Trabalho Prático de AEDS I 2022-2

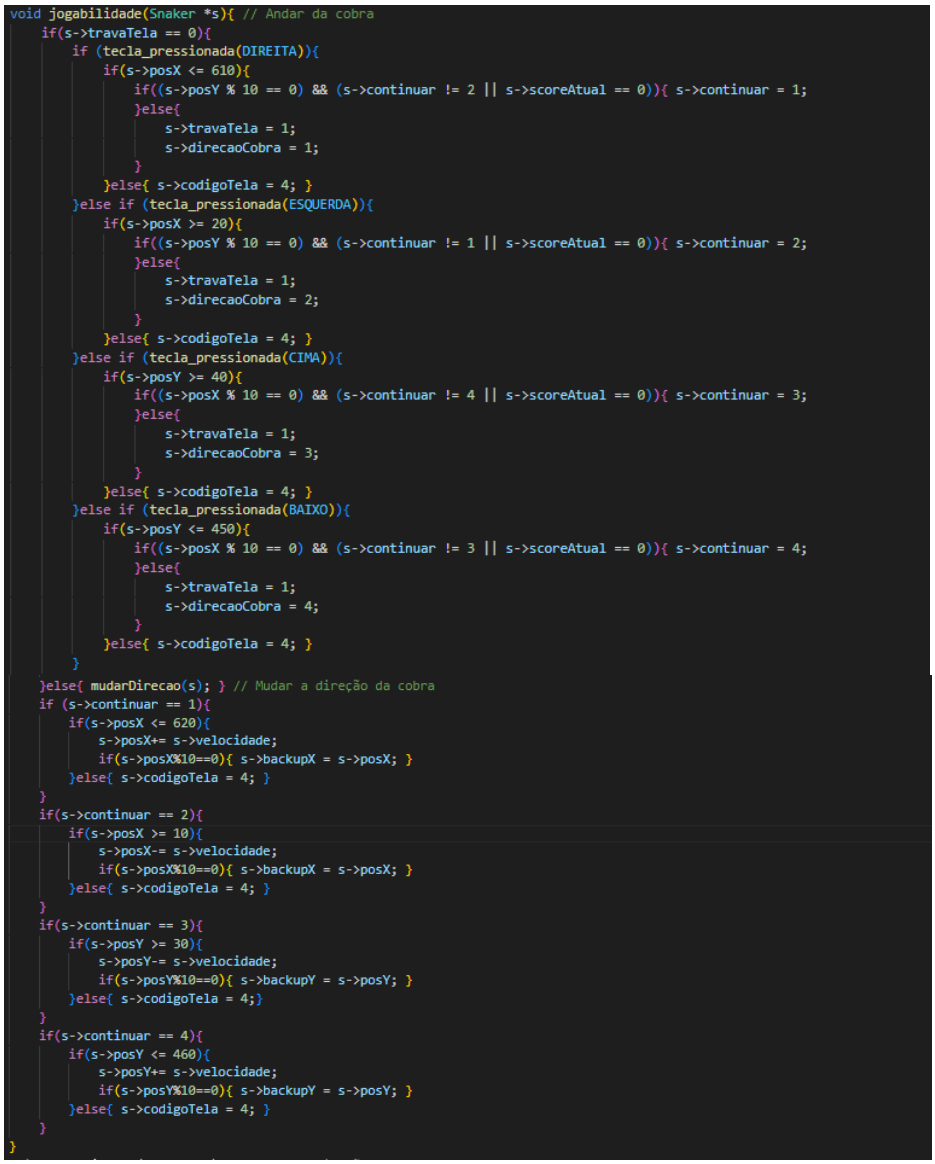
Implementação do Jogo Snake utilizando a biblioteca *jogo.h*

