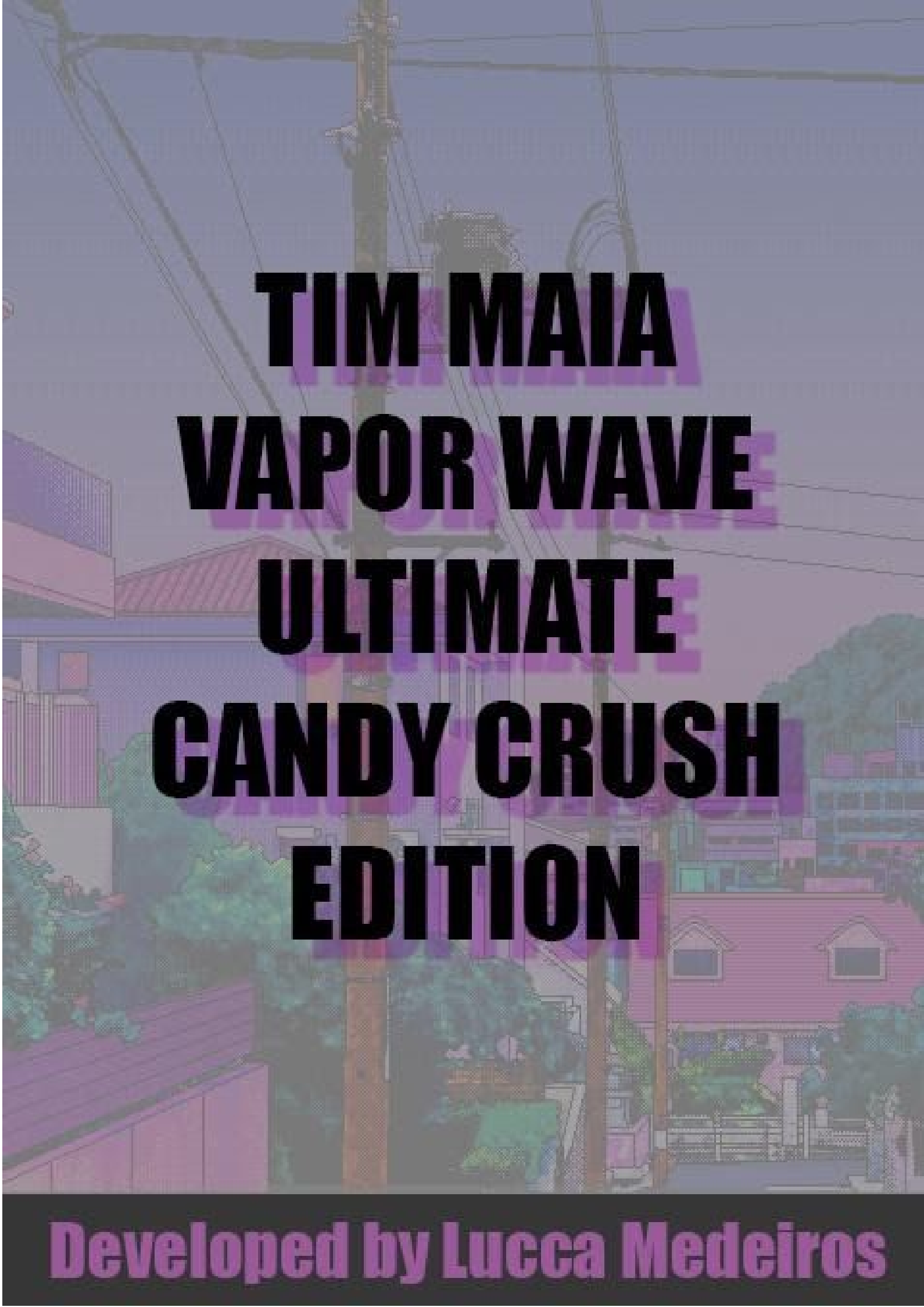

TIM MAIA
VAPOR WAVE
ULTIMATE
CANDY CRUSH
EDITION
Developed by Lucca Medeiros

DOCUMENTAÇÂO – TP ALLEGRO – 2019.2

Nome: Lucca Silva Medeiros
Disciplina: Programação e Desenvolvimento de Software I
Professor: Pedro O.S. Vaz de Melo

Conceito do jogo

Tim Maia Vapor Wave Ultimate Candy Crush Edition é um jogo que engloba dois conceitos em uma só experiência: o movimento musical Vapor Wave e o artista Sebastião Rodrigues Maia, conhecido pelo povo brasileiro como Tim Maia. Vapor Wave é um gênero musical que surgiu na década de 2010 entre diversas comunidades online. O gênero é caracterizado por uma fascinação nostálgica pela estética da cultura retrô do auge dos anos 80 e 90, anos em que o cantor Tim Maia fazia um notável sucesso nas rádios brasileiras. Com um legado indiscutível, Tim Maia é considerado o maior cantor brasileiro de todos os tempos pela revista Rolling Stone Brasil.

