MINI GOLF

CHRINTIAN BARNARD, JOÃO PEDRO MARANHÃO &
JÚLIA SALES

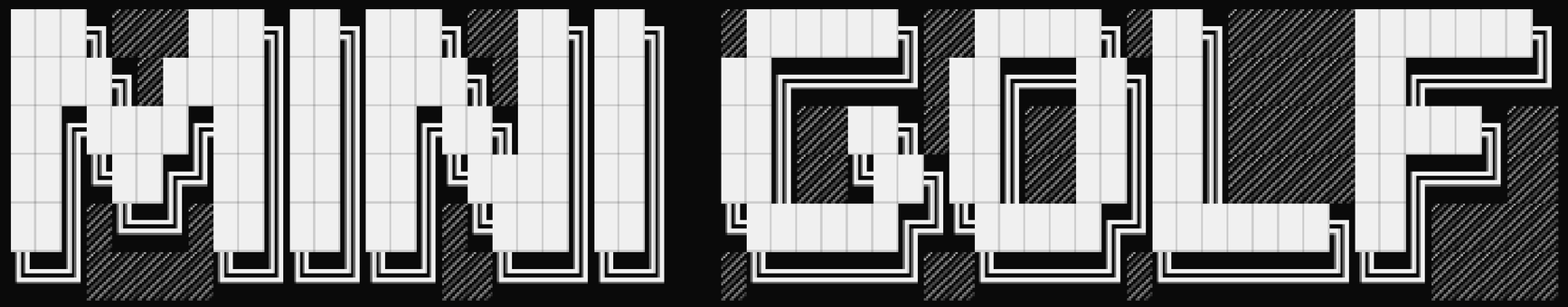
APRENSENTAÇÃO DO JOGO

- ## DESCRIÇÃO

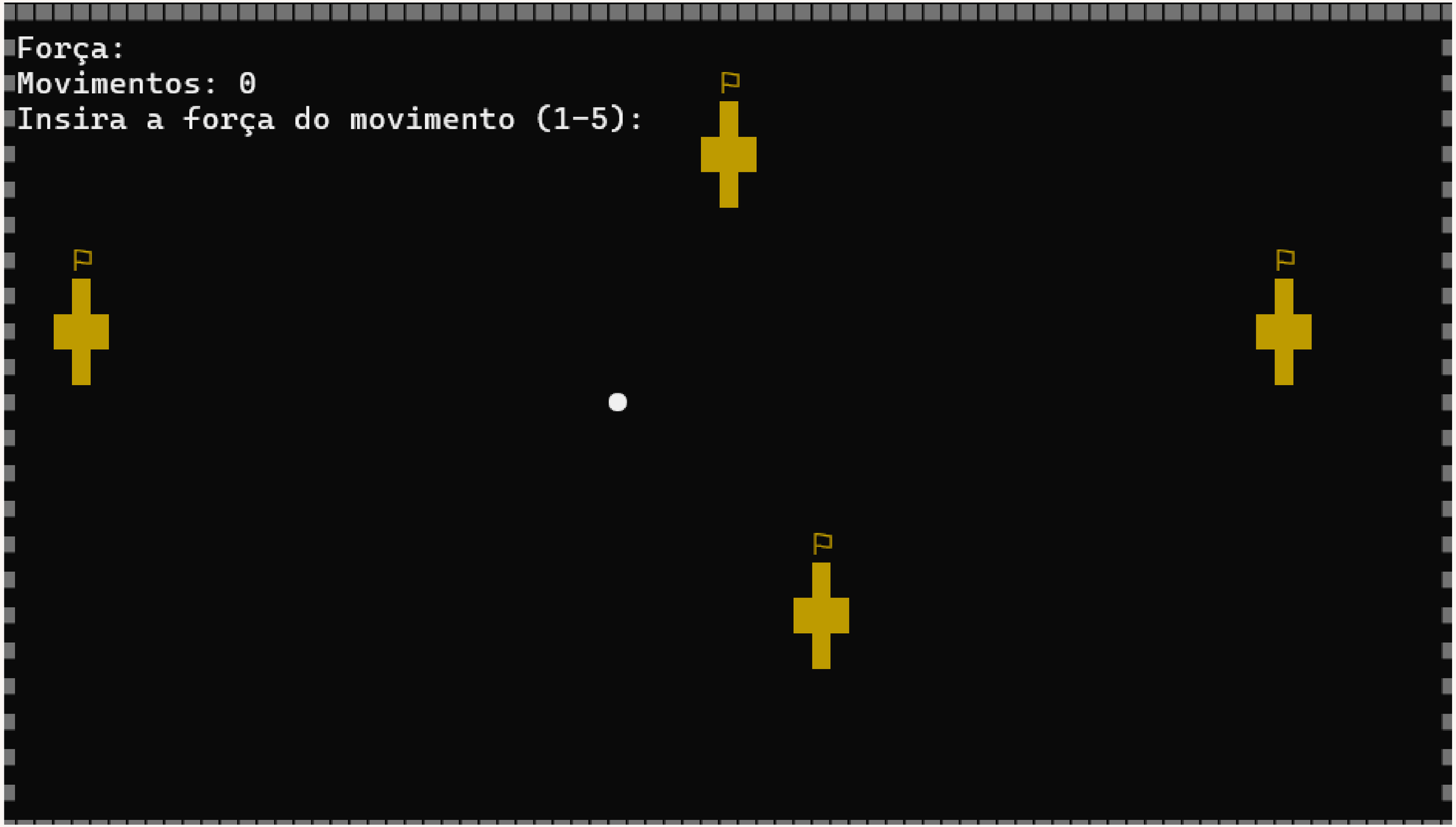
  Mini golf é um jogo baseado no esporte da vida real, onde jogadores competem entre si para acertarem o buraco apenas movimentandoa bola e desviando dos possíveis obstáculos, isso com o menor número de movimentos possíveis

