Haskross - Puzzle inspirado em Picross usando a biblioteca GTK2HS em Haskell

Felipe Leite Uematsu[1], **Melissa Lima**[1], **Vitor Costa Farias**[1]

[1]Centro de matemática, computação e cognição – Universidade Federal do ABC

{felipe.u, melissa.lima, vitor.costa}@ufabc.aluno.edu.br

1. Tutorial

Para a execução do Haskross, é necessário que a sua máquina deve ter instalado a biblioteca Gtk2H, o GHC e cabal. Para a instalação destes, siga os seguintes passos:

1.1. Instalando Gtk2Hs [Haskell 2019]

No terminal, execute o comando de instalação do Gtk2H de acordo sua respectiva distribuição de Linux:

- Debian/Ubuntu:

```
$ libghc-gtk-dev
```

- Fedora:

```
$ ghc-gtk-devel
```

- Gentoo:
  disponível no Haskell overlay como:

```
$ dev-haskell/gtk
```

- Arch:

```
$ gtk2hs-buildtools
```

1.2. Instalando cabal [Haskell.org 2019]

No terminal, execute o comando de instalação do de acordo com o gerenciador de pacotes da sua distribuição de Linux seguido de `caball-install`

1.3. Executando o Haskross

Após a instalação dos pacotes necessário, abra o ghci no terminal com o seguinte comando:

```
$ ghci
```

Feito isso, digite `:l Main.hs` e logo em seguida `main`. Uma outra forma é executar o comando `cabal run` dentro da pasta do projeto. Caso haja algum problema de versão, uma forma de resolver é recriar o projeto com o comando **cabal init**, seguido de **cabal build** e **cabal run**. Por fim podemos gerar o executável com o comando

```
$ ghc --make InputOutput.hs Defines.hs CheckFunctions.hs
Structures.hs MainMenu.hs Main.hs -o Haskross
```

e depois executando com o comando

```
$ ./Haskross
```

Para executar o arquivo de teste basta carregá-lo no **ghci** com o comando `:l Test.hs`

2. Processo de desenvolvimento

Tivemos bastante dificuldade para configurar o ambiente para poder rodar a biblioteca *gtk*. Não há tutoriais especificos de como usar o *gtk* junto com o *stack*. Ficamos muito tempo tentando acertar as configurações e dependências, porém não conseguimos achar documentação para os erros apresentados. Então mudamos para o *cabal*, que possui mais material para o uso do *gtk*.

Fizemos uma melhoria em relação às *labels* que ficam em torno da matriz de botões: a ideia era que essas informações fossem escritas em arquivos textos assim como a solução da fase, porém agora fizemos funções que geram essas informações automaticamente a partir da matriz de soluções.

Tinhamos a ideia de deixar apenas uma janela e ir modificando os objetos internos dela conforme fosse necessário, porém não obtivemos o comportamento esperado, o que nos obrigou a gerar uma nova janela para cada fase do jogo.

Para a checagem de fim de jogo primeiro pensamos em fazer uma matriz de *MVar* e mudar o estado apenas desse elemento e depois checar a matriz. Porém, não funcionou do jeito que gostariamos, o que nos fez mudar para uma matriz *MVar* com elementos *MVar*, porém resolvemos fazer de maneira mais simples, onde temos apenas uma matriz *MVar* e criamos uma nova matriz comum para colocarmos nessa *MVar*. Além disso pensamos em não usar *MVar*, onde sempre que um botão fosse clicado percorreriamos todo o tabuleiro conferindo com a solução porém não foi possível, pois ao criarmos a matriz de botões e depois atribuírmos suas funções elas não funcionavam.

Tivemos que criar alguma estruturas, pois quando atribuíamos os objetos na tabela criando variáveis locais esses objetos não apareciam, pois o objeto tabela do *gtk* é apenas para *layout*, então as referências para as variáveis locais eram perdidas.

Não foi possível fazer testes automatizados para todas as funções uma vez que muitas delas criam objetos visuais ou atribuem tais objetos em outros. Assim, muitos dos testes foram feitos de forma manual, como clicar nos botões e conferir se sua cor mudava ou não, jogar uma fase até o fim para ver se o fim de jogo era atingido, checar se os números guias do preenchimento estavam certos.

Não conseguimos tirar o efeito *hover* dos botões, então ao colocar o *mouse* sobre o botão ele ficará branco, mesmo dando um clique ele ainda ficará branco, quando o certo era ficar preto, porém é só retirar o *mouse* de cima do botão que a cor preta aparece. Pensamos em carregar uma imagem no botão para tentar driblar esse problema, mas mesmo assim as bordas ficavam brancas, pois a imagem não ocupava o botão por completo. Pensamos também em usar as imagens para representar o estado do botão, porém seria mais uma informação para colocar na estrutura da tabela e ainda sim ficariam bordas brancas quando a imagem preta fosse carregada.

Um problema que nos deparamos foi quando as cores de fundo dos botões não eram carregadas na máquina virtual de um dos integrantes. Não sabemos porque isso acontenceu.

Outro dificuldade foi criar várias janelas em cadeia destruindo ou escondendo a anterior de uma forma que repetisse menos código, por isso optamos por não fazer essa implementação para o *tutorial*.

Referências

Haskell, W. (Acesso em Agosto/2019). Gtkh2s/installation.

Haskell.org (Acesso em Agosto/2019). The haskell cabal — overview.