Dancing Plates - TP

Eric de Souza Botelho

Conteúdo

Se o código não couber na imagem, cheque o código na íntegra no github: https://www.google.com.

1 Estruturas

1.1 Personagem

O player possui posições x e y, velocidade dada por velocity, pontuação dada pela variável score, vidas dada pela variável lives, variáveis que indiquem se ele está se movimentando para a esquerda ou direita (left e right), variável que indica se o bitmap será desenhado refletido ou não, balancing indica se o player está equilibrando algum prato, record indica a melhor pontuação do jogador (por execução), Bounding_Box é um vetor que representa um retângulo de colisão do jogador, e sprite a imagem que ele está utilizando. Além disso ele possui effect e time, que representam o tipo de efeito ativo e a duração do mesmo.

```
typedef struct Character{
  float x;
  float y;
  unsigned int score;
  int lives;
  int right;
  int left;
  int rotated;
  int balancing;
  int velocity;
  int time;
  int effect;
  unsigned int record;
  int Bounding_Box[4];
  ALLEGRO_BITMAP *sprite;
} Player;
```

1.2 Pratos

Os pratos possuem posições x e y, time a variável que representa o tempo em segundos que o prato está no jogo, flag representa se o prato já foi spawnado ou não, spin representa se o prato está rodando ou não, falling representa se o prato está caindo ou não, balanced representa se o prato está sendo equilibrado ou não e Bounding_Box, como no personagem, é um vetor que representa um retângulo de colisão e image a imagem corrente.

```
typedef struct Plate{
  float x;
  float y;
  int time;
  int flag;
  int spin;
  int falling;
  int rotated;
  int balanced;
  int Bounding_Box[4];
```

```
  ALLEGRO_BITMAP *image;
} Plate;
```

1.3 Colunas

São estruturas simples que representam apenas as colunas que carregam os pratos, possuem posições x e y, e imagem dada pelo bitmap image.

```
typedef struct Column{
  float x;
  float y;
  ALLEGRO_BITMAP *image;
} Column;
```

1.4 Poderes

Representam poderes,possuem posições x e y, flag que representa se o poder apareceu, tipo de efeito, caixa de colisão e imagem.

```
typedef struct PowerUp{
  float x;
  float y;
  int flag;
  int effect;
  int Bounding_Box[4];
  ALLEGRO_BITMAP *image;
} PowerUp;
```

2 Funções

2.1 Inicializadoras

Apenas preenchem inicialmente os campos relevantes de cada estrutura para que não fique com lixo de memória. Note que a maioria das posições tento colocar em função do comprimento e altura da aba, mudar a resolução quebra o jogo.

2.1.1 Inicia jogador

```
void initPlayer(Player *player){
  player->x = 0.388*SCREEN_W;
  player->y = 0.69*SCREEN_H;
  player->score = 0;
  player->lives = 4;
  player->right = 0;
  player->left = 0;
  player->rotated = 0;
  player->balancing = 0;
  int i;
```

```
  for (i = 0; i < 4; i++){
    player->Bounding_Box[i] = 0;
  }
  player->sprite = al_load_bitmap("./Sprites/sprite1.png");
}
```

2.1.2 Inicia pratos e colunas

i é o índice do vetor coluna e prato correspondente.

```
void initScenario(Column *columns, Plate *plates, int i){
  columns->x = (SCREEN_W/(NUM_PLATES+1)) + (i*SCREEN_W/(NUM_PLATES+1));
  columns->y = 0.32*SCREEN_H;
  plates->x = columns->x - (0.0260*SCREEN_W);
  plates->y = columns->y - (0.0225*SCREEN_H);
  plates->time = 0;
  plates->flag = 0;
  plates->spin = 0;
  plates->rotated = 0;
  plates->falling = 0;
  plates->balanced = 0;
  for(i = 0; i < 4; i++){
    plates->Bounding_Box[i] = 0;
  }
  plates->image = al_load_bitmap("./Sprites/plate.png");
  columns->image = al_load_bitmap("./Sprites/column.png");
}
```

2.1.3 Inicia poderes

```
void initPowerUp(PowerUp *powerup){
  powerup->image = al_load_bitmap("./PowerUp/speed.png");
  powerup->x = RandBetween(0, SCREEN_W - GetWidth(powerup->image));
  powerup->y = 0;
  powerup->flag = 0;
  powerup->effect = 1;
  int i;
  for (i = 0; i < 4; i++){
    powerup->Bounding_Box[i] = 0;
  }
}
```

2.2 Spawners

Spawnam na tela efetivamente determinadas estruturas, 0.6 é um fator de escala pois as imagens de sprite que arrumei ficaram grandes comparadas a resolução.

