | The device output handles the rendering of the finished game state.
module AppParts.Game.Finished.DeviceOutput where

-- External imports
import App.Preferences     (Preferences)
import Game.VisualElem     (VisualElem)
import Graphics.UI.Collage (Collage)

import Playground          (displayWithBGColor)

-- Internal imports
import AppParts.Game.State (GameState)
import Resource.Manager    (RenderEnv, ResourceId (IdColorBg, IdSFinished))
import SAGE.Extra          (bgClgVEfromBgId)

-- * Rendering of the finished game state

render :: (Preferences, GameState) -> RenderEnv -> IO ()
render = display

-- ** Visual

display :: (Preferences, GameState) -> RenderEnv -> IO ()
display ctxt env = do
  cV <- completeVisual ctxt env
  displayWithBGColor cV env True

-- * Game specific elements

-- ** Visual

-- | Combination of background color and a collage with everything that should
-- be rendered.
completeVisual :: (Preferences, GameState) -> RenderEnv
               -> IO (ResourceId, Collage (VisualElem ResourceId) (Int, Int))
completeVisual _ctxt _env = return (IdColorBg, bgClgVEfromBgId IdSFinished)
```

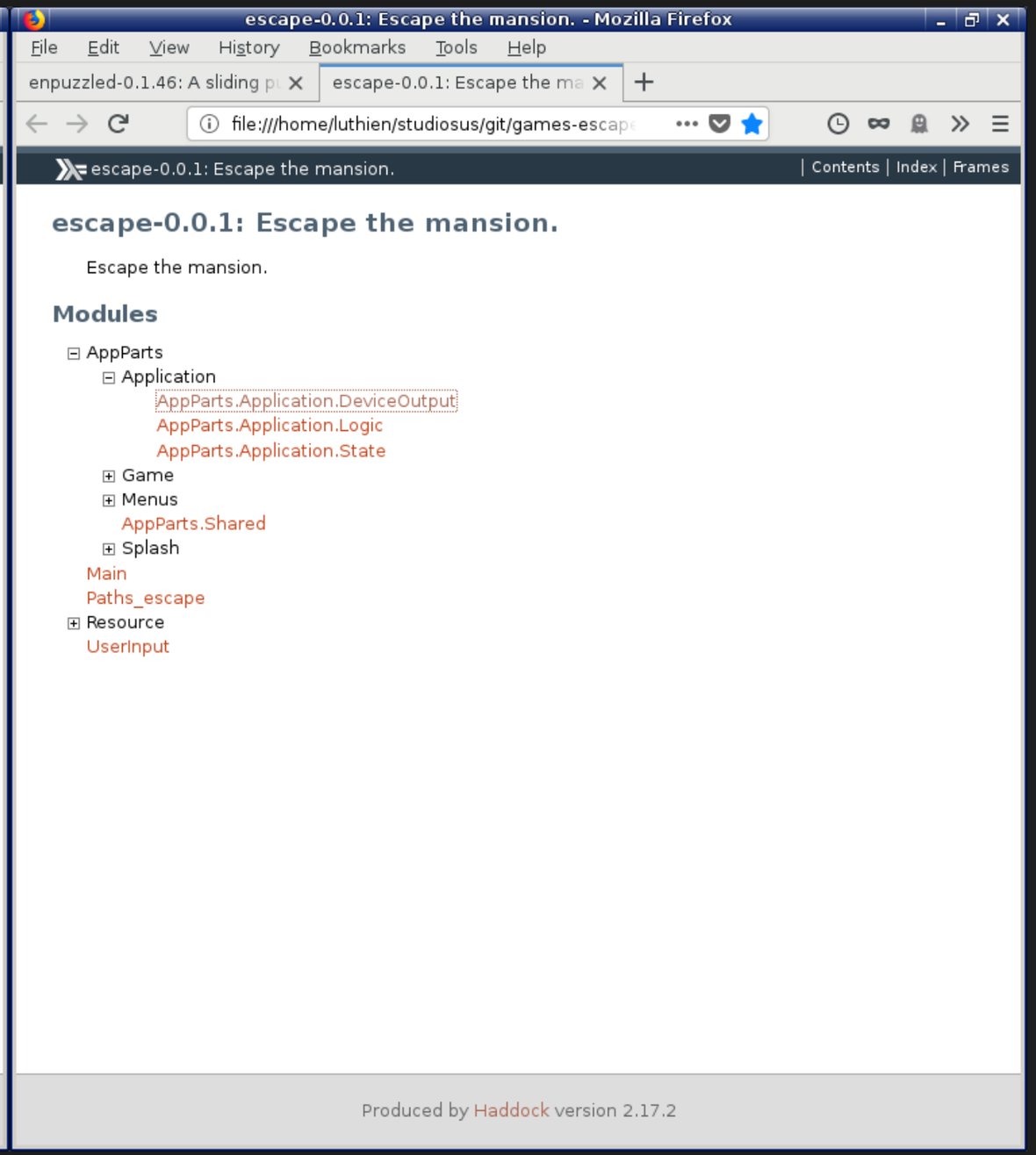
src/AppParts/Game/Finished/DeviceOutput.hs 40,1 Bot

Focus	Methods
Definitions	style guides, alignment, documentation, variable/function naming, signatures, level of abstraction / splitting functions, ghc-warnings, hlint
Whole Module	explicit imports, renaming, restructuring, haddock documentation
Across Modules	renaming, restructuring, haddock documentation, vi -d, diff -rq, abstraction, Shared-Module, Library.Extra, libraries, templates

Focus of Cleaning

- definitions
- module as a whole
- across modules
- **across libraries / applications**





