Hugo Manuel Alves Henriques e Silva, nº 99235
Diogo Luís Silveira Vieira, nº 99205

Projeto de IAC

Antes de inicializar o programa, são escritas todas as constantes, variáveis, strings e vetores. Neste trabalho, muitas das variáveis são usadas como variáveis de controlo, e desempenham um papel fundamental no decorrer do jogo.

O programa começa com a definição do “stack”. De seguida, e antes de entrar na “main”, que é o controlo de todo o programa, são ativadas as interrupções e o programa fica preso num loop infinito até que o botão b0 seja pressionado. Antes de pressionar nesse botão, encontra-se no centro do terminal a mensagem “PRESS “0” TO START”, que desaparece com a ativação do botão. Ainda antes de entrar na “main”, é feito um full reset de todas as variáveis de jogo, tal como um reset do terminal e dos valores presentes no vetor que contém o terreno de jogo, com a respetiva altura dos catos. Este full reset é feito, para que, aquando do recomeço de uma nova partida, o programa se encontre perfeito para ser novamente corrido, apenas com um terreno diferente. Após o full reset, é inicializado o timer. O timer controla a frequência a que cada loop do programa ocorre, e está definido para que a cada 100ms haja um evento, que desencadeará um novo loop. Aquando de cada evento, a variável “TIMER_TICK” é incrementada, de forma a que seja detetado que há um evento do timer para tratar. Assim, o tratamento deste evento implica que o programa se desloque finalmente para a “main”, local de controlo de todo o programa. Antes de tudo, é decrementado o valor da variável “TIMER_TICK” para afirmar que o evento está a ser tratado, e para que seja possível ocorrer um novo evento após o tratamento do presente. Na “main” encontram-se a maior parte das funções utilizadas em todo o programa. Assim, após a finalização de cada função, o programa regressa à “main”, que irá chamar a função seguinte e assim sucessivamente. No final da “main”, o programa regressa ao loop infinito do timer, à espera que exista um novo evento para ser tratado.

A primeira função a ser executada, é a “draw_dino”, em que é feito não só um reset do terminal para prepará-lo para que seja updated com o novo terreno, como também é nela onde é desenhado o dinossauro por cada loop. Este é desenhado na altura que se encontra no momento de escrita, independentemente se estar a decorrer um salto, ou não. O dinossauro é representado com um “o” colocado em cima de um “T”.

A segunda função a ser executada é a função “atualizajogo”. Esta função é a responsável pela escrita do terreno nas 80 posições do vetor e também é a função que desloca o terreno uma posição para trás por cada loop do programa. Dentro desta mesma função, existe uma função auxiliar que serve essencialmente para o desenho do terreno em si, escrevendo ao longo de uma linha inteira do terminal não só o chão, como também os catos nas posições em que existem, com a sua respetiva altura. A altura dos catos varia entre 1 e 4, sendo representado com o símbolo “|”, variando o número de vezes que este símbolo é representado, com a respetiva altura de cada cato, ou seja, num cato de altura 4, o símbolo será representado 4 vezes na vertical.

A terceira função é a “geracato”. Esta função é responsável pela criação ou não, de um cato na ultima posição do vetor. São desencadeados conjuntos de processos que levam a uma possível criação de um cato. A probabilidade de ser criado um cato, ronda os 5% uma vez que apenas 5% dos números representáveis com 16 bits são capazes de gerar um cato, e é a partir de um número “random” que essa possibilidade ocorre. A altura de cada cato também depende desse número, no entanto, nunca excedendo a altura máxima definida inicialmente, que é de 4. Por cada loop, a variável que permite esta criação random de um cato, “x”, é alterada e updated, de forma a que no próximo loop, exista uma nova chance de ser criado um cato na ultima posição do vetor.

A função que se segue é a “SCORE”. Esta função tal como o nome indica, é responsável pela pontuação que o utilizador está a obter no decorrer do jogo. Quanto mais tempo o utilizador sobrevive, maior a sua pontuação. Esta pontuação é mostrada nos displays de 7 segmentos, em decimal. Quando o valor de cada display ultrapassa o valor 9, regressa a 0 e o display seguinte aumenta em 1 unidade. Foram criadas variáveis para cada display de forma a que fosse possível fazer update do score nos respetivos displays. Por cada loop a pontuação é aumentada em 1, ou

seja, por cada 100ms a pontuação aumenta em 1. Apesar de ser quase impossível chegar ao limite da pontuação, caso se chegue, é feito um full reset dos displays, para afirmar que o utilizador ganhou o jogo.