2.2.1 Spawna Jogador

```
void SpawnPlayer(Player *player){
  al_draw_scaled_bitmap(player->sprite, 0, 0, GetWidth(player->sprite),
                        GetHeight(player->sprite),player->x, player->y,
                        GetWidth(player->sprite)*0.6,
                        GetHeight(player->sprite)*0.6,
                        player->rotated);
  if(player->rotated == 1) player->Bounding_Box[0] = player->x;
  if(player->rotated == 0) player->Bounding_Box[0] = player->x +
  GetWidth(player->sprite)*0.6;
  player->Bounding_Box[1] = player->y;
  player->Bounding_Box[2] = player->Bounding_Box[0] + 6;
  player->Bounding_Box[3] = player->Bounding_Box[1] + 6;
}
```

2.2.2 Spawna Colunas

```
void SpawnColumns(Column *columns){
  for (int i = 0; i < NUM_PLATES; i++){
    al_draw_scaled_bitmap(columns[i].image, 0, 0, GetWidth(columns[i].image),
                          GetHeight(columns[i].image),columns[i].x,
                          columns[i].y,
                          GetWidth(columns[i].image)*0.6,
                          GetHeight(columns[i].image)*0.6, 0);
    /*al_draw_scaled_bitmap(plates[i].image, 0, 0, GetWidth(plates[i].image),
                            GetHeight(plates[i].image),plates[i].x,
                            plates[i].y, GetWidth(plates[i].image)*0.6,
                            GetHeight(plates[i].image)*0.6, 0);*/
  }
}
```

2.2.3 Spawna Pratos

O ajuste é para encaixar o sprite de pratos girando na coluna novamente.

```
void SpawnPlates(Plate *plates){
  int i;
  for(i=0; i < NUM_PLATES; i++){
    plates[i].Bounding_Box[0] = plates[i].x;
    plates[i].Bounding_Box[1] = plates[i].y+(0.6*GetHeight(plates[i].image));
    plates[i].Bounding_Box[2] = plates[i].Bounding_Box[0]+(0.6*GetWidth(plate
    plates[i].Bounding_Box[3] = plates[i].Bounding_Box[1]+1;
    float Adjust_y = 0.0;
    float Adjust_x = 0.0;
    if(plates[i].spin == 1){
      Adjust_y = 0.3*GetHeight(plates[i].image);
      if(plates[i].rotated == 0)
```

```
        Adjust_x = 0.035*GetWidth(plates[i].image);
      else
        Adjust_x = -0.042*GetWidth(plates[i].image);
    }
    if (plates[i].flag == 1)
      al_draw_scaled_bitmap(plates[i].image, 0, 0, GetWidth(plates[i].image),
                            plates[i].x-Adjust_x, plates[i].y-Adjust_y, GetW
  }
}
```

2.2.4 Spawna Poderes

O ajuste é para encaixar o sprite de pratos girando na coluna novamente.

```
void SpawnPowerUp(PowerUp *pwup){
  pwup->Bounding_Box[0] = pwup->x;
  pwup->Bounding_Box[1] = pwup->y+(0.1*GetHeight(pwup->image));
  pwup->Bounding_Box[2] = pwup->Bounding_Box[0]+(0.1*GetWidth(pwup->image));
  pwup->Bounding_Box[3] = pwup->Bounding_Box[1]+1;
  al_draw_scaled_bitmap(pwup->image, 0, 0, GetWidth(pwup->image), GetHeight(p
                        pwup->x, pwup->y, GetWidth(pwup->image)*0.1, GetHeigh
}
```

2.3 Troca sprites

Funções chamadas sempre que é necessário uma mudança ou um reload de sprites.

