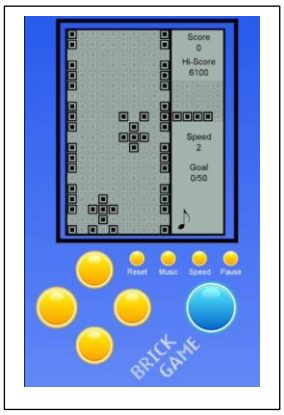
UNIVERSIDADE FEDERAL DE SANTA MARIA

CENTRO DE TECNOLOGIA



Peso: 4
Disciplina: Lógica e Algoritmo
Professor: Dr. Joaquim Assunção
Data de entrega #1: 10/jun/2024
Data de entrega #2: 12/jun/2024

Trabalho prático em duplas

Junte forças com seus colegas para criar o jogo "*BrickGameCar*". O jogo está parcialmente desenvolvido, interface e algumas funções estão prontas, bem como algumas variáveis já estão definidas. Assim, sua tarefa é criar o corpo de algumas funções para que o jogo funcione.

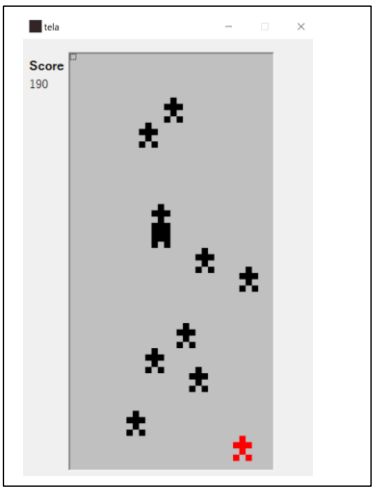
O jogo roda em uma matriz (chamada `mat`) com 30 linhas e 65 colunas. Há somente dois tipos de objetos na matriz:

- `C` caractere que desenha o carro, que posteriormente é renderizado em vermelho.
- `X` caractere que desenha os obstáculos, que posteriormente são renderizados em preto.

O jogo conta com as seguintes funções já implementadas (dentre outras de cunho gráfico):

noBorder(): garante que o carro não sai da tela.

randomStuff(): gera os demais veículos em posição aleatória.



O jogo também conta com as seguintes variáveis globais:

mat: matriz gráfica onde o jogo acontece.

tamX: tamanho do eixo X (quantidade de colunas na matriz).

tamY: tamanho do eixo Y (quantidade de linhas na matriz).

Você precisa programar as seguintes funções (escopo já estabelecido):

Timer(); //chamada automaticamente

Essa função é chamada automaticamente a cada fração de segundo. Você conta com as seguintes variáveis `i,j, count : integer`; e você deve implementar os seguintes algoritmos:

1) Varrer a tela em procura de obstáculos, fazendo-os baixar quando encontrados (o índice `[0,0]` fica no canto superior esquerdo).
2) Verificar se o carro está íntegro (todos os pixels continuam na tela). Caso não esteja, houve colisão e você deve chamar as seguintes funções:

```
timer.Enabled:=false; //para os processamentos
timer_score.Enabled := False; //conta os pontos
application.MessageBox('Game Over',':/');
application.terminate;
```

3) Chame a função para gerar outros veículos.

InicializaObjetos(); //chamada automaticamente

Você somente pode usar as variáveis globais para essa tarefa. Coloque o carro no centro inferior da tela, conforme imagem abaixo:

keyPress(); //chamada automaticamente

Essa função detecta quando você pressiona uma tecla e move o carro para a direita ou para a esquerda. Você conta com as seguintes variáveis `i,j : integer; key: char;` e você deve implementar um algoritmo que detecte as teclas “A” e “D” e move o carro para a esquerda e direita (respectivamente).

Variáveis globais já definidas

`mat`: matriz na tela, use ***mat.Cells[col, lin]*** para alterar os valores de uma célula.

`tamY`: integer; //Total de linhas

`tamX`: integer; //Total de colunas

Funções

`noBorder(direcao)` //dado o parâmetro `direcao` ('l' ou 'r') retorna verdadeiro caso o carro esteja na borda.

`round()` //Converte um real para um integer.

Sintaxe

Pseudo código	Linguagem
"String"	'string'
'char'	'char' //Strings e caracteres usam aspas simples.
SE … ENTAO	if( )then //precisa () para sub-testes lógicos e () para agregar todos
SE … ENTAO ... SENÃO	if( )then begin ... end else
{ ... }	begin ... end;
==	=
<-	:=
Para var de ini ate fim { ... }	for var := ini to fim do begin ... end;
Para var de fim ate ini { ... }	for var := fim downto ini do begin ... end;
Enquanto … faça	while(…) do begin … end;
matriz[I, j]	matriz.Cells[j, i]]

Matriz ordem:

1	2
3	4

Avaliação

O trabalho perfeito é entregue compilável. Pequenos erros de semântica serão descontados de acordo com o erro (0.1 a 0.2 por erro, limite de duas repetições). Erros lógicos terão desconto maior e, dependendo do erro, poderão zerar o trabalho.