Englobando esses dois conceitos é possível oferecer uma experiência nostálgica e imersiva ao usuário, ao jogar Tim Maia Vapor Wave Ultimate Candy Crush Edition é possível sentir todo o talento do cantor em uma combinação perfeita com gênero musical contemporâneo, carregando as canções inesquecíveis: Azul da Cor do Mar e Ela Partiu com uma carga emocional inexplicável, em conjunto com o famoso jogo mobile Candy Crush ao fundo.

Como o jogo funciona

O jogo possui um funcionamento bem semelhante ao Candy Crush Saga, jogo que possui mais de 27 milhões de downloads atualmente na Play Store. O jogador se depara inicialmente com uma interface simples que exibe uma matriz completamente preenchida por formas geométricas e um placar com duas informações: a pontuação e o número de movimentos restantes. A matriz representa os doces os quais o jogador deve realizar combinações para pontuar. Para concluir uma combinação o jogador deve conseguir formar uma sequência de 3 ou mais doces do mesmo tipo. Para realizar as sequências o jogador poderá mover apenas os doces vizinhos: se o movimento resultar em uma sequência válida de 3 doces, a pontuação subirá 100 pontos, se for uma sequência de 4 doces: 200 pontos, 5 doces: 300 pontos, ou seja, quanto mais doces o jogador conseguir combinar de uma vez,

mais pontos ele receberá pela sequência. Caso o movimento for realizado de maneira incorreta, ou não resultar em uma combinação válida, o jogador será penalizado em -10 pontos.

Ao todo são 10 jogadas, portanto o jogador deverá realizá-las da melhor forma possível, procurando fazer as melhores combinações sempre. A maior pontuação é armazenada em um arquivo e sempre aparece na tela final, a qual exibe ao jogador sua pontuação final e o recorde registrado.

Estruturas de Dados

Todo o jogo se baseia em uma Matriz principal, essa possui 6 linhas e 9 colunas. Cada célula dessa matriz recebe uma nova estrutura de dados como conteúdo: o candy. Existem 5 tipos diferentes de candy no jogo, cada um com sua forma e cor únicas. A estrutura candy possui 3 informações: tipo, cor e active.

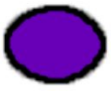
```
typedef struct Candy{
  int type;
  int active;
  ALLEGRO_COLOR cor;
} Candy;
```

- O tipo pode ser 1, 2, 3, 4 ou 5.
- Active varia entre 0 e 1 ao longo do programa e define basicamente se o candy está ou não em uma sequência.
- Cor é a coloração que o candy vai aparecer na tela, definida por uma escala RGB.

Se o candy for do tipo 2, ele será um quadrado vermelho.

Se o candy for do tipo 1, ele será um triângulo laranja.

Se o candy for do tipo 4, ele será um quadrado arredondado verde.



Se o candy for do tipo 3, ele será uma elipse azul.

Se o candy for do tipo 5, ele será um triângulo amarelo.

Como o código funciona

O código começa com as rotinas de inicialização, as quais são responsáveis por iniciar todo tipo de estrutura que será utilizada posteriormente pelo Allegro, entre elas estão: o display, a fila de eventos, os samples (sons), o timer, as imagens e as fontes de texto. Todas

essas estruturas precisam ser carregadas corretamente para que o programa rode, caso contrário mensagens de erro aparecerão e o programa não iniciará.

Em seguida a seed da função rand é definida com a função time, essa combinação faz com que toda vez que o jogo é inicializado a matriz inicial seja sempre diferente, uma vez que o tempo atual do computador definirá a semente geradora da função rand.

Inicia-se então a primeira matriz e verifica-se se esta possui alguma combinação válida, caso possua uma nova matriz aleatória é gerada e esse processo se repete até que a matriz original não possua combinações já formadas. A partir desse ponto começa a fila de eventos do Allegro, a qual estipula pra cada ação específica do usuário uma determinada consequência.

A fila de eventos em questão possui as seguintes ações:

```
488       if(ev.type == ALLEGRO_EVENT_KEY_DOWN) {
489         if(ev.keyboard.keycode == ALLEGRO_KEY_ESCAPE) {
490           playing = 0;
491         }
```

Se o usuário pressionar a tecla ESC, o jogo é fechado.

```
494       else if(ev.type == ALLEGRO_EVENT_MOUSE_BUTTON_DOWN) {
495         printf("\nclicou em (%d, %d)", ev.mouse.x, ev.mouse.y);
496         getCell(ev.mouse.x, ev.mouse.y, &lin_src, &col_src);
497       }
```