2.3.1 Para o Jogador

```
void ChangeSprite(Player *player){
  al_destroy_bitmap(player->sprite);
  if(player->balancing == 1){
    int choice = rand() % 2;
    if (choice == 0)
      player->sprite = al_load_bitmap("./Sprites/sprite2.png");
    else
      player->sprite = al_load_bitmap("./Sprites/sprite3.png");
  }
  else{
    int choice = rand() % 3;
    if (choice == 0){
      player->sprite = al_load_bitmap("./Sprites/sprite1.png");
    } else if (choice == 1){
      player->sprite = al_load_bitmap("./Sprites/sprite5.png");
    } else if (choice == 2){
      player->sprite = al_load_bitmap("./Sprites/sprite4.png");
    }
  }
}
```

2.3.2 Para os pratos

```
void SpinPlate(Plate *plate){
  int i;
  if (plate->flag == 1){
    if (plate->spin == 1){
      al_destroy_bitmap(plate->image);
      plate->image = al_load_bitmap("./Sprites/image.png");
    }
    if (plate->spin == 0){
      al_destroy_bitmap(plate->image);
      plate->image = al_load_bitmap("./Sprites/plate.png");
    }
   }
}
```

2.3.3 Para as colunas

```
void ReloadColumn(Column *column){
  al_destroy_bitmap(column->image);
  column->image = al_load_bitmap("./Sprites/column.png");
}
```

2.4 Atualiza Posicao

Atualiza posição do jogador

```
void UpdatePos(Player *player, ALLEGRO_TIMER *timer, ALLEGRO_SAMPLE_INSTANCE
  if (player->right == 1){
    al_play_sample_instance(Walk);
    player->x += player->velocity;
    if(al_get_timer_count(timer)%(int)(FPS/12) == 0) ChangeSprite(player);
    if (player->x >= 0.95*SCREEN_W) player->x = 0.95*SCREEN_W;
  }
  if (player->left == 1){
    al_play_sample_instance(Walk);
    player->x -= player->velocity;
    if(al_get_timer_count(timer)%(int)(FPS/12) == 0) ChangeSprite(player);
    if (player->x <= 0) player->x = 0;
  }
  if (player->balancing == 1){
    if(al_get_timer_count(timer)%(int)(FPS/12) == 0) ChangeSprite(player);
  }
}
```

2.5 Troca cor

Função chamada quando é preciso trocar a cor dos bitmaps de certas estruturas.

2.5.1 Para as colunas

```
void ChangeColor_Column(Column *column, int R, int G, int B){
  al_lock_bitmap(column->image, ALLEGRO_PIXEL_FORMAT_RGBA_8888, ALLEGRO_LOCK_
  al_set_target_bitmap(column->image);
  int x, y;
  for(x=0; x<GetWidth(column->image); x++){
    for(y=0; y<GetHeight(column->image); y++){
      ALLEGRO_COLOR pixel = al_get_pixel(column->image, x, y);
      unsigned char r, g, b, a;
      al_unmap_rgba(pixel, &r, &g, &b, &a);
      if(r >= 0 && r <= 61 && g >= 0 && g <= 40 && b >= 132 && b <= 165)
        al_put_pixel(x, y, al_map_rgba(R, G, B, a));
      else
        al_put_pixel(x, y, al_map_rgba(r, g, b, a));
    }
  }
  al_unlock_bitmap(column->image);
  al_set_target_backbuffer(al_get_current_display());
}
```

