
Número: Nome:

Instruções

- Aceda à versão mais recente do seu repositório individual criado no GitLab para LI1 (faça `git pull`). Crie uma subdirectoria chamada *"miniteste3exemplo"*.
- Resolva as questões pedidas e grave os ficheiros criados na subdirectoria *"miniteste3exemplo"* do repositório. No final do teste, faça `git add`, `git commit` e `git push` dos ficheiros.
- Preencha a informação pedida no enunciado e entregue-o ao docente quando terminar o teste.

Questões

Considere o seguinte programa:

```
module Main where

import Graphics.Gloss
import Graphics.Gloss.Interface.Pure.Game

type Coordenadas = (Float,Float)
type Velocidade = (Float,Float)
type Tempo = Float
type Estado = (Coordenadas, Velocidade, Tempo)

estadoInicial :: Estado
estadoInicial = ((0,0), (0.4, -0.8), 0)

desenhaEstado :: Estado -> Picture
desenhaEstado ((x,y), (vx,vy), t) =
   Pictures [ translate x y $ figura, tempo ]
  where
        figura = color c $ circleSolid 20
        c = if vy<0 then red else green
        tempo = translate 150 150 $ scale 0.1 0.1 $ Text (show $ floor t)

reageEvento :: Event -> Estado -> Estado
-- reageEvento (EventKey (SpecialKey KeyUp) Down _ _) ((x,y),v,t) = ...
--  ...
reageEvento _ s = s --  ignora qualquer outro evento

reageTempo :: Float -> Estado -> Estado
reageTempo n ((x,y),(vx,vy),t) = ((x+vx', y+vy'),(vx',vy'), t+n)
                 where vy' = if y+vy<=(-190) || y+vy>=190 then -vy else vy
                       vx' = if x+vx<=(-190) || x+vx>=190 then -vx else vx

fr:: Int
fr = 50
```

```
dm :: Display
dm = InWindow "Jogo Exemplo" -- título da janela
              (400, 400) -- dimensão da janela
              (200,200) -- posição no ecran

corFundo = (greyN 0.5)

main :: IO ()
main = do play dm -- janela onde irá decorrer o jogo
              corFundo -- cor do fundo da janela
              fr -- frame rate
              estadoInicial -- define estado inicial do jogo
              desenhaEstado -- desenha o estado do jogo
              reageEvento -- reage a um evento
              reageTempo -- reage ao passar do tempo
```

1. Copie o programa para um ficheiro chamado *"miniteste3exemplo.hs"*.
2. Descreva brevemente o que faz a função `reageTempo` neste programa.

3. Altere o programa de forma a que a posição inicial da figura seja `(-20, 100)`.
4. Altere o programa de forma a que a figura também reaja às teclas `KeyUp`, `KeyDown`, `KeyLeft` e `KeyRight`, mas ignore todas as outras. As coordenadas da posição da figura deverão ser actualizadas conforme a tecla pressionada: `KeyUp` transforma (x,y) em (x,y+5), `KeyRight` transforma (x,y) em (x+5,y), etc.
5. Pretendemos incluir no jogo vários outros círculos, dispersos pela janela (número, dimensão e posição dos círculos, à sua escolha; a cor deverá ser diferente da cor da figura já existente no jogo). Os círculos não se devem intersectar. A posição destes novos círculos não deverá ser alterada em função de teclas pressionadas, nem do tempo.
   - Altere o estado de forma a representar esta nova informação (Sugestão: represente no estado apenas a posição dos novos círculos).
   - Altera as funções `estadoInicial` e `desenhaEstado` para que incluam esta nova informação. Ajuste o tipo das restantes funções de forma a garantir que o programa continua a compilar.
   - Altere o programa de modo que os novos círculos desapareçam do mapa quando intersectados pela figura inicial. (Sugestão: use uma função distância entre pontos em vez de comparar directamente coordenadas.)
6. Documente o programa em Haddock.
7. Defina em HUnit um teste que verifique se no estado inicial todas as figuras estão posicionadas dentro da janela.
8. Actualize o repositório com todos os ficheiros produzidos. Se o programa não funcionar, submeta o código que conseguiu escrever. Se quiser, pode usar um único ficheiro para escrever todo o código.