Biblioteca WXHaskell

WxHaskell

- Biblioteca para construção de interfaces gráficas
  - Portátil
  - Derivada de WxWidgets
- Funciona no GHC
  - Utiliza o monad IO para modelagem de estado
  - Todas as operações são definidas como IO a

Preparação do Ambiente

- Se o computador de vocês não possuir o ghc (Glasgow Haskell Compiler) instalado:
  - Baixem o arquivo:
    - http://www.dsc.upe.br/~lcsm/ghc.zip
  - Descompactem o arquivo no diretório:
    - c:\ghc
  - Adicionem a seguinte entrada na variável PATH:
    - c:\ghc\ghc-6.2.2\bin;c:\ghc\wxhaskell-0.9.4\lib
  - Entrem no diretório:
    - c:\ghc\wxhaskell-0.9.2\bin
    - Executem o arquivo: wxhaskell-register.bat

Conceitos Básicos

Variaveis Imperativas

- Modela o conceito de variáveis que podem ter o seu valor consultado e alterado.
- Operações:
  - **type Var a = ...**
    - **Define o tipo variável que armazena um valor do tipo "a"**
  - **varCreate** :: a -> IO (Var a)
    - Cria uma nova variável a partir do seu valor inicial
  - **varGet** :: Var a -> IO a
    - Recupera o valor de uma variável
  - **varSet** :: Var a -> a -> IO ()
    - Modifica o valor de uma variável
  - **varUpdate** :: Var a -> (a -> a) -> IO a
    - Atualiza o valor de uma variável a partir de uma função.

Exemplo de Variáveis

```
teste = do
     x <- varCreate 10
     v0 <- varGet x
     print v0
     varSet x 20
     v1 <- varGet x
     print v1
     varUpdate x (\v -> v+1)
     v2 <- varGet x
     print v2
```

Atributos e Propriedades

- Atributos
  - Descrevem uma característica de um objeto WxHaskell
  - Tipo: **Attr o t**
    - Atributo de objetos "o" que armazena informação do tipo t
    - **text :: Attr (Window a) String**
- Propriedade:
  - Associação entre um atributo e valor.
  - Tipo: **Prop o**
    - Propriedade do Objeto O
    - Ex.: **Prop (Window a)**
  - Operador ":=" associa um attributo a um valor constante
    - (:=) :: Attr a t → t → Prop a
    - Exemplo: **text := "Press"**
  - Operador ":~" associa um attributo a um valor obtido a partir de uma função de atualização
    - (:~) :: Attr a t → (t → t) → Prop a
    - Exemplo: **text :~ (\s → "a"++s)**

Atributos e Propriedades

- Objetos do WxHaskell são definidos através uma lista de propriedades
- Funções auxiliares:
  - set :: a → [Prop a] → IO ()
  - get :: a → Attr a t → IO t

Eventos

- Identifica uma ocorrência na aplicação
  - Botão pressionado, movimento do mouse, etc..
- Tipo:

  Event w t: Evento que acontece em objetos do tipo "w" que é tratado por um tratador do tipo "t"

  Exemplo:

  anyKey :: Event (Window a) (Key → IO ())

  click :: Event (Window a) (Point → IO ())

  closing :: Event (Window a) (IO ())

  commanding :: Event (Button a) (IO ())

Eventos (cont)

- Função "on"
  - Define um atributo a partir de um evento

    on :: Event w a → Attr w a
  - Utilizada para definir como um controle vai responder a um determinado evento.
- Exemplo:

```
set button [
   on commanding := do (print "botao pressionao")
]
```

Objetos Gráficos

Formato de um Programa

```
import Graphics.UI.WX

main = start gui

gui :: IO ()
gui = ....
```

- A função start:

  start :: IO () → IO ()
- Inicializa a interface gráfica definida pelo argumento e executa o laço principal.
- Todas as operações sobre interfaces gráficas estão encapsuladas no monad IO

Execução de um Programa

- Interpretador:
  > ghci –package wc programa.hs
- Compilador:
  > ghc –package wc programa.hs

Janelas

Janela Funções

- Criação:
  - **frame** :: [Prop (Frame ())] -> IO (Frame ())
    - Cria a janela principal do programa
  - **frameFixed** :: [Prop (Frame ())] -> IO (Frame ())
    - Cria a janela principal do programa com tamanho fixo
- Funções úteis:
  - close j == Fecha a janela j
  - repaint j == Redesenha a janela j

Algumas Propriedades da Janelas

- Título da janela
  - text :: Attr w String
- Tamanho da janela
  - size :: Attr w Point
  - clientSize :: Attr w Point
- A janela é visível
  - visible :: Attr w Bool
- Cor de Fundo
  - bgcolor :: Attr w Color

Exemplo

```
import Graphics.UI.WX

gtest = do
  main <- frame [
     text := "Teste de Janela",
     size := Size 500 300,
     bgcolor := rgb 0 255 0
     ]
  return ()
```