- ## OBJETIVOS DE JOGO

  O objetivo do jogo é fazer com que os dois jogadores joguem entre si e acertarem o buraco com a menor quantidade de movimentos possível. Quem chegar ao objetivo final com menos jogadas ganhará uma maior quantidade de pontos e ganhando o jogo



[SELECT ANY KEY TO CONTINUE]


Força:
Movimentos: 0
Insira a força do movimento (1-5):

APRENSENTAÇÃO DO JOGO

- **PONTUAÇÃO**
  - 1 à 2 tacadas = 10
  - 3 à 4 tacadas = 8
  - 5 à 6 tacadas = 6
  - 7 à 8 tacadas = 3
  - 9 à 10 tacadas = 2
  - 11+ tacadas = 2

- **FORÇA DE LANÇAMENTO**

  O jogador poderá escolher o nível da força entre 1 e 5, representando não apenas números, mas também o crescimento da força das jogadas

APRENSENTAÇÃO DO JOGO

- **COMO JOGAR**
  - Use as teclas “w – A – S – D” para se mover nas quatro direções principais, respectivamente, CIMA, ESQUERDA, BAIXO, DIREITA
  - Use Q – E – Z – C para se mover nas quatro diagonais, respectivamente, CIMA ESQUERDA, CIMA DIREITA, BAIXO ESQUERDA E BAIXO DIREITA.
  - Certifique-se de manter o caps lock desativado
  - Caso a bola acerte a parede ela será ricocheteada e voltará a quantidade de casa de acordo com a força jogada dividido por 2.

EXECUTANDO O JOGO

- **ETAPAS**
  - Clone este repositório em sua máquina: https://github.com/julsales/mini-golf-pif.git
  - Compile o programa: gcc .\main.c .\cli-lib-main\src*.c -I.\cli-lib-main\include\ -o minigolfe
  - Rodou o programa:gcc .\main.c

DETALHES DA IMPLEMENTAÇÃO

- FUNÇÕES DE (KEYBOARD.H USADAS NA MAIN)
  - void keyboardInit();
    - Permite a leitura de teclas imediatamente sem a necessidade de pressionar a tecla "Enter".

