António Romeu, 92427
Francisco Lisboa, 92464
23 de Novembro de 2018

Projeto IAC

Introdução:

No decorrer da execução do nosso projeto, preocupámo-nos tanto com a eficiência como com a clareza do nosso código de forma a que a compreensão e leitura do mesmo seja fácil e acessível. Esforçámo-nos para não repetir código, através do uso de rotinas básicas que utilizam a pilha de maneira a preservar os registos usados.

Variáveis:

Ao longo da realização do projeto fomos criando diversas variáveis necessárias para guardar diferentes valores, que apenas são utilizados em determinadas rotinas. As variáveis principais irão ser explicadas de seguida.

Para calcular as posições da banana ao longo do voo são usadas as variáveis **tempo** (calculada através do **timer**), **vinic** (velocidade inicial) e **ang** (ângulo). Para gerar um valor aparentemente aleatório é chamada a rotina **RANDOM**, que guarda o valor na variável **random**. A variável **numero** serve para guardar os valores introduzidos através dos botões da placa, tanto para a velocidade como para o ângulo, através da rotina **REC_VAL**. **score** serve para armazenar a pontuação do jogador que é atualizada caso o jogador acerte no gorila alvo. A variável **posicao** resulta da concatenação das variáveis **posicaox** e **posicaoy**, correspondente às coordenadas x (nos 8 primeiros bits do valor) e y (nos 8 últimos) da banana na janela de texto.

Visão global:

Trata-se de um projeto que consiste no jogo "Gorillas". Quando o programa é iniciado, são escritas, na janela de texto, através da rotina **ESCREVE**, duas informações sobre o jogo: o objetivo do jogador e como dar início ao jogo em si. O modo de iniciar o mesmo consiste em pressionar o botão IA na placa do *P3*, que ativa a interrupção do mesmo, que, de acordo com o nosso código, equivale à instrução ENTER (usado para declarar os valores da

velocidade e do ângulo, começar o jogo e dar *restart* ao mesmo, caso este acabe ou o jogador introduza um valor inválido para o ângulo e/ou para a velocidade). Após o botão ser pressionado, aparecem na janela de texto três indicadores, ***Velocity*** (velocidade escolhida pelo jogador para o lançamento da banana), ***Angle*** (ângulo escolhido também pelo utilizador) e ***Score*** (pontuação que é inicializada a 0 e cujo valor incrementa em 1 sempre que o jogador acerta no gorila que funciona como um *target*). Tanto a velocidade introduzida como o ângulo são usados para calcular as posições x e y da banana, sendo que o valor do ângulo é tratado tanto pelas rotina pela **RAD** como pela rotina **COS** ou **SEN**, caso queiramos calcular a posição x ou a posição y respetivamente (ambas são usadas na rotina **VOO**, que será explicada mais adiante).

Após o jogador ter introduzido valores para a velocidade e para o ângulo (assumindo que ambos são válidos) e ter pressionado o botão IA, os indicadores são "limpos" da janela de texto e dá-se inicio ao "voo" da banana. São chamadas diversas rotinas para o calculo do mesmo sendo a rotina principal **ACT_TERM**. É através da rotina **VOO** que é escrita na janela de texto a trajetória da banana, sendo que, durante a mesma, são feitas diversas verificações, cujo objetivo é apurar se a banana atinge o "gorila alvo".

A partir do momento em que a banana entra em "voo", existem 3 *outcomes:*

1. caso esta acerte no gorila alvo e o *score* seja menor que 2, a pontuação é incrementada e existe um *restart* que permite que o jogador introduza de novo os valores;
2. se o pontuação for 2 e o jogador acertar no alvo, a pontuação máxima é atingida ("3") e aparece na janela de texto uma mensagem que indica que o jogador ganhou e, caso esta queira reiniciar o jogo, basta pressionar o botão IA;
3. por fim, caso a banana não atinja o alvo, irá eventualmente atingir valores limites ou da posição x (valores maiores que 004Fh (79), que correspondem ao fim do terminal) ou da posição y (qualquer valor da "coordenada" y que esteja entre 9700h e 1700h é tratado como se fosse inválido uma vez que os valores que aparecerem na janela de texto resultam da subtração entre os valores máximo e mínimo (referidos anteriormente) e o valor da posição y) fazendo o *restart* da jogada.

Interrupções:

A interrupção mais importante é o **TIMER**, através desta interrupção são alteradas tanto a variável **tempo** (usada para o calculo das posições), como a variável **atualiza**, que caso seja 1, dá *enable* às várias rotinas. Para além disso é alterado o valor do

TIMER_COUNT, que define quantos milisegundos (neste caso 1) têm que passar para que o *P3* volte a executar o **TIMER**.

Funções não triviais:

O calculo do ângulo é feito através das rotinas **EXP**, **COMPACT**, **FACT**, **RAD**, **COS** e **SEN**. As rotinas **EXP** e **FACT** calculam respetivamente a exponencial e o factorial do valor dado. **COMPACT** é a rotina que trata da posição da vírgula fixa. A rotina **RAD** passa o ângulo introduzido de graus para radianos para que a margem de erro no cálculo das funções trigonométricas seja o menor possível. Para efetuar o cálculo do cosseno e do seno do ângulo, são chamadas as rotinas **COS** e **SEN**; é necessário ter em conta que a rotina **SEN** é calculada a partir de **COS** subtraindo 005Ah (90) ao valor obtido. Todas estas rotinas têm um papel importante no cálculo das posições da banana. Para tal recorremos às rotinas **POSY** e **POSX**, que através do ângulo, do tempo e da velocidade inicial calculam as várias posições em termos de x e y que a banana irá ter ao longo da sua trajetória.

O movimento da banana recorre principalmente a duas rotinas (sem contar com aquelas onde é feito o calculo das posições) que são **ACT_TERM** e **VOO**. Na rotina **ACT_TERM** é feita uma atualização da posição do projétil que calcula as diferentes posições na janela de texto que a banana irá percorrer (trajetória) sendo que é nesta rotina que é feita a inversão da coordenada y para que a trajectória seja um parábola com concavidade voltada para baixo. Para além disso, através da rotina **VOO**, são escritas na janela de texto as várias posições da banana, ocorrendo também a animação da mesma (altera entre “>” e “<”).

Para verificar as colisões, são feitas várias verificações durante a rotina **VOO**. Através da sub-rotina **CHECK_H** são feitas 2 averiguações, para a posição x e y da banana, sendo ambas as posições da banana comparadas com as posições do gorila alvo. Caso se verifique que a banana atinge o gorila alvo é incrementado o *score* e é feito um *restart* à jogada (ou ao jogo em caso de pontuação máxima).

Notas:

Apesar de termos sido aconselhados a fazer uma tabela com os valores do seno e do cosseno, preferimos fazer uma rotina, com base na série de Taylor, que calculasse os valores dos mesmos porque concordámos que se enquadrava melhor no nosso estilo de programação (baseado em rotinas independentes).