Função

```
pong = function(            #cria a função pong
    mode=c("easy","hard","arcade"),    #modo de jogo com 3 opções
    placar=T                        #ter o placar como output?
)
{

###requerir pacotes apropriados####

if(!require(tcltk2)){install.packages("tcltk2");require(tcltk2)}
#janela da interface
if(!require(tkrplot)){install.packages("tkrplot");require(tkrplot)}
#plotador de gráficos


###acertar o modo de jogo####
mode= match.arg(mode)   #ajusta o que o usuário escreveu para as opções que
existem ou pega a primeira opção caso esteja em branco

ojogo= function(mode){  #não tem mistério, essa função é o jogo inteiro

###preparando objetos úteis####

##bolinha##
bx = 10         #posição em x
by = 5          #posição em y
dbx= 0          #velocidade em x
dby= 0.1         #velocidade em y

###paddle###
px = 10          #posição do centro
ix = 10         #posição do oponente

###atraso do oponente###
delay = switch( #coloca um atraso...
    mode,        #...de acordo com a dificuldde escolhida
    easy  = 20,    #no modo easy ele está 20 frames atrasado
    hard  = 10, #no hard são só 10
    arcade= 4    #no modo arcade o oponente não pode ser derrotado
)
###outros###
res  = "Aperte ok"        #mensagem caso o jogo acabe
GO   = F                #checa se o jogo acabou
pxV  = tclVar(px*10)    #dummy para poder capturar a posição do slider
depois
score= 0                #contador de pontos
```

```
plotatudo = function() {    #função que será chamada quando precisar plotar

    ##criar a base do cenário####

    par(            #retira os labels...
        xaxt="n",    #...do eixo x e...
        yaxt="n",    #...do eixo y
        xaxs="i",   #Coloca a borda do plot nos pontos onde ele
começa/termina (sem dar uma margem interna) em x...
        yaxs="i"    #... e em y
    )

    ##plota a bolinha####

    plot(                    #faz um plot novo
        bx,by,              #coordenadas da bolinha
        xlim=c(0,20),        #limite horizontal do gráfico
        ylim=c(-0.5,20.5),    #limite vertical do gráfico (como os paddles
ficam em 0 e 20, os valores aqui tem uma folga para a bolinha ter espaço
para cair)
        pch=19,                #formato da bolinha
        xlab = "",            #retirando os rótulos de x...
        ylab = ""            #... e de y
    )
    ##plota o paddle####

    segments(            #adiciona uma linha no plot
        x0 = px-2.5,    #dessa coordenada...
        x1 = px+2.5,    #até essa coordenada (calculamos os extremos do
paddle com base em seu ponto central)
        y0 = 0,            #e será horizontal, na altura zero (y1 = y0 por
default)
        lwd= 3            #a espessura da linha
    )
    ###plota o inimigo####
    segments(            #adiciona uma linha no plot
          x0 = ix-2.5,    #dessa coordenada...
         x1 = ix+2.5,    #até essa coordenada (calculamos os extremos do
paddle com base em seu ponto central)
          y0 = 20,        #e será horizontal, na altura zero (y1 = y0 por
default)
          lwd= 3            #a espessura da linha
    )
    ###insere o score####
    text(                #insere um texto na área do gráfico...
        x=20,            #...no canto esquerdo...
        y=20,            #...superior...
        labels=score,    #...contendo o score...
        pos=2            #...e com alinhamento à esquerda para não sair da
área do plot
```

```
    )
}

###cria a interface####

tela = tktoplevel()                      #abre uma janela
tktitle(tela) = "PONG!"                    #muda o título da janela

tela$env$plot = tkrplot(tela, fun = plotatudo, hscale=1.5, vscale=1.5)
#cria um plot na janela "tela" usando a função "plotatudo" criada  e com o
dobro to tamnho default

tela$env$slider <- tk2scale(         #cria um slider...
                tela,                    #...na janela "tela"...