2.5.2 Para os pratos

Essa função baseia-se no tempo que os pratos estão ativos para determinar a cor a ser mostrada, quanto mais tempo e mais perto de cair, a cor tende a ficar vermelha, quando o tempo ativo é pouco ou o prato está sendo equilibrado, a cor tende a voltar a padrão do sprite.

```
void ChangeColor_Plate(Plate *plate){
  SpinPlate(plate);
  al_lock_bitmap(plate->image, ALLEGRO_PIXEL_FORMAT_RGBA_8888, ALLEGRO_LOCK_F
  al_set_target_bitmap(plate->image);
  int x, y;
  for(x=0; x<GetWidth(plate->image); x++){
    for(y=0; y<GetHeight(plate->image); y++){
      ALLEGRO_COLOR pixel = al_get_pixel(plate->image, x, y);
      unsigned char r, g, b, a;
      al_unmap_rgba(pixel, &r, &g, &b, &a);
      if(r >= 99 && g >= 18 && b >= 127 && r <= 255 && g <= 255 && b <= 255){
        int R, G, B;
        R = r + 8*plate->time;
        G = g - 8*plate->time;
        B = b - 8*plate->time;
        if(R > 255) R = 255;
        if(G < 0) G = 0;
        if(B < 0) B = 0;
        al_put_pixel(x, y, al_map_rgba(R, G, B, a));
      }
      else
        al_put_pixel(x, y, al_map_rgba(r, g, b, a));
```

```
    }
  }
  al_unlock_bitmap(plate->image);
  al_set_target_backbuffer(al_get_current_display());
}
```

2.5.3 Para as colunas no sprite dos pratos

Basicamente no sprite de pratos girando existe pedaço de uma coluna, essa função existe para pintar essa parte, veja as imagens abaixo para compreender:

Figura 1: Imagem do prato parado

Figura 2: Imagem do prato girando

```
void ChangeColor_ColPlate(Plate *plate, int R, int G, int B){
  al_lock_bitmap(plate->image, ALLEGRO_PIXEL_FORMAT_RGBA_8888, ALLEGRO_LOCK_F
  al_set_target_bitmap(plate->image);
  int x, y;
  for(x=0; x<GetWidth(plate->image); x++){
    for(y=0; y<GetHeight(plate->image); y++){
      ALLEGRO_COLOR pixel = al_get_pixel(plate->image, x, y);
      unsigned char r, g, b, a;
      al_unmap_rgba(pixel, &r, &g, &b, &a);
      if((r <= 93 && r >= 0 && g <= 85 && g >= 0 && b <= 192 && b >= 130) ||
        al_put_pixel(x, y, al_map_rgba(R, G, B, a));
      else
        al_put_pixel(x, y, al_map_rgba(r, g, b, a));
    }
  }
  al_unlock_bitmap(plate->image);
  al_set_target_backbuffer(al_get_current_display());
}
```

2.6 Detectores de colisão

Retornam se houve colisão ou não.

2.6.1 Entre Jogador e Coluna

Existe uma pequena caixa de colisão da coluna também para que haja certa tolerância.

```
int CheckCollision(Player *player, Column *columns, Plate *plates){
  int i;
  for(i = 0; i < NUM_PLATES; i++){
    if(player->Bounding_Box[2] >= columns[i].x && player->Bounding_Box[0] < c
        if(player->balancing == 0 && plates[i].flag == 1 && plates[i].falling
          plates[i].balanced = 1;
          ChangeColor_Column(&columns[i], 255, 0, 0);
          if(plates[i].spin == 1)
            ChangeColor_ColPlate(&plates[i], 255, 0, 0);
        }
        //ChangeColor_Plate(&plates[i], 0, 0, 0);
        return 1;
    }
  }
  return 0;
}
```

Caixa de colisão do jogador(quadrado preto na imagem) em ação:

(a) Personagem colidindo com a coluna

(b) Caixa de colisão

Figura 3: Colisão em ação

2.6.2 Entre Jogador e Prato(caindo)

```
int CheckColPlates(Player *player, Plate *plates){
  if(plates->Bounding_Box[2] >= player->x && plates->Bounding_Box[0] <= playe
    return 1;
  }
  return 0;
}
```

Caixa de colisão dos pratos (retângulo preto na imagem)

(a) Retângulo de colisão do prato

(b) Retângulo de colisão no prato caindo

Figura 4: Colisão em ação

2.6.3 Entre Jogador e Poder

```
int CheckColPWUP(PowerUp *pwup, Player *player){
  if(pwup->flag == 1)
```

```
    if(pwup->Bounding_Box[2] >= player->x && pwup->Bounding_Box[0] <= player-
      return 1;
  return 0;
}
```