Messias Feres Curi Melo

Documentação

1. Movimento da cobra

O código que desenvolvi utiliza matriz, struct e modulo para a movimentação da cobra, nas imagens a seguir pode ser observado as funções principais para a execução do movimento da cobra, onde na tela main.c chamarei a função 'jogabilidade' mandando o struct Snaker como parâmetro para fornecer os dados e modifica-los, em seguida é feito um tratamento para verificar se o pixel do momento é um divisor de 10 ou não, pois ela pode andar em não divisores, mas só pode mudar de direção ou comer uma fruta em divisores de 10, já que ela mesma tem o tamanho 10 por 10.



```
void jogabilidade(Snaker *s){ // Andar da cobra
    if(s->travaTela == 0){
        if (tecla_pressionada(DIREITA)){
            if(s->posX <= 610){
                if((s->posY % 10 == 0) && (s->continuar != 2 || s->scoreAtual == 0)){ s->continuar = 1;
                }else{
                    s->travaTela = 1;
                    s->direcaoCobra = 1;
                }
            }else{ s->codigoTela = 4; }
        }else if (tecla_pressionada(ESQUERDA)){
            if(s->posX >= 20){
                if((s->posY % 10 == 0) && (s->continuar != 1 || s->scoreAtual == 0)){ s->continuar = 2;
                }else{
                    s->travaTela = 1;
                    s->direcaoCobra = 2;
                }
            }else{ s->codigoTela = 4; }
        }else if (tecla_pressionada(CIMA)){
            if(s->posY >= 40){
                if((s->posX % 10 == 0) && (s->continuar != 4 || s->scoreAtual == 0)){ s->continuar = 3;
                }else{
                    s->travaTela = 1;
                    s->direcaoCobra = 3;
                }
            }else{ s->codigoTela = 4; }
        }else if (tecla_pressionada(BAIXO)){
            if(s->posY <= 450){
                if((s->posX % 10 == 0) && (s->continuar != 3 || s->scoreAtual == 0)){ s->continuar = 4;
                }else{
                    s->travaTela = 1;
                    s->direcaoCobra = 4;
                }
            }else{ s->codigoTela = 4; }
        }
    }else{ mudarDirecao(s); } // Mudar a direção da cobra
    if (s->continuar == 1){
        if(s->posX <= 620){
            s->posX+= s->velocidade;
            if(s->posX%10==0){ s->backupX = s->posX; }
        }else{ s->codigoTela = 4; }
    }
    if(s->continuar == 2){
        if(s->posX >= 10){
            s->posX-= s->velocidade;
            if(s->posX%10==0){ s->backupX = s->posX; }
        }else{ s->codigoTela = 4; }
    }
    if(s->continuar == 3){
        if(s->posY >= 30){
            s->posY-= s->velocidade;
            if(s->posY%10==0){ s->backupY = s->posY; }
        }else{ s->codigoTela = 4;}
    }
    if(s->continuar == 4){
        if(s->posY <= 460){
            s->posY+= s->velocidade;
            if(s->posY%10==0){ s->backupY = s->posY; }
        }else{ s->codigoTela = 4; }
    }
}
```

Após verificar que é divisor e que queira mudar de direção, é chamado a função ‘mudarDirecao’, onde muda a direção da cobra e volta a execução normalmente, caso não queira mudar a posição, na função jogabilidade, terá continuidade e fará a cobra andar sozinha de acordo com a velocidade fornecida pelo modo (lento = 1, normal = 2, rápido = 5) e pela direção fornecida anteriormente dentro dessa mesma função.

```
void mudarDirecao(Snaker *s){ // Mudar a direção da cobra
    if((s->direcaoCobra == 1) && (s->continuar != 2 || s->scoreAtual == 0)){
        if(s->posY % 10 == 0){
            s->continuar = 1;
            s->travaTela = 0;
            s->direcaoCobra = 0;
        }
    }else if((s->direcaoCobra == 2)  && (s->continuar != 1 || s->scoreAtual == 0)){
        if(s->posY % 10 == 0){
            s->continuar = 2;
            s->travaTela = 0;
            s->direcaoCobra = 0;
        }
    }else if((s->direcaoCobra == 3) && (s->continuar != 4 || s->scoreAtual == 0)){
        if(s->posX % 10 == 0){
            s->continuar = 3;
            s->travaTela = 0;
            s->direcaoCobra = 0;
        }
    }else if((s->direcaoCobra == 4)  && (s->continuar != 3 || s->scoreAtual == 0)){
        if(s->posX % 10 == 0){
            s->continuar = 4;
            s->travaTela = 0;
            s->direcaoCobra = 0;
        }
    }else{
        s->travaTela = 0;
        s->direcaoCobra = 0;
    }
}
```

Em seguida tem a função ‘segmentoCobra’, onde é chamado quando se come uma fruta e é necessário aumentar um novo segmento a cobra, assim é adicionado na matriz da tela o valor 1 quando a cobra está na posição e 0 quando a cobra não está lá, e possui uma outra matriz para armazenar o valor da posição da cobra atual, onde quando ela saiu da primeira posição, vai mudando a posição de todas para manter o movimento da cabeça com seus segmentos.

```
void segmentoCobra(Snaker *s){ // Segmento da cobra
    int contador = 0;
    if(s->posX%10 == 0 && s->posY%10 == 0){
        s->seq[s->posX][s->posY] = 1;
        s->seqCoords[s->seqCoordsCont][0] = s->posX;
        s->seqCoords[s->seqCoordsCont][1] = s->posY;
        s->seqCoordsCont++;
        for(int i = 10; i <= 620; i++){
            for(int j = 30; j <= 460; j++){
                if(s->seq[i][j] == 1){ contador++; }
            }
        }
        int calculo = s->seqCoordsCont - contador;
        if(contador <= s->scoreAtual+1){ }else{
            s->seq[s->seqCoords[calculo][0]][s->seqCoords[calculo][1]] = 0;
        }
        colisao(s, calculo);
    }
    imagemCobra(s);
}
```