                from = 25,                #...com range de 25 (só números
inteiros são aceitos, então a escala está multiplicada por 10 para ter maior
precisão)...
                to = 175,                #...até 175 (não é de 0 a 200
porque o slider indica o centro do paddle e não queremos que ele saia da
área de jogo) ...
                variable = pxV,            #...associado à variável pxV...
                orient = "horizontal",    #...que desliza na horizontal...
                length = 400            #...e tem esse comprimento
            )
tkgrid(tela$env$plot)                                #Adiciona o plot na
janela
tkgrid(tela$env$slider, padx = 20, pady = c(5, 15)) #Adiciona o slider na
janela


##########################
### aqui começa o jogo ###
##########################


while(GO == F){     #enquanto não der game over
        ##atualiza as posições####

        px = as.integer(tclvalue(pxV))/10        #atualiza a posição do
paddle
        bx = bx+dbx                                #atualiza a posição da
bolinha em x
        by = by+dby                                #atualiza a posição da
bolinha em y
        ix = min(max(bx-(dbx*delay),2.5),17.5)    #atualiza a posição do
oponente sem deixar ele passar da borda (com um delay)

        ##colisão com as laterais####
        if(bx>=20 || bx <= 0){    #se sair da área de jogo na horizontal...
            dbx=dbx*(-1)        #...inverter a direção horizontal
        }
```

```
        ##colisão com o fundo####

        if(by<=0){                      #se sair da área de jogo para baixo
            bx0 = (by*dbx) + bx     #calcula onde o trajeto da bolinha
intercepta o eixo y, explicação no fim do código para poluir menos


            if(bx0>=px-2.5 & bx0<=px+2.5){                #se atravessou o
paddle
                score=score+round(abs(dbx)*10)*100+100    #dá uma pontuação
baseada na velocidade da bolinha quando foi pega, mínimo de 100 porque sim

                dby=dby*(-1)                              #inverte a diereção
vertical
                dbx= min(                                  #atualiza a
velocidade horizontal
                    dbx+(bx0-px)/(2.5)*0.05,              #dependendo da
distancia do centro onde a bolinha tocou, a inclinação do trajeto muda...
                    0.3                                    #..., mas tudo
tem limite
                    )
            }
            else {                              #se não...
                GO = "L"                                    #...game over,
derrota :(
                res= "Bom jogo!\n Aperte ok para fechar"    #mensagem de
derrota
                vic = F                                       #marca o
resultado negativo
            }
        }
        ##colisão com o teto####
        if(by>=20){                              #se sair da área de jogo para
cima:
          bx20 = ((20-by)*dbx) + bx         #calcula onde o trajeto da
bolinha intercepta o teto
          if(bx20>=ix-2.5 & bx20<=ix+2.5){    #se atravessou o oponente
            dby=dby*(-1)                    #inverte a diereção vertical
            dbx= min(                        #atualiza a velocidade
horizontal
              dbx+(bx20-ix)/(2.5)*0.05,     #dependendo da distancia do
centro onde a bolinha tocou, a inclinação do trajeto muda...
              0.3                            #..., mas tudo tem limite
            )
          }
          else {                                          #se não...
            GO = T                                          #...game over,
vitória!
            res= "Você venceu!\n Aperte ok para fechar"    #mensagem de
vitória,
```

```
            vic = T                                          #resultado positivo
            score = score*10                                 #vencer multiplica
seu score por 10
          }
      }

    ##renderizar####
      tkrreplot(tela$env$plot)    #atualiza o plot com as novas posições
calculadas
    }#fim da partida



    ###tela de game over#####
    tkmessageBox(title="Fim de Jogo",message=res, type="ok")    #exibe a
mensagem final
    tkdestroy(tela)                                                 #fecha o
jogo
    return(data.frame(score,vic))                                  #retorna
resultado como dataframe

} #termina ojogo()

##prepara o output####
resumo = data.frame(0,0)            #cria um dataframe para por os
resultados dos jogos
resumo = resumo[0,0]                #esvazia o dataframe

##prepara para jogar pelo menos 1 vez####
play=tclVar(0)  #variável que diz se é pra rodar o jogo

while(tclvalue(play)==0){                              #enquanto não dizer
chega
    resumo = rbind(resumo,ojogo(mode=mode))            #roda ojogo() e salva
o resultado em uma nova linha do resumo
    #depois do jogo
    win = tktoplevel()
#cria uma janela nova
    tktitle(win)   = "Novo Jogo"
#chama de Novo Jogo
    win$env$butsim = tk2button(win, text = "SIM", command = function()
tclvalue(play) <- 0) #Coloca um botâo sim que na prática não faz nada
    win$env$butnope= tk2button(win, text = "NÃO", command = function()
tclvalue(play) <- 1) #Coloca um botão não que diz chega
    tkgrid(tk2label(win, text = "Jogar Novamente?"))
