IST – Introdução à Arquitetura de Computadores
2015/2016

PÁSSARO BAMBOLEANTE

Programado em Assembly do P3

INTRODUÇÃO

O jogo consiste em fazer com que um pássaro percorra a maior distância possível sem que colida com os obstáculos. Para iniciar o jogo o utilizador tem de premir a interrupção I1. O pássaro cai segundo as leis da Física e sobe, através de um impulso, quando é premido o I0. O utilizador pode ainda usar as interrupções I1 e I2, no decorrer do jogo, para diminuir e aumentar, respetivamente, a dificuldade do jogo (velocidade com que os obstáculos se movem). O jogo termina quando o jogador atingir quer um dos limites do campo de jogo, quer um dos obstáculos.

Quando o jogo termina, é perguntado ao jogador se pretende reiniciar o jogo, bastando para isso voltar a premir I1.

IMPLEMENTAÇÃO

Representação Interna dos Obstáculos

0	**3**	0	0	0	0	0	**9**	0	0	0	0	0	**3**	…	**10**
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13		C78

Exemplo da representação interna dos obstáculos

Para representar os obstáculos decidimos utilizar um vetor, em que cada elemento contém ou zero caso não exista nenhum obstáculo nessa coluna, ou um número positivo entre 1 e 14 inclusive, que representa em que linha começa o espaço do obstáculo dessa coluna.

Implementação das Leis da Física

Para implementar o movimento do pássaro consideramos três variáveis: `P_LINHA_ATUAL`, `LINHA_ATUAL` e `VELOCIDADE`. Considerámos ainda duas constantes: `GRAV_POR_STEP` e `IMPULSO_SUBIDA`. Destes valores, `P_LINHA_ATUAL` é um número inteiro e os restantes são valores fracionários representados em virgula fixa, considerando os 8 bits mais significativos para a parte inteira e os 8 bits menos significativos para a parte fracionaria.

Cada vez que a função `PhysicsStep` é chamada (uma vez de 100ms em 100ms), a `VELOCIDADE` passa a ser igual a `VELOCIDADE - GRAV_POR_STEP` (uma vez que o pássaro tem que descer) e a `LINHA_ATUAL` passa a ser igual a `LINHA_ATUAL - VELOCIDADE` (utilizamos uma subtração em vez de uma soma porque o eixo dos yy “cresce” de cima para

baixo). No fim da rotina, convertemos `LINHA_ATUAL` para um número inteiro e usamos esse valor para verificar as colisões e atualizar a Janela de Texto, ficando o valor inteiro armazenado em `P_LINHA_ATUAL`.

Para além disso quando o botão I0 é premido, a `VELOCIDADE` é substituída pelo valor de `IMPULSO_SUBIDA`, o que faz com que quando `PhysicsStep` for chamado no próximo ciclo o pássaro vá subir em vez de descer, pois a VELOCIDADE é positiva.

Reinício de Jogo

Decidimos ainda implementar uma forma de o jogador poder recomeçar o jogo após ter perdido. Para isso criamos uma rotina `ReiniciaJogo` que coloca todas as variáveis no seu estado inicial, passando depois a execução do programa para o ciclo principal.

CONCLUSÕES

No decorrer deste projeto, foi necessária a criação de várias rotinas auxiliares de modo a implementar as funcionalidades pedidas. Para além das caraterísticas requeridas no enunciado, foi ainda implementada a rotina `ReiniciaJogo` que permite ao jogador o reiniciar o jogo diretamente na janela de texto. Recorremos ao uso de passagem de parâmetros pela pilha e à criação de um vetor para representar os obstáculos para uma maior conveniência.

Para a execução do trabalho, foi essencial a utilização do manual do P3 não só para consultar as instruções suportadas pelo processador, mas também para consultar o funcionamento dos diferentes periféricos (temporizador, janela de texto, LCD, display de 7 segmentos, botões de interrupção e LEDs).

Em suma, consideramos que o projeto teve o resultado esperado por implementar todas as funcionalidades descritas no enunciado (para além de outras que consideramos pertinentes) e correr tanto no simulador como na placa do processador.

Grupo 40
Mihail Brinza - 83533
Ricardo Brancas - 83557
Vasco Pais - 83573