2.7 Poderes

Toma uma ação enquanto um poder está ativo

2.7.1 Poder está caindo

```
void PWEffect(PowerUp *pwup, Player *player){
  if(pwup->flag == 1){
    if(CheckColPWUP(pwup, player) == 1){
      player->effect = pwup->effect;
      player->time = 5;
      ApplyEffect(player);
      al_destroy_bitmap(pwup->image);
      initPowerUp(pwup);
    }
    else if(pwup->y+GetHeight(pwup->image)*0.1 < player->y+GetHeight(player->
      SpawnPowerUp(pwup);
      pwup->y += GRAVITY;
    }
    else{
      al_destroy_bitmap(pwup->image);
      initPowerUp(pwup);
    }
  }
}
```

2.7.2 Efeito do poder

```
void ApplyEffect(Player *player){
  if(player->effect == 1){
    int new_velocity = 2;
    player->velocity = new_velocity;
  }
}
```

2.8 Prato está caindo

Toma uma ação se o prato está caindo, se ele colidir com o jogador o prato é reiniciado, se ele colidir com o chão, o prato é reiniciado e o jogador perde vida. Se nenhum dos dois ocorrer, o prato continua caindo conforme a gravidade.

```
void IsFalling(Plate *plates, Player *player, Column *columns){
  int i;
```

```
  for(i = 0; i < NUM_PLATES; i++){
    if(plates[i].falling == 1){
      if(CheckColPlates(player, &plates[i]) == 1){
        plates[i].x = columns[i].x - (0.0260*SCREEN_W);
        plates[i].y = columns[i].y - (0.0225*SCREEN_H);
        plates[i].time = -RandBetween(10, 20);;
        plates[i].flag = 0;
        plates[i].spin = 0;
        plates[i].rotated = 0;
        plates[i].falling = 0;
        plates[i].balanced = 0;
        al_destroy_bitmap(plates[i].image);
        plates[i].image = al_load_bitmap("./Sprites/plate.png");
      }
      else if(plates[i].y+GetHeight(plates[i].image)*0.6 < player->y+GetHeigh
        plates[i].y += GRAVITY;
        al_destroy_bitmap(plates[i].image);
        plates[i].image = al_load_bitmap("./Sprites/plate_falling.png");
      }
      else{
        plates[i].x = columns[i].x - (0.0260*SCREEN_W);
        plates[i].y = columns[i].y - (0.0225*SCREEN_H);
        plates[i].time = -RandBetween(10, 20);
        plates[i].flag = 0;
        plates[i].spin = 0;
        plates[i].rotated = 0;
        plates[i].falling = 0;
        plates[i].balanced = 0;
        al_destroy_bitmap(plates[i].image);
        plates[i].image = al_load_bitmap("./Sprites/plate.png");
        player->lives--;
      }
    }
  }
}
```

2.9 Utilitárias

2.9.1 Pega a altura de um Bitmap

```
int GetHeight(ALLEGRO_BITMAP *image){
  return al_get_bitmap_height(image);
}
```

2.9.2 Pega a largura de um Bitmap

```
int GetWidth(ALLEGRO_BITMAP *image){
  return al_get_bitmap_width(image);
```

```
}
```

2.9.3 Reflete pratos

Função cuja função é eventualmente refletir o sprite de pratos girando, para causar impressão de movimento.

```
void ReflectPlates(Plate *plates, ALLEGRO_TIMER *timer){
  int i;
  if(al_get_timer_count(timer)%(int)(FPS/4) == 0)
    for(i=0; i<NUM_PLATES; i++){
        if(plates[i].flag == 1){
          if(plates[i].spin == 1)
            if(plates[i].rotated == 0) plates[i].rotated = 1;
            else plates[i].rotated = 0;
        }
    }
  }
```

2.9.4 Confere se precisa atualizar bitmap

```
void CheckReload(Plate *plates, Column *columns){
  int i;
  for(i=0; i<NUM_PLATES; i++){
    if(plates[i].balanced == 1){
      plates[i].balanced = 0;
      ChangeColor_Plate(&plates[i]);
      ReloadColumn(&columns[i]);
    }
  }
}
```

2.9.5 Equilibra pratos

Determina qual o efeito de equilibrar algum prato.

```
void BalanceEffect(Plate *plate){
  plate->time -= 1;
  ChangeColor_Plate(plate);
}
```