DETALHES DA IMPLEMENTAÇÃO

- **FUNÇÕES DE (SCREEN.H USADAS NA MAIN)**
  - void screenInit();
    - É função responsável por inicializa a tela, limpando-a e restaurando-a para o estado inicial.
  - void screenClear();
    - É função responsável por limpar a tela movendo o cursor para a posição inicial.
  - void screenSetColor(screenColor fg, screenColor bg);
    - É a função responsável por definir as cores do texto na tela. fg sendo cor do primeiro plano e bg a cor de fundo.
  - void screenGotoxy(int x, int y);
    - É a função responsável por mover o cursor para uma posição especificada na tela, tendo como coordenadas (x, y).

DETALHES DA IMPLEMENTAÇÃO

- **FUNÇÕES DE (SCREEN.H USADAS NA MAIN)**
  - void screenSetBlink();
    - É a função responsável pro definir o modo de exibição do texto como piscante.
  - void screenSetNormal();
    - É a função responsável por definir o modo de exibição do texto como normal.
  - void screenUpdate();
    - Força a atualização imediata da tela.

DETALHES DA IMPLEMENTAÇÃO

- **FUNÇÕES DE (MAIN.C)**
  - void menuinit();
    - É a função responsável por exibir o menu inicial do jogo na tela.
  - void desenharParedes();
    - É a função responsável por desenhar as paredes do campo do jogo
  - void printbola(int nextX, int nextY);
    - É a função responsável por printar a bola na posição específica
  - void printburaco(struct buraco *bur, int x, int y, int raio);
    - É a função responsável por printar o buraco na tela na posição específicada
  - void apagarburaco(struct buraco *bur);
    - É a função responsável por apagar o buraco na posição especificada
  - void moverbola(struct player *ptp ,char direcao);
    - É a função responsável por mover a bola em uma direção e com uma força especificada

DETALHES DA IMPLEMENTAÇÃO

- **FUNÇÕES DE (MAIN.C)**
  - `void animacaobola(int initX, int initY, int endX, int endY);`
    - É a função responsável por fazer a animação da bola de uma posição inicial até uma posição final
  - `int calcularForca(int mov);`
    - É a função responsável por calcular a força de acordo com o movimento escolhido
  - `int colisao(struct buraco *bur, int Ox, int Oy);`
    - É a função responsável por verificar se a bola colide com o buraco.

HIGHLIGHTS DO CÓDIGO

- ENQUANTO (*REMAININGFORCE > 0), A BOLA CONTINUA A MOVER-SE MESMO QUANDO COLIDE COM UMA PAREDE, DESDE QUE A FORÇA SEJA SUFICIENTE PARA SUPERÁ-LA, RETORNANDO O VALOR EXCEDENTE.
- A PRIMEIRA PARTE MOVE A BOLA DA SUA POSIÇÃO ATUAL PARA A PRÓXIMA. OS DOIS PRIMEIROS IF SERVEM PARA ATUALIZAR A PRÓXIMA POSIÇÃO DA BOLA COM BASE NAS DIREÇÕES X E Y.
- OS DOIS PRÓXIMOS IF VERIFICAM A COLISÃO COM AS PAREDES E RETORNAM À POSIÇÃO ANTERIOR USANDO CURRENTX, QUE RECALCULA A POSIÇÃO DA BOLA COM BASE NA EXPRESSÃO E NA LARGURA, REDUZINDO PELA METADE A FORÇA AO DEVOLVÊ-LA, UTILIZANDO REMAININGFORCE, QUE FOI DECLARADO ANTERIORMENTE.

- TUDO ISSO COMPÕE A ANIMAÇÃO DA BOLA.

```c
void animacaobola(int initX, int initY, int endX, int endY, int *forcarebote) {
    int atualX = initX;
    int atualY = initY;

    while ((atualX != endX || atualY != endY) && *forcarebote > 0) {
        screenSetColor(fg:WHITE, bg:DARKGRAY);
        screenGotoxy(atualX, atualY);
        printf(" ");

        if (atualX < endX) {
            atualX++;
        } else if (atualX > endX) {
            atualX--;
        }
        if (atualY < endY) {
            atualY++;
        } else if (atualY > endY) {
            atualY--;
        }
        if (atualX <= ESPESSURA) {
            atualX = ESPESSURA;
            endX = LARGURA - ESPESSURA - 1;
        } else if (atualX >= LARGURA - ESPESSURA - 1) {
            atualX = LARGURA - ESPESSURA - 1;
            endX = ESPESSURA;
        }
        if (atualY <= ESPESSURA) {
            atualY = ESPESSURA;
            endY = ALTURA - ESPESSURA - 1;
        } else if (atualY >= ALTURA - ESPESSURA - 1) {
            atualY = ALTURA - ESPESSURA - 1;
            endY = ESPESSURA;
        }

        screenGotoxy(atualX, atualY);
        printf("●");
        screenUpdate();
        usleep(useconds:50000);
        (*forcarebote)--;
    }

    x = atualX;
    y = atualY;
}
```

