

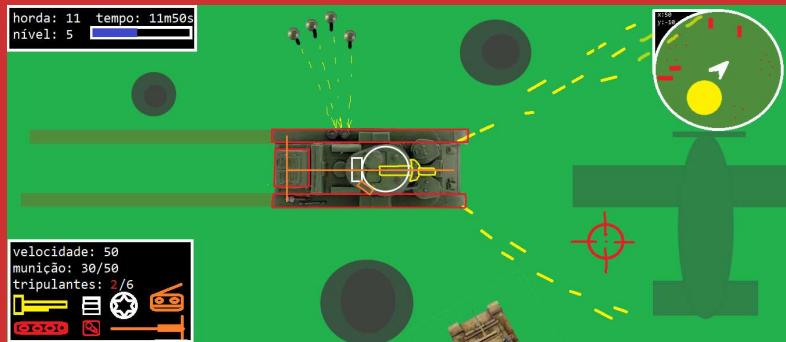


BELLICUS APRESENTAÇÃO FINAL

BRUNO, ERALDO E MARCOS

EXPECTATIVA X RESULTADO

GAMEPLAY



EXPECTATIVA X RESULTADO

FIM DE JOGO



VOCÊ
MORREU

Pontuação: 41
Tempo vivo: 26s

ARQUITETURA GERAL

ARQUIVOS:

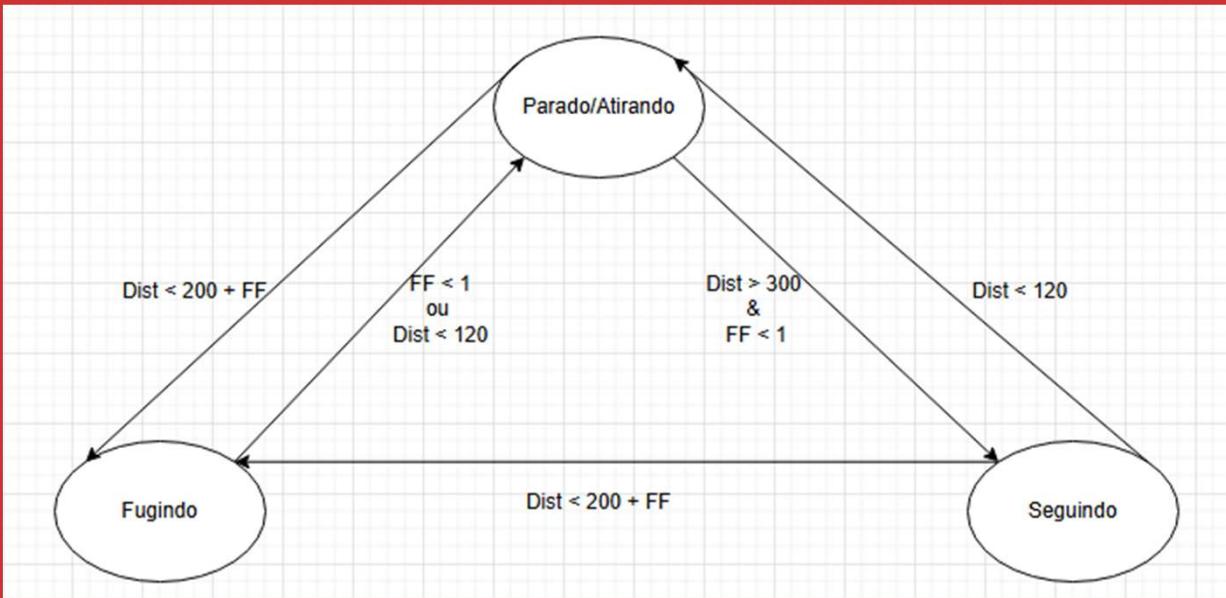
- **ARITMÉTICA** **COPERAÇÕES MATEMÁTICAS BÁSICAS**
- **LÓGICA** **CEVALUAÇÃO DE CONDIÇÕES**
- **COLISÕES** **CVERIFICAÇÃO DE COLISÕES**
- **QUADTREE** **CATURALMENTE INUTILIZADO**
- **ESPERA CRUX** **_WRITEVENTTIMEOUTCOUNT**
- **ESTADOS** **CESTADOS DAS MÁQUINAS DE ESTADOS**
- **ESTRUTURAS** **CSTRUCTS DO JOGO**
- **GLOBAL** **CVARIÁVEIS GLOBAIS**
- **INFO** **CFUNÇÃO PARA PRINTAR TEXTO FORMATADO NA TELA**
- **MENU** **CMENU DO JOGO**
- **SAVE CHIP** **: SALVAR O JOGO**
- **MAIN**