Alguns Eventos de Janela

- Uma tecla foi pressionada:
  - **anyKey** :: Reactive w => Event w (Key -> IO ())
- Ações do Mouse:
  - **motion** :: Reactive w => Event w (Point -> IO ())
  - **drag** :: Reactive w => Event w (Point -> IO ())
  - **click** :: Reactive w => Event w (Point -> IO ())
  - **unclick** :: Reactive w => Event w (Point -> IO ())
  - **doubleClick** :: Reactive w => Event w (Point -> IO ())
- A Janela precisa ser redesenhada:
  - **paint** :: Paint w => Event w (DC () -> Rect -> IO ())

Exemplo

```
import Graphics.UI.WX

gtest = do
  main <- frame [
     text := "Teste de Janela",
     size := Size 500 300,
     bgcolor := rgb 0 255 0,
     on click := (\p -> do
        putStr “mouse pressionado na pos: “
        print p
        return ()
        )
     ]
  return ()
```

Funções de Desenho

- Utilizadas na para desenhar na janela
  - circle :: DC a -> Point -> Int -> [Prop (DC a)] -> IO ()
  - arc :: DC a -> Point -> Int -> Double -> Double -> [Prop (DC a)] -> IO ()
  - ellipse :: DC a -> Rect -> [Prop (DC a)] -> IO ()
  - ellipticArc :: DC a -> Rect -> Double -> Double -> [Prop (DC a)] -> IO ()
  - line :: DC a -> Point -> Point -> [Prop (DC a)] -> IO ()
  - polyline :: DC a -> [Point] -> [Prop (DC a)] -> IO ()
  - polygon :: DC a -> [Point] -> [Prop (DC a)] -> IO ()
  - drawPoint :: DC a -> Point -> [Prop (DC a)] -> IO ()
  - drawRect :: DC a -> Rect -> [Prop (DC a)] -> IO ()
  - roundedRect :: DC a -> Rect -> Double -> [Prop (DC a)] -> IO ()
  - drawText :: DC a -> String -> Point -> [Prop (DC a)] -> IO ()

Exemplo

```
import Graphics.UI.WX

gtest = do
  main <- frame [
     text := "Teste de Janela",
     size := Size 500 300,
     bgcolor := rgb 0 255 0,
     on paint := (\dc rect -> do
        drawText dc "Texto" (Point 100 100) []
        circle dc (Point 400 200) 50 []
       return ()
        )
     ]
  return ()
```

Controles

Propriedades Comuns aos Controles

- Todos os controles possuem:
  - **position** :: Attr w Point
  - **area** :: Attr w Rect
  - **clientSize** :: Attr w Size
  - **virtualSize** :: Attr w Size

Botões

- Criação:
  - **button** :: Window a -> [Prop (Button ())] -> IO (Button ())
- Alguns Atributos:
  - Título do Botão:
    - text :: Attr b String
- Eventos:
  - Botão pressionado:
    - **command** :: Commanding w => Event w (IO ())

Textos Estáticos

- Construtor:
  - **staticText** :: Window a ->
    [Prop (StaticText ())] -> IO (StaticText ())
- Propriedade:
  - text :: Attr b String

Exemplo

```
btest = do
   main <- frame [ text := "Teste de Texto",
           size := Size 450 300 ]
   staText <- staticText main [ text := "Hello",
           position := Point 20 100 ]
   bYes <- button main [ text := “Press Me",
           position := Point 10 20,
           on command := do
               set staText [
                  text :~ \s -> s ++ " World!!!"
               ]
           ]
   return ()
```

Caixas de Textos

- Caixas de Textos:
  - textEntry :: Window a -> [Prop (TextCtrl ())] -> IO (TextCtrl ())
    - Editor de 1 linha
  - textCtrl :: Window a -> [Prop (TextCtrl ())] -> IO (TextCtrl ())
    - Editor de várias linhas
- Propriedades:
  - **text** :: Attr w String
- Eventos:
  - ?????

Exemplo

```
etest = do
   main <- frame [ text := "Teste de Texto",
                   size := Size 150 100 ]
   e1 <- textEntry main [ position = Point 50 20]
   return ()
```

Choices

- Criação:
  - **choice :: Window a -> [Prop (Choice ())] -> IO (Choice ())**
  - **singleListBox :: Window a -> [Prop (... ())] -> IO (SingleListBox ())**
- Atributos:
  - **items** :: Attr w [a]
  - **item** :: Int -> Attr w a
  - **selection** :: Attr w Int
- Eventos:
  - **select** :: Selecting w => Event w (IO ())

Exemplo

```
etest = do
   main <- frame [ text := "Teste de Texto",
                   size := Size 150 100 ]
   c1 <- singleListBox main [ items := ["o1", "o2", "o3" ] ]
   set c1 [
             on select := do
                 pos <- get c1 selection
                 v <- get c1 (item pos)
                 putStr "Selecionado :"
                 print  v
          ]
   return ()
```