HIGHLIGHTS DO CÓDIGO

- INT CALCULARFORCA(INT MOV);

- DEFAULT: RETURN 1;(CASO NÃO SEJA NENHUM DOS VALORES A CIMA, VAI RETORNAR O VALOR DE FORÇA 1)

```
int calcularForca(int mov) {
    switch (mov) {
        case 1:
            return 1;
        case 2:
            return 3;
        case 3:
            return 5;
        case 4:
            return 7;
        case 5:
            return 9;
        default:
            return 1;
    }
}
```

HIGHLIGHTS DO CÓDIGO

- *BUR É PONTEIRO QUE REPRESENTA O BURACO, COM AS CORDENADAS SEQUENCIAS DO X E Y, QUE FICA NO CENTRO DO MAPA (0 0).
- NO GERAL ELA APONTA AS COLISÕES DA BOLA RECALCULANDO A CADA MOVIMENTO ELA EM RELAÇÃO AOS EIXOS X E Y, COM O DY =BUR ->X (Y), CALCULANDO O RAIO E RETORNANDO 0 CASO TENHA COLISÃO E 1 CASO NÃO TENHA COLISÃO.

- VOID PRINTBOLA(INT NEXTX, INT NEXTY);

```c
int colisao(struct buraco *bur, int 0x, int 0y) {
    int dx = bur->x - 0x;
    int dy = bur->y - 0y;
    int distancia = dx * dx + dy * dy;
    int raioQuadrado = bur->raio * bur->raio;
    return distancia <= raioQuadrado;
}
```

```c
void printbola(int nextX, int nextY) {
    screenSetColor(WHITE, DARKGRAY);
    screenGotoxy(x, y);
    printf(" ");
    x = nextX;
    y = nextY;
    screenGotoxy(x, y);
    printf("●");
}
```

HIGHLIGHTS DO CÓDIGO

- VOID PRINTBURACO(STRUCT BURACO *BUR, INT X, INT Y, INT RAIO);

- BUR ARMAZENA AS COOREDENADAS DO X/Y/RAIO NA STRUCT BUR.
- O FOR SÃO DOIS LOOPS PARA FORMAR O BURACO APRESENTANDO TRAÇOS PEQUENOS E GRANDES PARA SIMBOLIZAR O MEIO.
- E O SCREENGOTOXY APENAS REPRESENTA A POSIÇÃO DETALHADA TANTO DO BURACO COMO DA BANDEIRA QUE SIMBOLOZA A LOCALIZAÇÃO DO BURACO

```
void printburaco(struct buraco *bur, int x, int y, int raio) {
    screenSetColor(fg:BROWN, bg:BROWN);
    bur->x = x;
    bur->y = y;
    bur->raio = raio;
    int i, j;
    for (i = y - raio; i <= y + raio; i++) {
        for (j = x - raio; j <= x + raio; j++) {
            if ((x - j) * (x - j) + (y - i) * (y - i) <= raio * raio) {
                screenGotoxy(x:j, y:i);
                printf(" ");
            }
        }
    }
    screenSetColor(fg:BROWN, bg:DARKGRAY);
    screenGotoxy(x:j - 2, y:i - 4);
    printf("P");
    screenSetColor(fg:WHITE, bg:DARKGRAY);
}
```

HIGHLIGHTS DO CÓDIGO

- VOID APAGARBURACO(STRUCT BURACO *BUR);