ARQUITETURA GERAL

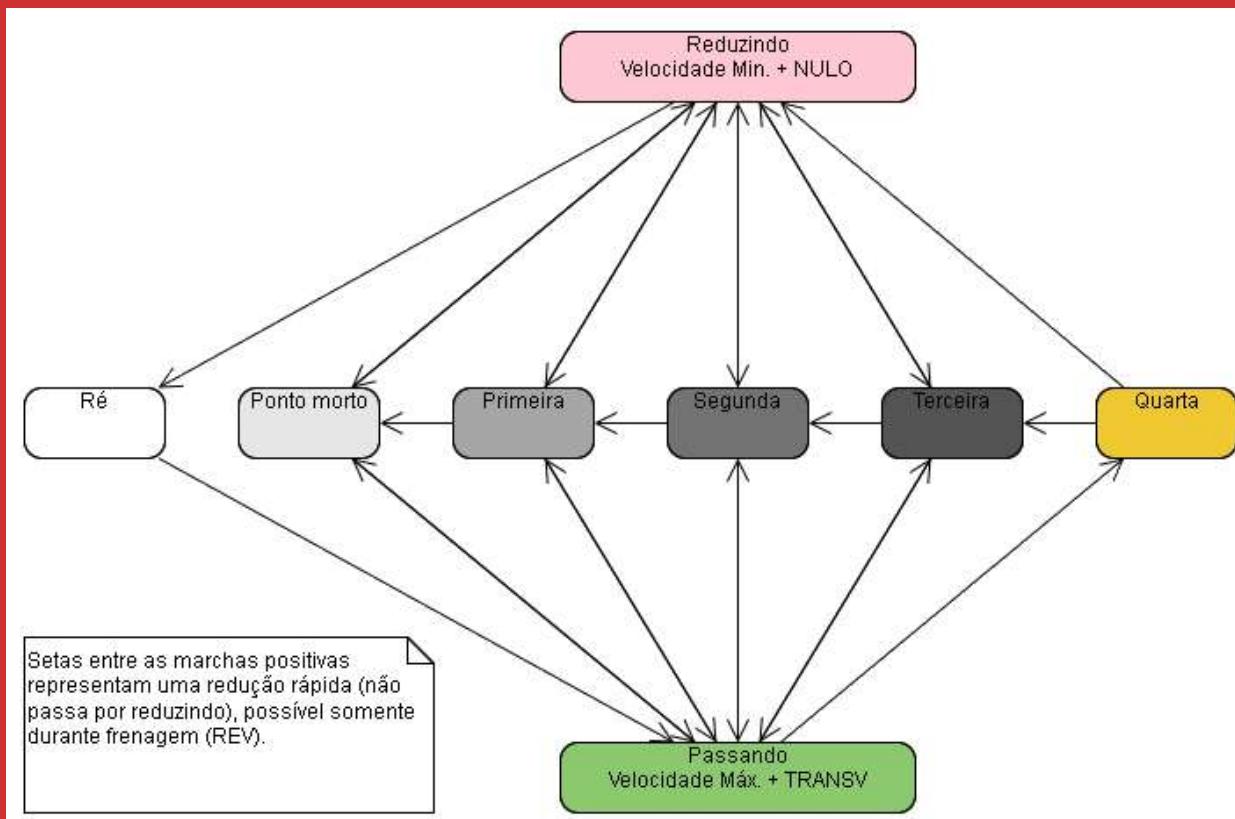
ARQUIVOS:

- 1-19 (19 linhas): header
- 23-163 (141 linhas): funções da main
- 166-182 (17 linhas): inicialização do SDL2, SDL_Mixer e variaveis essenciais
- 186-217 (32 linhas): carregamento de texturas e sons
- 223-352 (130 linhas): inicialização de variaveis, arrays de objetos, timers e mapa
- 356-469 (114 linhas): coleta de input
- 472-1138 (667 linhas): atualização das variaveis, criação e destruição de objetos e renderização
- 1142-1177 (36 linhas): destruição de texturas e sons, finalização do SDL2 e SDL_Mixer
- 1-1178 (1178 linhas): amor e dedicação

MÁQUINAS DE ESTADOS SOLDADOS

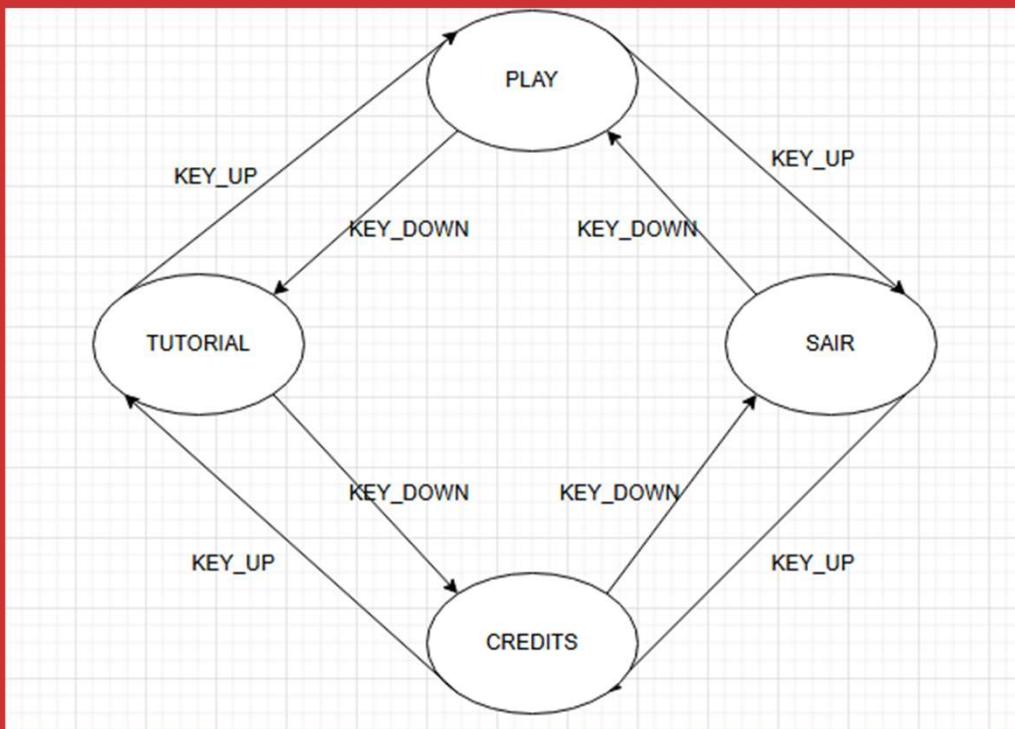


MÁQUINAS DE ESTADOS MARCHA



MÁQUINAS DE ESTADOS

MENU



TRECHOS INTERESSANTES METRALHADORA

```
//metralhadora move
SDL_FPoint ponta_da_metra_chassi;
if(metra && SDL_GetTicks()-ultimoDisparoMetraChassi >= TEMPO_DE_RECARGA_METRA/VELOCIDADE_DE_RECARGA_METRA){
    ponta_da_metra_chassi = somar(somar(local, rotacionar(zero, metra_chassi_offset, angulo)), mover(14, angulo_metra));

    //centro do flash da metralhadora
    base_flash = somar(escalonar((SDL_FPoint){-5,-15},zoom),centro_flash_absoluto);
    centro_flash = escalonar((SDL_FPoint){5,15},zoom);

    for (int s = 0; s < nSoldados; s++) {
        if (nBalasMetra < 100) {
            double anguloDeDisparo = anguloEntrePontos(batalhao[s].local,ponta_da_metra_chassi);
            int noCampoDeVisao = anguloDentroIntervalo(anguloDeDisparo,angulo-45,angulo+45),
                noAlcance = distanciaEntrePontos(batalhao[s].local,ponta_da_metra_chassi) < distanciaEntrePontos(zero,(SDL_FPoint){MWIDTH+50,MHEIGHT+50});
            if (noCampoDeVisao && noAlcance) {
                ultimoDisparoMetraChassi = SDL_GetTicks();
                angulo_metra = anguloDeDisparo;
                projeta newtiro = {ponta_da_metra_chassi,
                    3000,
                    0,
                    angulo_metra-3+rand()%7,
                    2000,
                    1,
                    0};
                bullet[nBalasMetra++] = newtiro;
                if (nParticulas < 299) {
                    particula newpart = {base_flash,
                        SDL_GetTicks(),
                        16,
                        1,
                        0};
                    marcos[nParticulas++] = newpart;
                    newpart = (particula) {base_cup,
                        SDL_GetTicks(),
                        16,
                        2,
                        0};
                    marcos[nParticulas++] = newpart;
                }
                Mix_PlayChannel(-1,disp_metralhadora,0);
            }
        }
    }
}
```