A quinta função é a “process_key_up”. Esta função apresenta um funcionamento semelhante à deteção da interrupção do botão “b0”. Assim, quando é pressionado o botão seta para cima do p4, é gerada uma interrupção. O tratamento dessa interrupção é feito exatamente por esta função. Caso seja detetado o click neste botão, é alterado o valor de uma variável de controlo, a variável “key_up_var”. Quando o valor desta variável se encontra a 1, significa que foi pressionado o botão e o programa irá saltar para esta função. Este botão desempenha a função de salto do dinossauro, ou seja, quando é pressionado enquanto o dinossauro se encontra no chão, este salta. Se estiver no ar, nada acontece. Assim, a primeira coisa que é feita ao entrar nesta função, é verificar se decorre um salto. Se sim, o programa regressa à “main”, e o valor desta variável volta a ser 0, se não, é alterado o valor de uma outra variável de controlo do salto, a variável “estado_do_salto” que se encontra a 0 quando não há salto, e se encontra a 1 quando decorre um salto. Por fim, é também alterada a variável “subir_descer” que se encontra a 1 quando o dinossauro está a subir, e a 0 quando está a descer, ou seja, o valor desta variável é alterado para 1 no final desta função.

A penúltima função é a “SALTO”. O programa apenas irá executar esta função, caso o valor da variável “estado_do_salto” se encontre a 1, ou seja, quando decorre um salto. Caso esta condição não se verifique, o programa passa à frente esta função. Ao entrar nesta função, é verificado se o dinossauro está a subir ou a descer através do valor da variável “subir_descer”. Dependendo do valor da mesma, a altura do dinossauro será updated descendo ou subindo. Caso esteja a subir, não terminará de subir até que seja atingido o valor da altura máxima de salto, que está definida como 7. Ao atingir a altura máxima, começa a descer, alterando o valor da variável “subir_descer” e terminando a descida quando atinge a altura uma unidade acima do chão. A altura atual do dinossauro em valor decimal, é controlada pelo valor dar variável “contador”, que é o que determina o fim da subida e o fim da descida, sendo alterado no decorrer do salto. Ou seja, a partir do momento em que o valor da variável “contador” atinge o valor da altura maxima do salto, o dinossauro começa a descer até a variável “contador” atingir o valor da altura inicial do dinossauro antes do salto. Após terminar a descida, a variável “estado_do_salto” é novamente colocada a zero, para indicar que deixou de existir salto. Esta função prepara a escrita do dinossauro para cada loop do programa.

Finalmente, a ultima função do programa é a “COLISAO”. Esta função indica se houve colisão entre um cato e o dinossauro. A forma de a implementar é muito simples. Se na posição do vetor em que o dinossauro se encontra existir um cato e a altura em que o dinossauro se encontra for menor ou igual à altura do cato, há colisão e o jogo termina com a mensagem de “GAME OVER” e o terreno parado com o dinossauro na posição onde houve a colisão. Além desta mensagem, é representada a mensagem de “ PRESS “0” TO RESTART”. Assim, quando for pressionado novamente o botão “b0” o jogo recomeça após ser feito um full reset do programa e de todas as suas variáveis de controlo, terminal e vetor como referido anteriormente. O jogo pode ser jogado o número de vezes que o utilizador quiser.

As mensagens de texto que se apresentam no terminal foram feitas através de strings e loops que percorrem essas strings caracter a caracter e as escreve no terminal, numa posição centrada.

As interrupções do timer, do botão “b0” e do botão “seta para cima” ocupam valores específicos de memória. Para as interrupções do timer, foi criada uma função auxiliar já que os 16 espaços existentes não eram suficientes para o tratamento dessa interrupção.

Finalmente, o programa está organizado ao máximo, com as respetivas funções separadas e identificadas, tal como é referida a linha em que se encontram no programa.