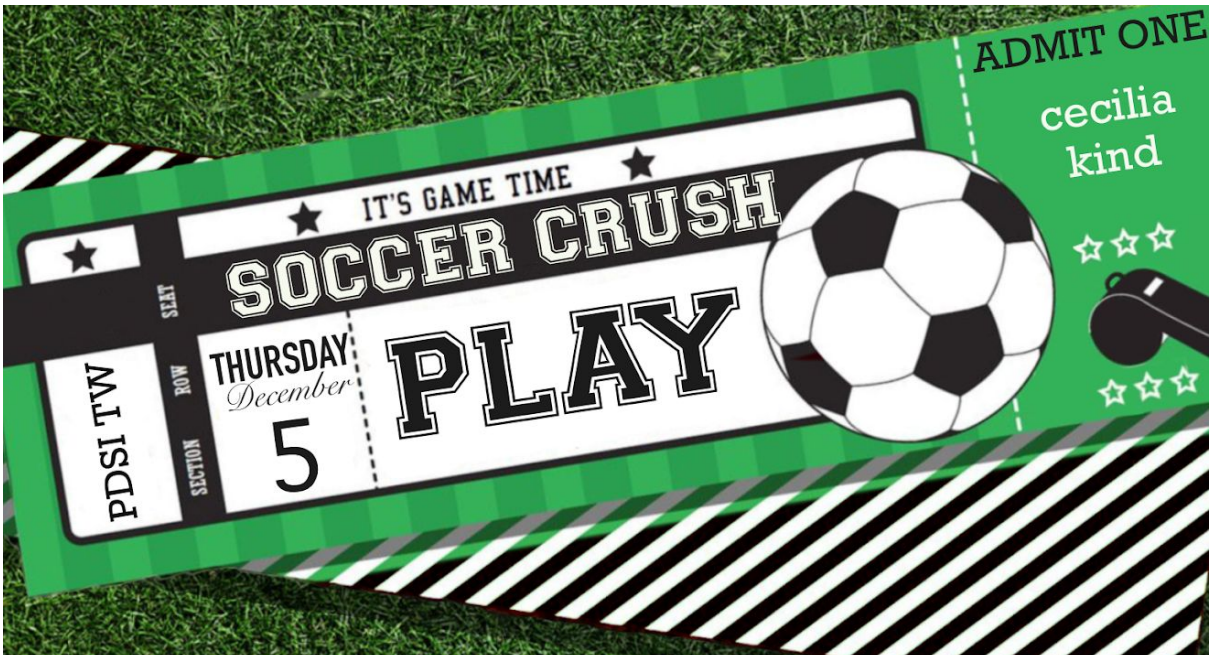
Documentação Trabalho Prático
“Soccer Crush”

Programação e Desenvolvimento de Software I
05/12/2019

Cecília Conde Kind



1. Instruções do Jogo:

1.1 Descrição

O jogo Soccer Crush é uma versão simples do jogo Candy Crush, cujo objetivo é somar a maior quantidade de pontos dentro do número de jogadas disponíveis. O jogador obtém pontos ao formar sequências de 3 ou mais peças da mesma cor na vertical ou horizontal. Para trocar as peças de lugar, o jogador deve clicar em duas peças distintas que estejam lado a lado, ou uma logo acima da outra. Caso o jogador tente selecionar uma segunda peça em uma posição inválida, a primeira peça é desselecionada e a segunda se torna a nova referência. Caso a troca do jogador ocasione em uma nova sequência, a sequência é removida do jogo e as peças que estão logo acima são reposicionadas de forma a ocupar o espaço vazio. Finalmente, peças aleatórias são colocadas na posição dos espaços vazios após a movimentação de todas as peças. Uma vez que as peças se movimentam, novas sequências podem ser geradas a partir da primeira, e o jogador deve esperar até que não haja mais sequências para fazer a próxima jogada. Quando o jogador

atinge o número máximo de jogadas disponíveis o jogo acaba. Caso a pontuação seja superior ao recorde atual o jogo é atualizado de acordo.

1.2 Tela do jogo

A tela principal do jogo possui três indicadores. Score representa a somatória dos pontos do jogador a cada jogada; o visor central representa o número de jogadas restante; o visor Goal indica o recorde a ser superado.

1.3 Controles

O movimento das peças é feito apenas com o click do mouse.