TRECHOS INTERESSANTES SPAWN SOLDADO

```
//spawn e atualização de soldados
int s1,s2;
if (nSoldados < 96 && SDL_GetTicks()-ultimoSpawn >= esperaPorInimigo) {
    ultimoSpawn = SDL_GetTicks();
    SDL_FPoint coord;
    if (rand()%2 == 0) {
        //coordenada fora dos limites superior e inferior da tela
        coord = (SDL_FPoint){rand()%(WIDTH+201)-MWIDTH-100,rand()%2*(HEIGHT+201)-MHEIGHT-100};
    } else {
        //coordenada fora dos limites esquerdo e direito da tela
        coord = (SDL_FPoint){rand()%2*(WIDTH+201)-MWIDTH-100,rand()%(HEIGHT+201)-MHEIGHT-100};
    }
    coord = somar(coord,local);
    for (s1 = 0; s1 < 5; s1++) {
        SDL_FPoint var = {rand()%65,rand()%65};
        soldado newsd = {somar(coord,var),
                         1,
                         rand()%4,
                         0,
                         0,
                         (SDL_FPoint){rand()%15,rand()%15}};
        batalhao[nSoldados++] = newsd;
    }
}
for (s1 = 0; s1 < nSoldados; s1++) {
    if (checkarVida(batalhao,s1,&nSoldados,local,angulo,sangue,velocidade,&nSangue,ren, &soldadosmortos)) {
        continue;
    }
    atualizarPosicaoSoldado(&batalhao[s1],local);
    for (s2 = s1+1; s2 < nSoldados; s2++) {
        corrigirColisao(&batalhao[s1],&batalhao[s2]);
    }
    renderizarSoldado(ren,soldados,batalhao[s1],local,centro_tanque,zoom);
}
```

TRECHOS INTERESSANTES

TELA DE MORTE

```
void desenhar_tela_de_morte(SDL_Renderer *ren, SDL_Texture *tex_morte, int width, int height) {
    if (flash_alpha > 0) {
        SDL_SetRenderDrawColor(ren, 255, 255, 255, flash_alpha);
        SDL_RenderFillRect(ren, NULL);
        flash_alpha -= 5;
        if (flash_alpha < 0) flash_alpha = 0;
        return;
    }

    SDL_SetRenderDrawColor(ren, 0, 0, 0, 180);
    SDL_FRect overlay = {0, 0, (float)width, (float)height};
    SDL_RenderFillRectF(ren, &overlay);

    if (fade_morte < 255)
        fade_morte += 2;
    if (zoom_morte < 1.0f)
        zoom_morte += 0.005f;

    float pulsar = 0.7f + 0.3f * sin(SDL_GetTicks() * 0.004f);
    Uint8 alpha_final = (Uint8)(fade_morte * pulsar);

    SDL_SetTextureAlphaMod(tex_morte, alpha_final);
    SDL_FRect dst = {(width - width * zoom_morte)/2,
                     (height - height * zoom_morte)/2,
                     width * zoom_morte,
                     height * zoom_morte};
    SDL_RenderCopyExF(ren, tex_morte, NULL, &dst, 0, NULL, SDL_FLIP_NONE);
}
```

OBRIGADO