2. Colisão da cobra com os limites da tela, com si própria e com a fruta

Para a colisão da cobra com os limites da tela, foi realizado junto com sua movimentação, como pode ser visto nas fotos anteriores, onde é verificado se a nova posição da cobra passa de alguma das bordas, por exemplo se ultrapassa 630 ou se é menor que 10 na posição x, caso seja verdade irá chamar a variável codTela 4 e dará game over.

Em seguida tem a função ‘colisao’ onde é realizada a verificação se a posição atual da cobra bate em outra parte dela mesma, no caso dos segmentos, caso houver a colisão, chama a tela de game over.

```
void colisao(Snaker *s, int calculo){ // Colisao da cobra com ela mesma
    if(s->seq[s->posX][s->posY] == 1){
        int colisao = 0;
        for(int i = calculo; i <= s->seqCoordsCont; i++){
            if(s->seqCoords[i][0] == s->backupX && s->seqCoords[i][1] == s->backupY){ colisao++; }
        }
        if(colisao > 1 && s->scoreAtual>0){ s->codigoTela = 4; }
    }
}
```

De ultimo tem a colisão com a fruta, para isso é chamado inicialmente uma função para gerar a fruta em uma posição onde não tem segmentos da cobra e que seja dentro da tela limitada e após é chamado a função ‘pegaFruta’ onde é realizado a verificação se a posição da cabeça é a mesma da fruta, se for, aí retornar mais um score e chama a função ‘geradorDeFrutas’ novamente, para assim ter uma próxima para pegar.

```
void geradorDeFrutas(Snaker *s){ // Gera a fruta na imagem
    if(s->gerarFruta == 0){
        s->frutaX = 10 + rand() % 610;
        s->frutaY = 30 + rand() % 430;
        for(int i = 1; i<10; i++){
            if(s->frutaX%10 == i){
                if(i>=6){ s->frutaX+= 10 - i;
                }else{ s->frutaX-= i; }
            }
        }
        for(int i = 1; i<10; i++){
            if(s->frutaY%10 == i){
                if(i>=6){ s->frutaY+= 10 - i;
                }else{ s->frutaY-= i; }
            }
        }
        if(s->seq[s->frutaX][s->frutaY] == 0){ s->gerarFruta = 1; }
    }
    if(s->gerarFruta == 1){ desenha_imagem(s->frutaX+5, s->frutaY+5, "assets/FrutaVerme.png"); }
}
void pegarFruta(Snaker *s){ // Pega a fruta na tela
    if((s->frutaX == s->posX || s->frutaX == s->posX+9) && (s->frutaY == s->posY || s->frutaY == s->posY+9)){
        toca_som("assets/Comendo.mp3");
        s->scoreAtual++;
        s->gerarFruta = 0;
    }else{ s->gerarFruta = 1; }
}
```

3. Informações que o aluno considerar importante

Além das principais funções do jogo, também existe outras desenvolvidas a parte, por exemplo a função pausa durante a movimentação da cobra, onde se aperta a tecla ‘P’ e pausa a movimentação da cobra.

```
void pausarTela(Snaker *s){ // Pausa o jogo em andamento
    if (tecla_acabou_de_ser_apertada('P')){
        if(s->continuar!=0){
            s->continuarBackup = s->continuar;
            s->continuar = 0;
        }else{ s->continuar = s->continuarBackup; }
    }
    if(s->continuar==0){ desenha_imagem(320, 240, "assets/Fundo_Pausado.png"); }
}
```

Também possui uma funcionalidade para trocar a skin da cobra, onde a cobra pode ser em formato de círculo ou quadrado, junto com o código que faz a mudança da imagem da cabeça da cobra, já que possui uma carinha na cabeça que muda de acordo com a direção seguida.

```
void imagemCobra(Snaker *s){ // Carrega imagem da cobra na tela
    if(s->continuar == 1){ desenha_imagem(s->backupX+5, s->backupY+5, s->skin == 0 ? "assets/CobraDIREITA.png" : "assets/CobraDIREITA2.png");
    }else if(s->continuar == 2){ desenha_imagem(s->backupX+5, s->backupY+5, s->skin == 0 ? "assets/CobraESQUERDA.png" : "assets/CobraESQUERDA2.png");
    }else if(s->continuar == 3){ desenha_imagem(s->backupX+5, s->backupY+5, s->skin == 0 ? "assets/CobraCIMA.png" : "assets/CobraCIMA2.png");
    }else if(s->continuar == 4){ desenha_imagem(s->backupX+5, s->backupY+5, s->skin == 0 ? "assets/CobraBAIXO.png" : "assets/CobraBAIXO2.png"); }
    for(int i = 10; i <= 620; i++){
        for(int j = 30; j <= 460; j++){
            if(i == s->backupX && j == s->backupY){
            }else if(s->seq[i][j] == 1){ desenha_imagem(i+5, j+5, s->skin == 0 ? "assets/Cobra.png" : "assets/CobraSEGMENTO2.png"); }
        }
    }
}
```