```
vi -d enpuzzled/src/AppParts/Menus/Main/DeviceOutput.hs escape/src/AppParts/Menus/Main/DeviceOutput.hs
```

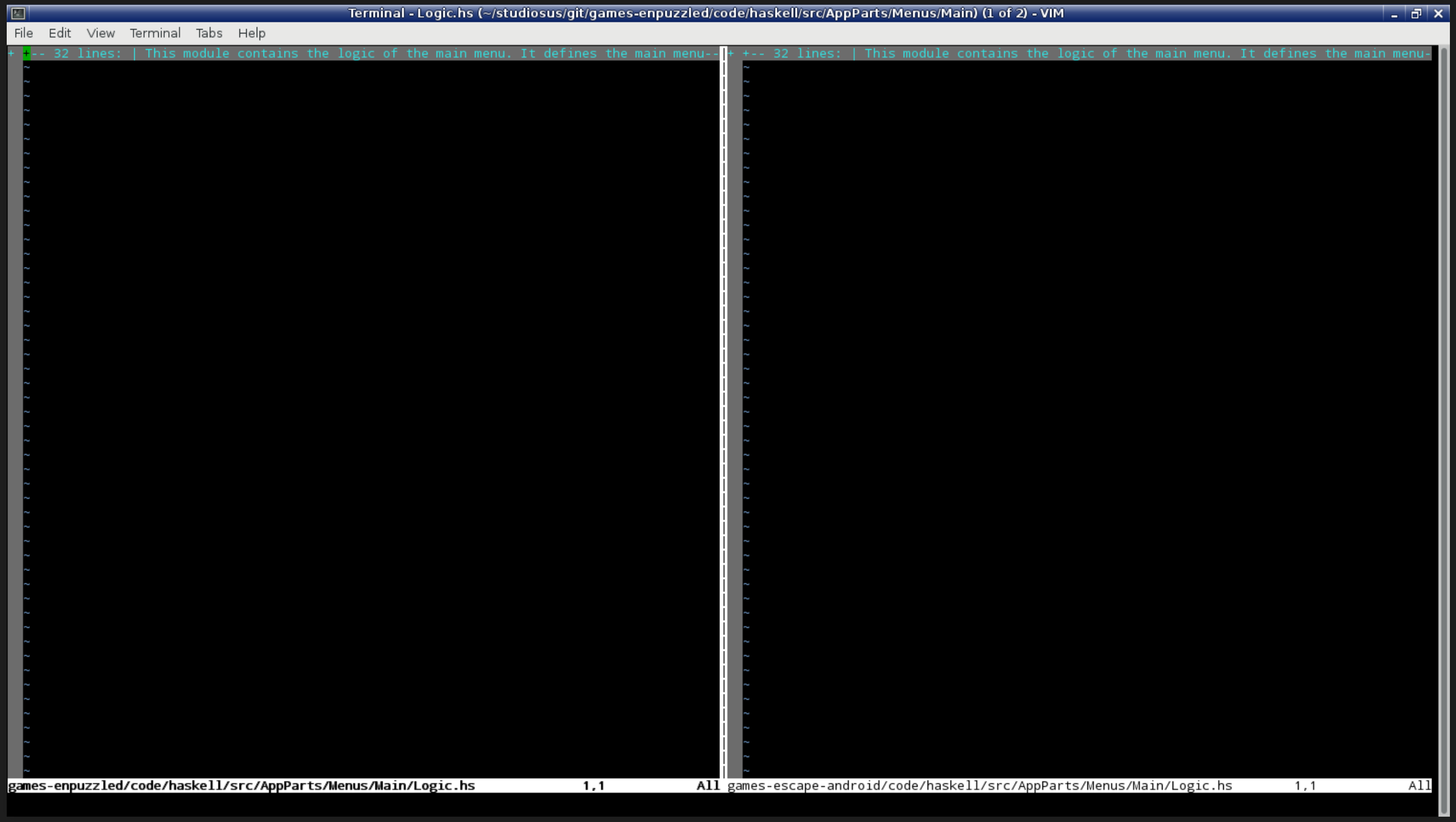
Terminal - DeviceOutput.hs + (~/studiosus/git/games-enpuzzled/code/haskell/src/AppParts/Menus/Main) (1 of 2) - VIM

File Edit View Terminal Tabs Help