2.9.6 Tempo de aparecimento dos pratos

Configura o tempo de aparecimento dos pratos, é um tempo negativo aleatório, quando ele chega a 0, o flag dos pratos vai para 1 e ele aparece na tela.

```
void SetTimeForPlates(Plate *plates){
  int i;
  plates[NUM_PLATES/2].flag = 1;
```

```
  plates[(NUM_PLATES/2) - 1].flag = 1;
  for (i = (NUM_PLATES/2)+1; i < NUM_PLATES; i++){
    plates[i].time = plates[i-1].time - RandBetween(10, 20);
  }
  for (i = (NUM_PLATES/2) - 2; i < NUM_PLATES; i--){
    plates[i].time = plates[i+1].time - RandBetween(10, 20);
  }
}
```

2.9.7 Numero randômico

Determina qual o efeito de equilibrar algum prato.

```
int RandBetween(int min, int max){
  return rand()%(max - min + 1) + min;
}
```

3 Procedimento

3.1 Constantes

```
const unsigned int SCREEN_W = 640; // Don't change
const unsigned int SCREEN_H = 480; // Don't change
const unsigned int FPS = 60;
const unsigned int NUM_PLATES = 10; // Odd numbers not tested
const float GRAVITY = 1.5;
```

3.2 Inicialização do módulo Allegro e da randomicidade

```
srand(time(NULL));
al_init();
al_init_font_addon();
al_init_ttf_addon();
al_init_image_addon();
al_install_keyboard();
al_init_primitives_addon();
al_install_audio();
al_init_acodec_addon();
al_reserve_samples(1);
```

3.3 Inicialização de fontes, display, som, logo, estrutura

FRT_END é uma variável de controle que é responsável por indicar em qual parte do jogo estamos, e a opacidade é apenas uma variável que causa o efeito de piscar o "PRESS START TO PLAY"

```
ALLEGRO_SAMPLE *DTheme = al_load_sample("./Audio/DancingPlates.ogg");
ALLEGRO_SAMPLE *Walk = al_load_sample("./Audio/mariowalking.wav");
ALLEGRO_SAMPLE_INSTANCE *DThemeInstance = al_create_sample_instance(DTheme);
ALLEGRO_SAMPLE_INSTANCE *WalkInstance = al_create_sample_instance(Walk);
al_attach_sample_instance_to_mixer(DThemeInstance, al_get_default_mixer());
al_attach_sample_instance_to_mixer(WalkInstance, al_get_default_mixer());
ALLEGRO_DISPLAY *display = al_create_display(SCREEN_W, SCREEN_H);
al_set_window_position(display, 200, 200);
ALLEGRO_FONT *font = al_load_font("./Atari.ttf", 45, 0);
ALLEGRO_FONT *corp_font = al_load_font("./score.otf", 20, 0);
ALLEGRO_FONT *pixel_font = al_load_font("./AtariSmall.ttf", 20, 0);
ALLEGRO_TIMER *timer = al_create_timer(1.0 / FPS);
ALLEGRO_BITMAP *logo = al_load_bitmap("./logo/Logo2.png");
ALLEGRO_EVENT_QUEUE *event_queue = al_create_event_queue();
al_register_event_source(event_queue, al_get_display_event_source(display));
al_register_event_source(event_queue, al_get_timer_event_source(timer));
al_register_event_source(event_queue, al_get_keyboard_event_source());
float OPACITY = 1.0;
int OPACITY_INCREASING = 0;
int FRT_END = 0;
Player player;
Column columns[NUM_PLATES];
Plate plates[NUM_PLATES];
PowerUp powerup;
player.record = 0;
int NewRecord = 0;
al_start_timer(timer);
```