Além disso existe uma função chamada ‘atualizar’ que tem a responsabilidade de atualizar todas as telas, com os novos dados, encurtando muito o código.

```
void atualizar(Snaker *s){ // Atualiza a tela em tempo real
    desenha_imagem(320, 240, "assets/Fundo_DificuldadesGrade.png");
    fonte("assets/ubuntu.ttf", 20);
    desenha_texto(15, 20, "Score: %d", s->scoreAtual*s->velocidade);
    if(s->velocidade == 1){ desenha_texto(290, 20, "Lento");
    }else if(s->velocidade == 2){ desenha_texto(290, 20, "Normal");
    }else{ desenha_texto(290, 20, "Rapido"); }
    pausarTela(s);
    if(s->continuar!=0){
        geradorDeFrutas(s);
        segmentoCobra(s);
        jogabilidade(s);
        pegarFruta(s);
    }
}
```

Por fim tem as funções de ‘adicionarDados’ e ‘lerDados’, responsável por armazenar o ranking do jogo atual.

```
void adicionarDados(Snaker *s){ // Adiciona dados ao arquivo texto
    lerDados(s);
    FILE *arquivo = fopen("assets/ranking.txt", "w");
    if(s->scoreAtual>s->scoreS[4]){
        s->scoreS[4] = s->scoreAtual;
        s->nomeS[4] = s->nome;
    }
    for(int i = 0; i<5; i++){
        for(int j = 0; j<5; j++){
            if(i != j){
                if(s->scoreS[j] < s->scoreS[i]){
                    int copiaS = s->scoreS[i];
                    s->scoreS[i] = s->scoreS[j];
                    s->scoreS[j] = copiaS;
                    char *copiaN = s->nomeS[i];
                    s->nomeS[i] = s->nomeS[j] ;
                    s->nomeS[j] = copiaN;
                }
            }
        }
    }
    char vetorScore[10];
    for(int i = 0; i<5; i++){
        itoa(s->scoreS[i], vetorScore, 10);
        fputs(s->nomeS[i], arquivo);
        fputc(',', arquivo);
        fputs(vetorScore, arquivo);
        fputc(',', arquivo);
        fputc('\n', arquivo);
    }
    fclose(arquivo);
}
```

```
void lerDados(Snaker *s){ // Le os dados do arquivo texto

    FILE *arquivo = fopen("assets/ranking.txt", "r");
    char buf[125];
    int cont = 0;
    int scoreL;
    char nomeL[15];

    if(arquivo != NULL){
        fgets(buf, 125, arquivo);
        while(!feof(arquivo) && cont<5){
            strcpy(nomeL, strtok(buf,","));
            scoreL = atoi(strtok(NULL,","));
            // printf("Nome: %s Score: %d\n", nomeL, scoreL);
            s->scoreS[cont] = scoreL;
            for(int i = 0; i<15; i ++){
                s->nomeS[cont][i] = nomeL[i];
            }
            fgets(buf, 125, arquivo);
            cont++;
        }
    }
    fclose(arquivo);

    if(s->codigoTela==5){ // Escreve o rank na tela ranking
        cor(BRANCO);
        fonte("assets/ubuntu.ttf", 37);
        desenha_texto(215, 143, "%s - %d", s->nomeS[0], s->scoreS[0]);
        fonte("assets/ubuntu.ttf", 35);
        desenha_texto(225, 200, "%s - %d", s->nomeS[1], s->scoreS[1]);
        fonte("assets/ubuntu.ttf", 30);
        desenha_texto(240, 255, "%s - %d", s->nomeS[2], s->scoreS[2]);
        fonte("assets/ubuntu.ttf", 25);
        desenha_texto(255, 310, "%s - %d", s->nomeS[3], s->scoreS[3]);
        fonte("assets/ubuntu.ttf", 20);
        desenha_texto(265, 360, "%s - %d", s->nomeS[4], s->scoreS[4]);
    }
}
```

4. Imagens do jogo em execução

A seguir temos as 4 imagens principais do jogo