**DOCUMENTAÇÃO - TRABALHO PRÁTICO**

**ALUNO:** DIEGO VINICIUS DE OLIVEIRA SILVA

**DISCIPLINA:** PROGRAMAÇÃO E DESENVOLVIMENTO DE SOFTWARE I

**PROFESSOR:** PEDRO O. S. VAZ DE MELO

**Como o jogo funciona**

O jogo é similar ao Candy Crush, ele é controlado pelo mouse, e no início da partida várias figuras geométricas são dispersas em uma matriz, que é a tela, com cada uma ocupando a mesma altura e largura. O objetivo é realizar combinações com essas figuras, pois cada uma possui uma forma geográfica. O usuário clica em uma figura e a arrasta para outra a uma casa de distância de onde clicou. Após a troca ser realizada, o jogo verifica se existe alguma combinação válida, e caso haja, as figuras são removidas da tela e contabilizadas como pontos para o jogador. Uma combinação válida é aquela que possui três ou mais figuras do mesmo tipo na mesma linha ou coluna, quanto mais figuras tiverem, mais pontos são adquiridos. O jogo termina quando as jogadas são esgotadas.

**Estruturas de dados**

Para construir o jogo foi utilizado de uma estrutura principal, que é uma matriz n x m, em que cada posição dela é armazenado um Candy. O Candy é uma Struct que contém os campos “type”, “active” e “color”. Esses campos são utilizados para poder validar as sequências que serão realizadas dentro do jogo, juntamente com o design de cada figura geométrica, que é dependente de “type“ e “color”. Essa estrutura reflete diretamente na tela do jogo, onde cada posição da matriz é vinculada a uma posição da tela, seguindo a ordem da matriz.

**Como o código funciona**

O código consiste em diversas funções que realizam ações envolvendo a matriz e a organização da mesma na tela. E da função main, que executa o jogo e o mantém rodando até os movimentos do jogador serem esgotados.

Na main, nós temos a inicialização do jogo e a inicialização do loop em que o jogo funciona. O loop verifica os eventos que são ativados ao decorrer do jogo, e realiza algumas ações com cada tipo diferente de evento.

No final, após o loop, existe uma tratativa para as rotinas do fim do jogo.

**Eventos:**

* Tecla ESC pressionada: Fecha o jogo
* Clique do mouse no X: Fecha o jogo
* Tecla espaço pressionada quando o jogo foi finalizado: Reinicia o jogo
* Clique do mouse em um candy, e arrastando-o para outro a uma casa de distância dele, faz com que os candies sejam trocados de lugar.
* Temporizador:

Nesse evento o jogo valida se existe alguma pontuação na matriz, caso exista a sequência é realizada e removida da tela, com os pontos sendo armazenados na pontuação do jogador. Esse evento também realiza o preenchimento da matriz, após alguma sequência ser completada. O temporizador também valida o reiniciamento do jogo, após uma partida ser finalizada.

Todo o funcionamento do jogo é feito com base nos eventos acima, e esses eventos chamam estas funções presentes no código.

**Funções:**

***int* initializeAllegro(ALLEGRO\_DISPLAY \*\*display, ALLEGRO\_EVENT\_QUEUE \*\*event\_queue, ALLEGRO\_TIMER \*\*timer, ALLEGRO\_SAMPLE \*\*moveSound, ALLEGRO\_SAMPLE \*\*scoreSound)**

É responsável por iniciar o allegro, inicializando desde o display, timer, até as as variáveis globais utilizadas no sistema.

Candy generateCandie()

É responsável por gerar um Candy, que é a Struct utilizada dentro da matriz primordial do sistema.

*void* initCandies()

É responsável por inicializar a matriz dos Candies, gerando um Candy por posição da matriz.

*void* pause(ALLEGRO\_TIMER \*timer, *float* seconds)

É responsável por pausar o jogo, é utilizada para manter alguns textos na tela por um certo período de tempo, ela também pausa o timer.

*void* drawCandy(*int* lin, *int* col)

É responsável por desenhar um Candy na tela, de acordo com sua posição da matriz e do seu tipo.

*void* drawScenario(ALLEGRO\_DISPLAY \*display)

É responsável por desenhar todo o cenário, ele atualiza o cenário como um todo com base nas variáveis do sistema, como a matriz de candies, pontuação, jogadas restantes, etc.

*void* getCell(*int* x, *int* y, *int* \*lin, *int* \*col)

É responsável por receber o valor de x e y de onde foi clicado na tela, e atribuir para lin e col os valores da linha e coluna relacionados ao Candy da matriz.

*int* getDistance(*int* lin\_src, *int* col\_src, *int* lin\_dest, *int* col\_dest)

É responsável por retornar a distância entre dois pontos.

*int* isCandyDiagonal(*int* lin\_src, *int* col\_src, *int* lin\_dest, *int* col\_dest)

É responsável por validar se um Candy está em uma posição diagonal, quando comparado a outro.

*void* changeCells(*int* lin\_src, *int* col\_src, *int* lin\_dest, *int* col\_dest)

É responsável por trocar de posição na matriz dois candies, um com o outro.

*int* scoreSequence(*int* lin, *int* lin2, *int* col, *int* col2)

É responsável por realizar a sequência que foi identificada e passada para a função.

*int* checkGame()

É responsável por validar o jogo, essa função consiste em verificar se existe alguma sequência a ser feita, se existir, ela a realiza e retorna os pontos feitos.

*void* updateGame()

É responsável por atualizar os Candies na matriz, fazendo com que os Candies caiam caso não exista algum Candy abaixo deles.

*void* fillMatrix()

É responsável por preencher a matriz com novos Candies.

*void* startGame(ALLEGRO\_DISPLAY \*display)

É responsável por iniciar o jogo, ele trata a inicialização dos Candies para a matriz sempre ser inicializada sem nenhuma sequência já feita.

*int* newRecord(*int* score, *int* \*record)

É responsável por validar se a pontuação do jogador foi além do recorde armazenado, se foi, então armazena essa nova pontuação.

*void* endGame(ALLEGRO\_TIMER \*\*timer)

É responsável por finalizar o jogo após as jogadas esgotarem, ela limpa a tela e, caso você tenha ultrapassado o último recorde, ela te alerta disso, se não, exibe o último recorde.

*void* restartGame(ALLEGRO\_DISPLAY \*display)

É responsável por reiniciar o jogo, limpando as variáveis globais e iniciando o jogo novamente.