- APAGA O BURACO LOGO APÓS A BOLA ACERTAR O CENTRO DO BURACO, FAZENDO COM QUE O MESMO DESAPAREÇA APÓS TER SIDO ‘CHOCADA’ COM A BOLA, FEITO INTERLIGADO COM A COLISÃO

```c
void apagarburaco(struct buraco *bur) {
    int x = bur->x;
    int y = bur->y;
    int raio = bur->raio;
    int i, j;
    for (i = y - raio; i <= y + raio; i++) {
        for (j = x - raio; j <= x + raio; j++) {
            if ((x - j) * (x - j) + (y - i) * (y - i) <= raio * raio) {
                screenGotoxy(x:j, y:i);
                printf(" ");
            }
        }
    }
    screenGotoxy(x:j - 2, y:i - 4);
    printf(" ");
    screenSetColor(fg:WHITE, bg:DARKGRAY);
}
```

HIGHLIGHTS DO CÓDIGO

- MOVERBOLA
- SCREENGOTOXY PRINTA A FORÇA E MOVIMENTOS, USANDO PONTEIROS COM PTP –>MOVIMENTOS.

- EM SEGUIDA ATÉ O ULTIMO IF ELE APENAS CONFERE SE ESTÁ DENTRO DO NUMERO DE MOVIMENTO DE 1 A 5 E RECALCULA A FORÇA EM OUTRA STRUCT.

- PTP–>SCORE CALCULA A FORÇA DO MOVIMENTO QUE FOI DADO A CIMA PELO USUÁRIO E RETORNA EM VALOR DE FORÇA, EX 2 = 4, MUDANDO A QUANTIDADE DE MOVIMENTOS E ATUALIZANDO A INFORMAÇÃO NO CONTADOR.

- POR FIM ELE APENAS CALCULA A DIREÇÃO COM OS 'CASE'

```c
void moverbola(struct node *ptp, char direcao) {
    screenGotoxy(x:2, y:2);
    printf("Força:");
    screenGotoxy(x:2, y:3);
    printf("Movimentos: %d", ptp->movimentos);

    int nextX = x;
    int nextY = y;
    int mov;
    screenGotoxy(x:2, y:4);
    printf("Insira a força do movimento (1-5): ");
    if (scanf("%d", &mov) != 1) {
        return;
    }
    if (mov < 1) {
        mov = 1;
    }
    if (mov > 5) {
        mov = 5;
    }
    ptp->score = calcularPontuacao(ptp->movimentos)
    int score = ptp->score;
    screenGotoxy(x:2, y:10);
    printf("Pontuação: %d", score);
    int forca = calcularForca(mov);
    screenGotoxy(x:2, y:2);
    printf("Força: %d", forca);
    ptp->movimentos++;
    screenGotoxy(x:2, y:3);
    printf("Movimentos: %d", ptp->movimentos);
```

```c
    switch (direcao) {
        case 'w':
            nextY = y - forca;
            break;
        case 'a':
            nextX = x - forca;
            break;
        case 's':
            nextY = y + forca;
            break;
        case 'd':
            nextX = x + forca;
            break;
        case 'q':
            nextX = x - forca;
            nextY = y - forca;
            break;
        case 'e':
            nextX = x + forca;
            nextY = y - forca;
            break;
        case 'z':
            nextX = x - forca;
            nextY = y + forca;
            break;
        case 'c':
            nextX = x + forca;
            nextY = y + forca;
            break;
    }
    int forcarebote = forca;
    animacaobola(initX:x, initY:y, endX:nextX, endY:nextY, &forcarebote);
}
```

HIGHLIGHTS DO CÓDIGO

- VOID DESENHARPAREDES();

- DESENHAR PAREDE É UMA FUNÇÃO ENCADEADA PROXIMA A UMA MATRIZ, ONDE SE DECLARA INICIALMENTE A EXPRESSURA DA PAREDE (QUANTIDADE DE BLOCOS)
- LARGURA DA PAREDE (ATÉ QUANTO ELA VAI DA ESQUERDA PARA DIREITA)
- ALTURA DA PAREDE (DE CIMA A BAIXO), REPETINDO A LARGURA TAMBÉM DA PAREDE PAREDE.

```c
void desenharParedes() {
    int i, j;
    screenSetColor(fg:DARKGRAY, bg:BLACK);
    for (i = 0; i < ESPESSURA; i++) {
        for (j = 1; j < LARGURA; j++) {
            screenGotoxy(x:j, y:i);
            printf("■");
            screenGotoxy(x:j, y:ALTURA - i);
            printf("■");
        }for (j = 1; j < ALTURA; j++){
            screenGotoxy(x:i, y:j);
            printf("■");
            screenGotoxy(x:LARGURA - i, y:j);
            printf("■");
        }
    }
}
```