```
games-enpuzzled/code/haskell/src/AppParts/Menus/Main/DeviceOutput.hs [+]  1,1  All
```

```
<ames-escape-android/code/haskell/src/AppParts/Menus/Main/DeviceOutput.hs 1,1  All
```

```
vi -d enpuzzled/src/AppParts/Menus/Main/Logic.hs escape/src/AppParts/Menus/Main/Logic.hs
```



```
$ find enpuzzled/src/AppParts/Menus/ -type f
enpuzzled/src/AppParts/Menus/Main/DeviceOutput.hs
enpuzzled/src/AppParts/Menus/Main/Constants.hs
enpuzzled/src/AppParts/Menus/Main/Logic.hs
enpuzzled/src/AppParts/Menus/Levels/DeviceOutput.hs
enpuzzled/src/AppParts/Menus/Levels/Constants.hs
enpuzzled/src/AppParts/Menus/Levels/Logic.hs
enpuzzled/src/AppParts/Menus/About/DeviceOutput.hs
enpuzzled/src/AppParts/Menus/About/Constants.hs
enpuzzled/src/AppParts/Menus/About/Logic.hs
enpuzzled/src/AppParts/Menus/DeviceOutput.hs
enpuzzled/src/AppParts/Menus/Constants.hs
enpuzzled/src/AppParts/Menus/State.hs
enpuzzled/src/AppParts/Menus/Logic.hs
```

```
$ find enpuzzled/src/AppParts/Menus/ -type f
enpuzzled/src/AppParts/Menus/Main/DeviceOutput.hs
enpuzzled/src/AppParts/Menus/Main/Constants.hs
enpuzzled/src/AppParts/Menus/Main/Logic.hs
enpuzzled/src/AppParts/Menus/Levels/DeviceOutput.hs
enpuzzled/src/AppParts/Menus/Levels/Constants.hs
enpuzzled/src/AppParts/Menus/Levels/Logic.hs
enpuzzled/src/AppParts/Menus/About/DeviceOutput.hs
enpuzzled/src/AppParts/Menus/About/Constants.hs
enpuzzled/src/AppParts/Menus/About/Logic.hs
enpuzzled/src/AppParts/Menus/DeviceOutput.hs
enpuzzled/src/AppParts/Menus/Constants.hs
enpuzzled/src/AppParts/Menus/State.hs
enpuzzled/src/AppParts/Menus/Logic.hs

$ diff -rq enpuzzled/src/AppParts/Menus/ escape/src/AppParts/Menus/
Files enpuzzled/src/AppParts/Menus/Main/DeviceOutput.hs and escape/src/AppParts/Menus/Main/DeviceOutput.hs differ
Files enpuzzled/src/AppParts/Menus/Main/Constants.hs and escape/src/AppParts/Menus/Main/Constants.hs differ
Files enpuzzled/src/AppParts/Menus/Levels/DeviceOutput.hs and escape/src/AppParts/Menus/Levels/DeviceOutput.hs dif
Files enpuzzled/src/AppParts/Menus/Levels/Constants.hs and escape/src/AppParts/Menus/Levels/Constants.hs differ
Files enpuzzled/src/AppParts/Menus/About/DeviceOutput.hs and escape/src/AppParts/Menus/About/DeviceOutput.hs diffe
Files enpuzzled/src/AppParts/Menus/State.hs and escape/src/AppParts/Menus/State.hs differ
```

Focus	Methods
Definitions	style guides, alignment, documentation, variable/function naming, signatures, level of abstraction / splitting functions, ghc-warnings, hlint
Whole Module	explicit imports, renaming, restructuring, haddock documentation
Across Modules	renaming, restructuring, haddock documentation, vi -d, diff -rq, abstraction, Shared-Module, Library.Extra, libraries, templates
Across Apps	renaming, restructuring, haddock documentation, vi -d, diff -rq, abstraction, libraries, templates

Additional comments

- Iterate

Additional comments

- Iterate
- Discuss your code

Additional comments

- Iterate
- Discuss your code
- Let others 'clean'

Quiz

- When is it worth to clean?

Quiz

- When is it worth to clean?
- How much effort does cleaning request?

Quiz

- When is it worth to clean?
- How much effort does cleaning request?
- What methods can be applied?

Quiz

- When is it worth to clean?
- How much effort does cleaning request?
- What methods can be applied?
- How can you convince others that cleaning is important?

Quiz

- When is it worth to clean?
- How much effort does cleaning request?
- What methods can be applied?
- How can you convince others that cleaning is important?
- What is more fun than working with clean code?