#coloca um texto simples na janela
    tkgrid(win$env$butsim, win$env$butnope, padx = 20, pady = 15)
#coloca os botões na janela e em linha
    tkwait.variable(play)    #espera a variável play ser atualizada (na
prática espera um botão ser apertado) para continuar com o código
    tkdestroy(win)            #fecha a janela
}
```

```
if(placar==T){                                  #se o placar foi pedido
    colnames(resumo) = c("score","vitoria")    #ajeita o nome das colunas
    return(resumo)                               #e pode entregar
}



}
#considerações: ####

#1. Colocar o controle e o jogo em telas diferentes (como dito na proposta) tornava a experiência pior, agora eles estão na mesma janela

#2. O grau de precisão da posição da bolinha é maior do que eu pretendia originalmente, então os cálculos para corrigir a posição após colisões foi retirado por ser quase imperceptível

#3. Na proposta original seria retornado o tempo de jogo como forma de score, mas o calculo de score usado aqui é mais interessante

#4. O aumento de velocidade em y da bolinha também foi retirado por estar ativamente tornando o jogo menos divertido, além disso, para derrotar o oponente jogador terá que acelerar a bolinha por conta própria

#5. Dependendo do programa que você usar para trabalhar no R, as janelas criadas pelo tcltk podem aparecer quando você criar a função (mesmo sem dar o comando para rodar) e podem estar desconfiguradas....
#...Nesse caso, surgirá a janela de fim de jogo com a mensagem padrão "Aperte Ok". Apertar ok deve fazer as janelas fecharem. A função deve rodar normalmente independente disso acontecer.

#6. Os pacotes tcltk permitem que as funções do R sejam chamadas de acordo com eventos e rodem em paralelo, por exemplo o slider pode ter uma função chamada sempre que seu valor é mudado...
#...Provavelmente tem como fazer esse jogo rodar ainda melhor usando essas opções orientadas a eventos...
#..., mas considerando a justificativa da proposta achei melhor evita-las o máximo possível...
#...A janela de novo jogo atribui uma função aos botões que é ativada no evento do botão ser apertado, mas como tk.wait mantém o R em pausa até um botão ser apertado...
#...e essa função apenas atribui um novo valor a uma variável, a lógica não é diferente de pedir input do usuário na linha de comando, usei a janela pela estética.



#Cálculo por trás da intercepção da bolinha:####
# dado que a cada instante bolinha tem movimento retilíneo e uniforme, seu trajeto pode ser expresso em y = x*a + b
# logo:
#
```

```
#   by = bx * a + b             [1]
#   by - dby = (bx - dbx) * a + b    [2]
#
# para o caso em que estamos interessados, dby = -1
#
# substituímos [1] em [2] para descobrir
#
#   a = -1/dbx                  [3]
#
# substituímos [3] em [1] para descobrir
#
#   b = by+bx/dbx            [4]
#
# agora podemos usar [3] e [4] para calcular o valor de bx quando by = 0
#
#   0 = bx0 * (-1/dbx) + by+ bx/dbx
#   bx0 = (by*dbx) + bx
#
#
# o mesmo tipo de raciocínio foi utilizado para calcular bx20
########
```