HIGHLIGHTS DO CÓDIGO

- VOID MENUINIT();

- O INT X1/Y1 DECLARA A POSIÇÃO PRINCIPAL DO CENTRO ONDE FICA A ESCRITURA, CADA UM DOS SCREENGOTOXY APENAS REFAZ A POSIÇÃO DOS PRINTS CONSECULTIVOS.

```c
void menuinit() {
    int x1 = 10;
    int y1 = 8;
    screenGotoxy(x1, y1);
    screenSetColor(fg:WHITE, bg:DARKGRAY);
    printf("▓▓▓╗▒▒▓▓▓╗▓▓▓╗▒▒▓▓▓╗▓▓▓╗EMSPEMSP▓▓▓▓▓▓╗▒▒▓▓▓▓▓╗▓▓▓╗▒▒▒▒▓▓▓▓▓▓▓▓╗");
    screenGotoxy(x1, y1 + 1);
    printf("▓▓▓▓╗▒▓▓▓▓╗▓▓╗▓▓▓▓╗▒▓▓╗▓▓╗EMSPEMSP▓▓╔═══╝▓▓╔═▓▓╗▓▓╗▒▒▒▒▓▓╔═══╝");
    screenGotoxy(x1, y1 + 2);
    printf("▓▓╔▓▓▓▓╔▓▓╗▓▓╗▓▓╔▓▓╗▓▓╗▓▓╗EMSPEMSP▓▓▓▓▓▓▓▓╗▓▓▓▓▓▓▓╗▓▓╗▒▒▒▒▓▓▓▓▓▓▓▓╗");
    screenGotoxy(x1, y1 + 3);
    printf("▓▓╚═╝▓▓╔═╝▓▓╗▓▓╗▓▓╚═══╝▓▓╗EMSPEMSP▓▓╔═══╝▓▓╔═▓▓╗▓▓╗▒▒▒▒▓▓╔═══╝");
    screenGotoxy(x1, y1 + 4);
    printf("▓▓╗▒▒▒▓▓╗▒▒▓▓╗▓▓╗▒▒▓▓▓▓▓▓╔╝EMSPEMSP▓▓▓▓▓▓▓▓╔╝▓▓▓▓▓▓▓▓╔╝▓▓╗▒▒▒▒");
    screenGotoxy(x1, y1 + 5);
    printf("╚═╝▒▒▒▒▒▒╚═╝╚═╝╚═╝▒▒▒▒▒▒╚═╝EMSPEMSP▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒");
    screenSetColor(fg:YELLOW, bg:DARKGRAY);
    screenGotoxy(x:25, y:22);
    screenSetBlink();
    printf("[SELECT ANY KEY TO CONTINUE]");
    screenGotoxy(x:26, y:23);
    printf("                                ");
    screenSetNormal();
}
```

DESAFIOS DO CÓDIGO

- OS MAIORES DESAFIOS DE FAZER O CÓDIGO FORAM COM CERTEZA NA MOVIMENTAÇÃO E O RICOCHETE DA BOLA, POIS É NECESSÁRIA VÁRIAS INTERAÇÕES E FORMULAS PARA FAZER O MESMO
- ALÉM DISSO, O SITEMA DE PONTUAÇÕES FOI UM POUCO CONFUSO DE FAZER POIS É DEPENDENTE DA QUANTIDADE DE TACADAS
- TAMBÉM HOUVE UMA TENTATIVA DE IMPLEMETAÇÃO DE OBSTACULOS, QUE ACABOU NÃO SENDO REALIZADA
- OUTRA DIFICULDADE FOI CORRIGIR O PROBLEMA DA BOLA APAGAR AS BORDAS DO MAPA, MESMO COLOCANDO PARES INVISIVEIS,A BOLA ULTRAPASSAVA E APAGAVA A BORDA.