3.4 Loop principal do jogo

3.4.1 Menu do jogo

```
if(FRT_END == 0){
    al_clear_to_color(al_map_rgb(88, 0, 112));
    al_draw_bitmap(logo, 0.1*SCREEN_W, 0.05*SCREEN_H, 0);
    al_draw_textf(font, al_map_rgba_f(0.0, 0.0, 0.0, OPACITY), 0.15*SCREEN_W,
    if (OPACITY_INCREASING == 0) OPACITY -= 0.01;
    else OPACITY += 0.01;

    if (OPACITY <= 0.01) OPACITY_INCREASING = 1;
    if (OPACITY >= 1.0) OPACITY_INCREASING = 0;
    if(event.type == ALLEGRO_EVENT_KEY_DOWN){
      printf("KEY DOWN\n");
      if(event.keyboard.keycode == ALLEGRO_KEY_SPACE){
        printf("START\n");
       FRT_END = 1;
      }
    }
```

```
}
```

3.4.2 Inicialização das estruturas

```
else if(FRT_END == 1){
    //Initializing player and scenario
    int i;
    initPlayer(&player);
    initPowerUp(&powerup);
    for (i = 0; i < NUM_PLATES; i++){
      initScenario(&columns[i], &plates[i], i);
    }
    SetTimeForPlates(plates);
    FRT_END = 2;
}
```

3.4.3 O Jogo

3.4.3.1 Eventos de teclado

Segurar espaço enquanto estiver colidindo com a coluna causará o equilíbrio do prato. A e D geram movimentação do personagem.

```
else if(FRT_END == 2){
    //Spawn player and scenario, main loop of the game, where the game is run
    switch(event.type){
      case ALLEGRO_EVENT_KEY_DOWN:
        if(event.keyboard.keycode == ALLEGRO_KEY_D){
          player.left = 0;
          player.right = 1;
          player.rotated = 0;
        }
        if(event.keyboard.keycode == ALLEGRO_KEY_A){
          player.right = 0;
          player.left = 1;
          player.rotated = 1;
        }
        if(event.keyboard.keycode == ALLEGRO_KEY_SPACE && player.left == 0 &&
          player.balancing = 1;
        }
        break;
      case ALLEGRO_EVENT_KEY_UP:
        if(event.keyboard.keycode == ALLEGRO_KEY_D){
          player.right = 0;
        }
        if(event.keyboard.keycode == ALLEGRO_KEY_A){
          player.left = 0;
        }
        if(event.keyboard.keycode == ALLEGRO_KEY_SPACE){
```

```
            player.balancing = 0;
            CheckReload(plates, columns);
        }
        break;
```

3.4.3.2 Spawn e atualiza tela

Atualiza e coloca coisas na tela do jogo

```
case ALLEGRO_EVENT_TIMER:
    int i;
    al_play_sample_instance(DThemeInstance);
    al_clear_to_color(al_map_rgb(0, 0, 0));
    al_draw_filled_rectangle(0, 0, SCREEN_W, 0.83*SCREEN_H, al_map_rgb(88, 0,
    al_draw_filled_rectangle(0, 0.861*SCREEN_H, SCREEN_W, (0.861*SCREEN_H)(0.
    al_draw_text(corp_font, al_map_rgb(255, 255, 255), 0.199*SCREEN_W, 0.913*
    al_draw_textf(pixel_font, al_map_rgb(0, 0, 0), 0.659*SCREEN_W, (0.861*SCR
    PWEffect(&powerup, &player);
    if(player.balancing == 1 && CheckCollision(&player, columns, plates) == 0
      player.balancing = 0;
      CheckReload(plates, columns);
    }
    UpdatePos(&player, timer, Walk);
    ReflectPlates(plates, timer);
    SpawnColumns(columns);
    SpawnPlates(plates);
    SpawnPlayer(&player);
    IsFalling(plates, &player, columns);
    //al_draw_filled_rectangle(plates[5].Bounding_Box[0],plates[5].Bounding_B
    //al_draw_filled_rectangle(player.Bounding_Box[0], player.Bounding_Box[1]
```