2. Implementação

2.1 Estruturas

Ball: esta é a única estrutura utilizada no jogo, e possui os seguintes membros: row e col são inteiros e representam o índice da linha e da coluna ocupada pela peça respectivamente. O inteiro “sel” indica se a peça está selecionada ou não. A peça é considerada selecionada se de alguma forma ela está envolvida em algum movimento. O inteiro ncolor indica a cor da peça obtida aleatoriamente, e representa também o índice de um vetor de Bitmaps correspondente a bcolor, que indica qual imagem deve ser colocada em função da cor de cada peça.

2.2 Procedimentos e funções

int main(): A função da main chama a função gamestate para iniciar o menu o contém o “while”principal do jogo. Os eventos possíveis são de término do jogo, que finaliza o Display, e a identificação de cliques do mouse. Uma vez que duas peças são selecionadas a função handlemouseclick é chamada para realizar as trocas. O while apenas termina quando o jogador fecha a janela.

void changegamestate(int gamestate): atualiza a tela de acordo com o estado do jogo: menu (state1), jogo (state2) e gameover(state3).

void initmatrix(Ball matrix[MAX_ROW][MAX_COL]): Esta função é chamada apenas na inicialização do jogo. Ele preenche a matriz com o tipo Ball e declara os membros de cada item da matriz. Row e col são preenchidos a partir da posição ocupada na matriz. A cor é determinada aleatoriamente e o bitmap é determinado em função do número representante da cor. Todas as variáveis são inicializadas com sel = 0. Uma vez que a matriz é preenchida, a função que identifica sequências é chamada e as sequências são atualizadas até que não haja mais sequências.

void initstate2(Ball matrix[MAX_ROW][MAX_COL]): Esta função desenha o plano de fundo, a tabela com os pontos, jogadas restantes e o recorde. Também preenche o display com as peças por meio da função color.

void color(Ball matrix[MAX_ROW][MAX_COL], int row1, int col1): desenha o bitmap referente a cada peça da matriz de acordo com o membro bcolor. Caso o membro sel esteja ativo, a peça não é desenhada.

Ball get_piece(int mouse_x, int mouse_y, Ball matrix[MAX_ROW][MAX_COL]): essa função recebe a posição x e y do clique do mouse, calcula a linha e a coluna equivalente a essa posição e retorna a peça que as ocupa.

int insidegrid(int x, int y): Verifica se o clique ocorre dentro do “tabuleiro” do jogo.

void handlemouseclick(Ball auxball[], Ball current): esta função é chamada na main para administrar os cliques do mouse. Caso não haja nenhuma peça selecionada, aux[0] é preenchido com os valores equivalentes no clique. Caso haja uma peça selecionada mas a nova peça não encontra-se em uma posição válida, a primeira peça é desselecionada e a segunda passa a ter o valor equivalente ao novo clique. Caso a primeira peça esteja selecionada e o segundo clique indique uma posição válida, o valor de aux[1] é preenchido a partir do segundo clique. Uma vez que duas peças estão selecionadas, a função decrementa o número de jogadas disponíveis, chama a função para movimentar as peças (swapballsanimation), troca as peças (swapballs) e chama a função para buscar sequências (possiblemoves).

int checkrelativeposition(Ball b1, Ball b2): Verifica se a nova peça selecionada está lado a lado na horizontal ou vertical com a primeira peça selecionada.

void swapballsanimation(Ball matrix[MAX_ROW][MAX_COL], Ball b1, Ball b2): utiliza um timer para fazer a movimentação da troca de duas peças dependendo da posição relativa entre elas. Termina quando uma peça ocupa completamente o lugar da outra.

void swapballs(Ball matrix[MAX_ROW][MAX_COL], Ball b1, Ball b2): Troca a posição de duas peças: primeiro os membros da struct row e col, e depois a posição na matriz.

void possiblemoves(Ball matrix[MAX_ROW][MAX_COL]): Esta função é responsável por administrar a identificação de sequências e atualização da matriz. O procedimento inicia chamando a função para zerar a matriz auxiliar e em seguida a de identificar as sequências presentes. Se alguma sequência é identificada, um loop é iniciado até que não haja mais sequências disponíveis. Nesse loop são

chamadas as funções responsáveis por eliminar as sequências, atualizar a matriz, preencher os espaços vazios e fazer a movimentação das peças. Uma vez que a matriz é atualizada, a função de identificar novas sequências é chamada. A função também atualiza os pontos e troca o estado de jogo caso o número de jogadas disponíveis chegue a 0.

