Relatório de Projeto de IAC

Diogo Ramalho, n.86407
Rafael Pestana de Andrade, n.86503

Na nossa implementação, o ciclo de jogo principal consiste no teste de variáveis de estado (flags), que consoante o seu estado atual (ativado/desativado) faz com que sejam chamadas rotinas apropriadas a cada estado. O ciclo de jogo testa diretamente 3 *flags*: F_INTERRUPT, que assinala um botão carregado; F_TICKTACK, assinala uma interrupção do temporizador; e F_GAME_OVER, que assinala uma situação de fim de jogo. Antes de entrar no ciclo de jogo, é chamada a rotina *Intro*, que escreve no ecrã as palavras iniciais, bem como rotinas que iniciam o temporizador (*Recontagem*) e desenham os limites de jogo (*Wall*) e a nave (*Desenha_nave*).

A *flag* F_INTERRUPT chama a rotina Interrupt, que testa outras *flags* relacionadas com interrupções de botões: F_BAIXO, F_CIMA, F_TRAS, F_FRENTE, F_SHOT, F_PAUSE e F_REINIT. Cada uma destas flags, se estiver ativa, chama rotinas que tratam de ações correspondentes: movimentam a nave para baixo, cima, trás e frente, criam o tiro, pausam o jogo ou reiniciam o jogo. Após cada rotina, a *flag* que a chamou é desativada. No final da verificação das flags associadas a movimento da nave, é verificada a *flag* F_GAME_OVER, uma vez que a deslocação da nave pode ter gerado uma colisão com um asteróide ou buraco negro. Se esta *flag* estiver ativa, é chamada a rotina do fim de jogo (rotina Game_over).

A *flag* F_TICKTACK chama a rotina Temporizador, que rege os objetos que são atualizados segundo o temporizador. Em primeiro lugar, verifica se qualquer tiro ou objeto celeste (asteróides ou buracos negros) sai dos limites de jogo se a sua coordenada for atualizada (rotina Saiu_limites). Se sim, apaga esse objeto (tiro ou objeto celeste) do ecrã e da memória. Depois, incrementa as coordenadas de todos os tiros que existem em memória (rotina Atualiza_tiro). Após este incremento, verifica se esta rotina também deve atualizar objetos celestes (se já houveram 2 interrupções de relógio desde a última atualização). Se sim, a rotina permite o decremento das coordenadas dos objetos celestes (rotina Atualiza_Cel) e verifica se existem colisões com a nave (rotina Colisoes_nave). Se existirem, é ativada a *flag* F_GAME_OVER; e se esta *flag* estiver ativa, é feito um salto para o fim desta rotina, para poder chamar a rotina do fim de jogo a partir do ciclo principal. Se a *flag* F_GAME_OVER não estiver activa, procede-se ao desenho no ecrã de todos os objetos celestes em memória, bem como ao desenho de espaços vazios nas posições que anteriormente ocupavam (rotina Desenha_cel). Após a execução desta rotina, caso já existam 5 espaços desde o último objeto celeste gerado , é chamada a rotina Cria_ASTER_BHOLE, que cria um asteróide ou um buraco negro dependendo das condições de geração, chamando a rotina Random para gerar um novo número pseudo-aleatório e uma nova coordenada para o objeto celeste.

Tenha atualizado as coordenadas dos objetos celestes ou não, a rotina Temporizador chama depois rotinas que verificam se houve colisão entre tiros e objetos celestes (rotina Colisoes_tiro), realizando todas as ações necessárias (apagar o tiro do ecrã

e da memória, incrementar a pontuação e mostrar no display de 7 segmentos, etc.) e desenha os tiros existentes em memória, apagando do ecrã as suas antigas posições (rotina Desenha_tiro). Após estas ações, retorna ao ciclo principal, onde testa se há uma situação de fim do jogo.

Havendo uma situação de fim de jogo, a rotina Game_over é chamada e limpa o ecrã, mostra a mensagem de fim de jogo e a pontuação final, ficando à espera de uma interrupção. Quando a interrupção é gerada, os valores de estado da memória são reiniciados, e recomeça o jogo.

Os dados do jogo estão todos organizados a partir da posição de memória 8000h, sendo que estão estruturados ou por tabelas (como as posições dos tiros e de asteróides) ou em posições de memória singulares, por exemplo as variáveis de estado.

O código do ciclo principal está guardado no início da memória (posição 0000h em diante), as rotinas chamadas por interrupções estão imediatamente a seguir à própria tabela (após FE00h); todas as outras rotinas, chamadas pelo ciclo principal, foram guardadas a partir da posição 4000h.

Quanto à organização do código nas páginas, na primeira página estão constantes, como posição inicial da nave, número de colunas e linhas, ou a máscara de interrupções; na segunda, temos um “Arquivo de dados”, onde está descrita as variáveis de estado e posições de todo o jogo, e como estão organizadas na memória. Entre a segunda e a terceira páginas, está a tabela de interrupções com as rotinas de interrupção. A partir do meio da página 3, temos as rotinas que são usadas em situações de início ou fim de jogo (ex: rotina Intro ou rotina Game_over); na página 7 e seguintes temos as rotinas que são ativadas por variáveis de estado correspondentes às interrupções dos botões; nas páginas 10 a 16 temos código referente a rotinas chamadas pela variável de estado do temporizador (ex: rotina Cria_ASTER_BH ou rotina Colisoes_tiro). Da página 16 à 18 estão rotinas que tratam dos periféricos (ex: rotina Inicia_LCD ou rotina Escreve_Pontos). Na última página encontra-se o código que é inicialmente executado, bem como o ciclo principal de jogo.

Neste projeto, a funcionalidade de voltar a iniciar o jogo usando um interruptor em vez de um botão não foi implementada, tal como não foi implementado múltiplos níveis de jogo. No entanto, foi implementado um mecanismo de pausa durante o jogo, carregando no botão ID. Carregando no botão IE, o novo jogo não se inicia imediatamente, apresentando antes um novo ecrã de introdução. Foi implementada a possibilidade de disparar múltiplos tiros, mas só podendo ter 15 tiros simultaneamente no jogo, para possibilitar um jogo rápido sem que haja abusos da parte do jogador. Após atingir esse limite, o jogador tem de esperar até que um dos tiros disparados seja destruído. Aquando da criação de objetos celestes, estes estão entre a terceira e a antepenúltima linha, para possibilitar a destruição de todos os asteróides gerados. Quando a nave toca num dos limites de jogo, o jogo não termina, para permitir deslocação rápida até aos limites de jogo sem a preocupação de poder perder só por tocar nas paredes.