3.4.3.3 Manipulação de eventos de tempo

Atualiza os timers de cada prato, verifica se algum deles deve começar ou parar de girar(5 segundos no exemplo), verifica se algum dos pratos vai cair(20 segundos no exemplo), muda a cor dos pratos conforme tempo girando, a cada 30 segundos libera um poder.

```
if(al_get_timer_count(timer)%(int)FPS == 0){
    printf("\n%d segundos se passaram\n", (int)al_get_timer_count(timer)/FPS)
    if((int)al_get_timer_count(timer)/FPS % 30 == 0){ //Every 30 seconds, a p
      PowerUP(&powerup);
    }
    if(player.time > 0){
      ApplyEffect(&player);
      player.time -= 1;
    }
    if(player.time == 0 && player.effect != 0){
      player.velocity = 1;
      player.effect = 0;
    }
```

```
    for(i=0; i < NUM_PLATES; i++){
      if(plates[i].balanced == 0 && plates[i].falling == 0){
        plates[i].time += 1;
        if(plates[i].time > 0){
          ChangeColor_Plate(&plates[i]);
        }
      }
      else if(plates[i].balanced == 1 && plates[i].time >= 1 && plates[i].fal
        BalanceEffect(&plates[i]);
      }
      if(plates[i].time == 0) plates[i].flag = 1;
      else if(plates[i].time == 5){
        plates[i].spin = 1;
        ChangeColor_Plate(&plates[i]);
      }
      else if(plates[i].time < 5 && plates[i].time > 0){
        plates[i].spin = 0;
        ChangeColor_Plate(&plates[i]);
      }
      else if(plates[i].time == 20){
        plates[i].falling = 1;
        plates[i].spin = 0;
      }
    }
}
```

3.4.3.4 Pontuação, vidas e fim de jogo

Atualiza os timers de cada prato, aumenta o score, verifica e desenha os quadradinhos de vida, verifica se algum deles deve começar ou parar de girar(5 segundos no exemplo), verifica se algum dos pratos vai cair(20 segundos no exemplo), muda a cor dos pratos conforme tempo girando, a cada 30 segundos libera um poder.

```
if(al_get_timer_count(timer)%(int)(FPS/6) == 0)
    player.score += 10;
for(i=0; i < player.lives; i++)
  al_draw_filled_rectangle((0.125*SCREEN_W)+(16*i)+(16*i), 0.78125*SCREEN_H,
(0.125*SCREEN_W(i+1)*16(i*16), (0.78125*SCREEN_H)+16, al_map_rgb(204, 172, 1
if(player.lives <= 0){
  if(player.score > player.record) player.record = player.score;
 FRT_END = 3;
}
break;
}
}
```

3.4.3.5 Fim de Jogo

O jogo acabou, espaço reseta o game.

```
 else if (FRT_END == 3){
    switch(event.type){
      case ALLEGRO_EVENT_TIMER:
        al_stop_samples();
        al_draw_filled_rectangle(SCREEN_W/4, SCREEN_H/4, SCREEN_W/4+SCREEN_W/
al_map_rgb(0, 0, 0));
        al_draw_text(pixel_font, al_map_rgb(255, 255, 255), SCREEN_W/2-48, SC
        al_draw_textf(pixel_font, al_map_rgb(255, 255, 255), SCREEN_W/2+90, S
IS: %d", player.record);
        al_draw_text(pixel_font, al_map_rgb(255, 255, 255), SCREEN_W/2+115, S
TO TRY AGAIN");
        break;
      case ALLEGRO_EVENT_KEY_DOWN:
        if(event.keyboard.keycode == ALLEGRO_KEY_SPACE){
          FRT_END = 1;
          al_set_timer_count(timer, 0);
        }
        break;
    }
}

al_flip_display();

}
```

3.4.4 Desalocação de memória

```
al_destroy_bitmap(logo);
al_destroy_font(font);
al_destroy_font(corp_font);
al_destroy_font(pixel_font);
al_destroy_sample(DTheme);
al_destroy_timer(timer);
al_destroy_display(display);
al_destroy_event_queue(event_queue);

return 0;
```

4 Como jogar ?

No menu inicial pressione espaço para iniciar o jogo, para controlar o personagem utilize as teclas A para se deslocar a esquerda e D para se deslocar a direita, para equilibrar um prato chegue próximo a coluna e segure o espaço (segurar o espaço antes e chegar próximo a coluna não equilibra o prato). Sua pontuação é proporcional ao tempo em que está vivo, tente manter os pratos equilibrados e quando eles caírem tente os pegar para não perder vida. A cada 30 segundos é spawnado um poder que aumenta sua velocidade de movimento por um breve período.