void zero_aux_matrix(int m[MAX_ROW][MAX_COL]): preenche todas as posições da matriz com 0.

void findsequence(Ball m[MAX_ROW][MAX_COL]): esta função é dividida em duas partes. Na primeira parte ocorre a busca de sequências na matriz horizontalmente. Uma vez que uma sequência é identificada a função mark_row é chamada para marcar a sequência na matriz auxiliar. Em seguida, o mesmo processo é repetido na vertical.

void mark_row(int startrow, int startcol, int sequence): esta função recebe a coluna e a linha da primeira posição da sequência e o tamanho da sequência a ser preenchida na horizontal, que é marcada como 1 na matriz auxiliar inicialmente zerada.

void mark_col(int startrow, int startcol, int sequence): esta função recebe a coluna e a linha da primeira posição da sequência e o tamanho da sequência a ser preenchida na vertical, que é marcada como 1 na matriz auxiliar inicialmente zerada.

int remainingSequence(): Conta a quantidade de 1s na matriz auxiliar e retorna este número, indicando a presença de uma sequência se o valor de retorno é maior do que zero.

void zero_matrix_sequence(Ball matrix[MAX_ROW][MAX_COL]): Esta função atualiza a matriz original de acordo com as sequências marcadas na matriz auxiliar. Caso uma peça seja correspondente ao número um na matriz auxiliar, sua cor será atualizada para 0, indicando que ela será eliminada.

void updatematrix(Ball matrix[MAX_ROW][MAX_COL]): Esta função atualiza a matriz de forma que as sequências formadas são colocadas no topo da matriz e as demais peças são reposicionadas para ocupar o espaço desocupado. A função é percorrida de baixo para cima começando do canto esquerdo. Quando uma peça com o valor da cor 0 é identificado, essa peça é movida para a primeira linha e as demais são reposicionadas. Todas as peças que trocam de lugar são selecionadas e sua posição é trocada apenas na matriz. O valor dos membros row e col não são alterados para auxiliar na movimentação futuramente.

void randmatrix(Ball matrix[MAX_ROW][MAX_COL]): Esta função verifica as peças com a cor correspondentes a 0 na matriz e substituem o valor por um novo número aleatório (outra cor). A função também desseleciona essas peças para que sejam desenhadas na próxima atualização e atualizam os membros row e col para que sejam equivalentes a linha e a coluna da matriz.

void move matrix(Ball matrix[MAX_ROW][MAX_COL]): essa função realiza o movimento das peças que foram afetadas pela remoção de uma sequência. Há um loop principal que termina apenas quando todas as peças se movimentam para seus devidos lugares. Dentro do while, um for percorre toda a matriz, verifica quais peças estão selecionadas e faz seu movimento comparando os membros row e col de cada peça (que apresentam a posição da peça anterior ao movimento) e a posição da peça na matriz, que foi atualizada na função updatematrix (indicando sua nova posição). Uma vez que a peça encontra-se no devido lugar, row e col são atualizados e a peça é desselecionada.

int newMaxScore(): Essa função abre um arquivo para leitura e lê o número do recorde registrado, salvando-o em uma variável. Em seguida, compara esse valor com a pontuação do jogador. Caso a pontuação seja superior ao recorde, o arquivo é aberto para escrita e o recorde é atualizado. Caso não haja um arquivo inicial, um novo arquivo é criado.

int insidebutton(int x, int y): Confere se o clique do mouse encontra-se dentro dos limites do botão da página final para retornar ao menu.

void initstate3(): inicia a tela de final do jogo e mostra a pontuação obtida pelo jogador. Caso a pontuação seja maior do que o recorde atual, o jogador é parabenizado e o recorde é atualizado. Caso contrário, o jogador é convidado para jogar novamente e sua pontuação é descartada.

void printmatrix(int m[MAX_ROW][MAX_COL]): imprime a matriz que identifica as sequências no terminal.

void restart_aux_ball(Ball auxball[], int i): reinicia as duas peças auxiliares para que possam ser utilizadas na próxima jogada novamente.

void play_sound(): toca um som em função do total de objetos eliminados na sequência.

void createtimer(): esta função cria uma fila de eventos apenas para a utilização de um timer, que é criado e destruído quando a movimentação desejada acontece.

bool inicializar(): inicializa as bibliotecas do Allegro necessárias para o jogo, assim como a display e os itens que serão utilizadas: bitmaps fontes e áudio.