Se o usuário pressionar o botão esquerdo do Mouse registra a célula em que o jogador clicou.

```
499       else if(ev.type == ALLEGRO_EVENT_MOUSE_BUTTON_UP) {
500         printf("\nsoltou em (%d, %d)", ev.mouse.x, ev.mouse.y);
501         getCell(ev.mouse.x, ev.mouse.y, &lin_dst, &col_dst);
502         if (lin_src!=lin_dst || col_src!=col_dst) {
503           swap(lin_src, col_src, lin_dst, col_dst, &score, &wrongSound);
504           moves = moves-1;
505             while (verifyCombination()!=0) {
506               destroyCandy(&score);
507               al_play_sample(moveSound, 1.0, 0.0, 1.0, ALLEGRO_PLAYMODE_ONCE,NULL)
508               //Fazer a animação do destroy
509               draw_scenario(display);
510               al_flip_display();
511               pause(timer, 0.8);
512               dropCandy();
513               replaceCandy();
514               draw_scenario(display);
515               al_flip_display();
```

Se o usuário soltar o botão esquerdo do Mouse, o programa executará uma série de funções

que realizarão a validação do movimento, a troca dos doces, a identificação das sequências formadas, a animação de explosão das sequências, a eliminação das mesmas e a substituição por novos doces para repovoar a matriz. Cada função será explicada com mais detalhes na parte específica de funções.

```
521       else if(ev.type == ALLEGRO_EVENT_TIMER) {
522           draw_scenario(display);
523           al_flip_display();
524       }
```

Se o evento for o timer atualiza-se o cenário através do objeto display.

```
526       else if(ev.type == ALLEGRO_EVENT_DISPLAY_CLOSE) {
527         playing = 0;
528       }
```

Se o usuário pressionar o botão X de saída, o jogo fecha.

Essa fila de eventos possibilita a funcionalidade completa do jogo, portanto o usuário poderá executar suas ações na matriz de doces e isso resultará em diferentes resultados dependendo de suas ações. Ao realizar 10 jogadas o jogo abre a tela final e exibe a pontuação final do jogador. Ao registrar a pontuação final, o programa compara com o recorde (registrado no arquivo record.txt). Caso a pontuação final tenha superado o recorde uma mensagem informativa aparecerá indicando ao jogador que ele conseguiu bater o recorde. Caso contrário o jogo apenas exibe a pontuação final e o recorde registrado.

O jogo se encerra quando o usuário pressiona ESC na tela final.

Funções

```
void pause(ALLEGRO_TIMER *timer, float seconds)
```

Essa função possibilita que haja delay durante o código, pausando o timer pelo tempo desejado.

```
int verifyCombination(){
```

Função que verifica se há combinações na Matriz atual, esta retorna o número de combinações encontradas.

```
void replaceCandy(){
```

Essa função realiza a reposição de todos os doces que formaram uma sequência por outros de tipos definidos aleatoriamente.

```
void dropCandy(){
```

Função que realiza a troca, de baixo para cima, de todos os doces que já formaram uma sequência pelos doces que ainda não formaram. Essa ação resulta em um efeito de queda dos doces, dando um efeito visual de que os doces que formaram a sequência desaparecem e outros doces caem no lugar.

```
void destroyCandy(int* score){
```

Função que realiza a animação da destruição da sequência formada pelo usuário. Ao identificar a sequência ela realça a mesma com a cor branca, esse efeito combinado com um delay gerado pela função pause, completa a animação de destruição. Essa função também computa a pontuação pela formação das sequências.

```
void initCandies(){
```

Função de preenchimento da matriz com doces aleatórios. Cada doce segue uma estrutura já explicada na sessão Estrutura de Dados desse documento.

```
int getXCoord(int col){
  return col*COL_W;
  }
int getYCoord(int lin){
  return lin*LIN_W;
  }
```

Funções que computam os inícios das coordenadas das células da matriz.

```
void desenhaCandy(Candy c,int lin,int col){
```

Função que desenha os doces na Matriz principal através de funções presentes na biblioteca Allegro, essas desenham formas geométricas nas coordenadas corretas de cada célula da matriz com base nas funções getXCoord e getYCoord.

```
int newRecord(int score, int *record) {
```

Função que lê o recorde atual no arquivo record.txt e realiza a troca caso um novo record seja registrado.

```
void endGame(ALLEGRO_TIMER **timer, int* playing)
```

Função que encerra o jogo após todos os 10 movimentos serem realizados. Essa função é responsável por exibir a tela final e escrever nela a pontuação final do jogador e o recorde atual.

```
void draw_scenario(ALLEGRO_DISPLAY *display) {
```

Função responsável por atualizar o placar de pontuação e de movimentos já realizados, além de chamar a função desenhacandy para cada célula da Matriz individualmente, atualizando também todos os doces.

```
void getCell(int x, int y, int *lin, int *col){
```

Função que identifica quais são as coordenadas da célula que sofreu a ação do usuário.

```
void swap(int lin_src,int col_src,int lin_dst,int col_dst,int *score, ALLEGRO_SAMPLE **wrongSound){
```

Função primordial para o funcionamento do jogo. Ela é responsável por trocar os doces clicados pelo usuário de lugar. Essa função valida a ação do jogador identificando se foram realizadas apenas trocas entre doces vizinhos, além de permitir apenas trocas que resultem em combinações válidas, caso o usuário realize uma troca inválida ou que não resulte em combinações válidas ele será penalizado com – 10 pontos.