Lista 2

1. As etapas envolvidas no processo de preenchimento de polígonos são: primeiramente deve-se calcular as bordas de todo polígono, pois elas indicaram quais pontos devem ser pintados ou não. Depois, deve-se percorrer todos os pontos da grade onde é mostrado o objeto, começando pelo ponto mais acima a esquerda do objeto, e então verificar se o ponto está dentro do objeto, se estiver o ponto é pintado, caso contrário não.

/\*considerar para os exercícios seguintes que a janela de visulização possui 400 pontos discretos (20x20)\*/

void preencherPoligono(){

int cor, i, j, flag = 0, numP = 4, inicioX = 20, fimX = 0, inicioY = 20, fimY = 0;

ponto \*vetPontos;

/\*retorna um vetor com os pontos base do retângulo \*/

vetPontos = desenharPoligono();

/\*verifica os pontos extremos do retângulo\*/

for(i = 0; i < numP; i++){

if(vetPontos[i].x < inicioX)

inicioX = vetPontos[i].x;

if(vetPontos[i].x > fimX)

fimX = vetPontos[i].x;

if(vetPontos[i].y < inicioY)

inicioY = vetPontos[i].y;

if(vetPontos[i].y > fimY)

fimY = vetPontos[i].y;

}

printf("Informe uma cor(numero != 0)");

scanf("%d",&cor);

/\*preenche o retângulo\*/

for(i = inicioX; i < fimX; i++){

flag = 0;

for(j = inicioY; j <= fimY; j++){

if(matriz[i][j] == 1)

flag = !flag;

else{

if(flag)

matriz[i][j] = cor;

}

printf("\ni: %d j: %d ponto = %d flag = %d",i,j,matriz[i][j], flag);

}

}

/\*mostra o retângulo\*/

for(i = 0; i < 20; i++){

printf("\n");

for(j = 0; j < 20; j++){

printf(" %d", matriz[j][19-i]);

}

}

}

void preencherPoligono(){

int cor, i, j, flag = 0, numP = 4, inicioX = 10, fimX = -10, inicioY = 10, fimY = -10;

ponto \*vetPontos;

vetPontos = desenharPoligono();

/\*mostra a matriz\*/

for(i = 0; i < 20; i++){

printf("\n");

for(j = 0; j < 20; j++){

printf(" %d", matriz[j][19 - i]);

}

}

for(i = 0; i < numP; i++){

if(vetPontos[i].x < inicioX)

inicioX = vetPontos[i].x;

if(vetPontos[i].x > fimX)

fimX = vetPontos[i].x;

if(vetPontos[i].y < inicioY)

inicioY = vetPontos[i].y;

if(vetPontos[i].y > fimY)

fimY = vetPontos[i].y;

}

printf("inicioX: %d",fimX);

printf("Informe uma cor(numero != 0)");

scanf("%d",&cor);

for(i = inicioX; i < fimX; i++){

flag = 0;

for(j = inicioY; j <= fimY; j++){

if(matriz[i+10][j+10] == 1)

flag = !flag;

else{

if(flag)

matriz[i+10][j+10] = cor;

}

printf("\ni: %d j: %d ponto = %d flag = %d",i,j,matriz[i][j], flag);

}

}

/\*mostra a matriz\*/

for(i = 0; i < 20; i++){

printf("\n");

for(j = 0; j < 20; j++){

printf(" %d", matriz[j][19-i]);

}

}

}

1. Os pixels que pertencem as retas das arestas, não foram preenchidos.
2. Nos exercícios 2 e 3 já foi realizada esta operação.

void preencherPoligono(){

int cor, i, j, flag = 0, numP = 4, inicioX = 10, fimX = -10, inicioY = 10, fimY = -10;

ponto \*vetPontos;

vetPontos = desenharPoligono();

/\*mostra a matriz\*/

for(i = 0; i < 20; i++){

printf("\n");

for(j = 0; j < 20; j++){

printf(" %d", matriz[j][19 - i]);

}

}

for(i = 0; i < numP; i++){

if(vetPontos[i].x < inicioX)

inicioX = vetPontos[i].x;

if(vetPontos[i].x > fimX)

fimX = vetPontos[i].x;

if(vetPontos[i].y < inicioY)

inicioY = vetPontos[i].y;

if(vetPontos[i].y > fimY)

fimY = vetPontos[i].y;

}

printf("inicioX: %d",fimX);

printf("Informe uma cor(numero != 0)");

scanf("%d",&cor);

for(i = inicioX; i <= fimX; i++){

flag = 0;

for(j = inicioY; j <= fimY; j++){

if(matriz[i+10][j+10] == 1){

flag = !flag;

matriz[i+10][j+10] = 4;

}

else{

if(flag)

matriz[i+10][j+10] = cor;

}

printf("\ni: %d j: %d ponto = %d flag = %d",i,j,matriz[i][j], flag);

}

}

/\*mostra a matriz\*/

for(i = 0; i < 20; i++){

printf("\n");

for(j = 0; j < 20; j++){

printf(" %d", matriz[j][19-i]);

}

}

}

void preencherPoligono(){

int cor, i, j, flag = 0, numP = 4, inicioX = 10, fimX = -10, inicioY = 10, fimY = -10;

ponto \*vetPontos;

vetPontos = desenharPoligono();

/\*mostra a matriz\*/

for(i = 0; i < 20; i++){

printf("\n");

for(j = 0; j < 20; j++){

printf(" %d", matriz[j][19 - i]);

}

}

for(i = 0; i < numP; i++){

if(vetPontos[i].x < inicioX)

inicioX = vetPontos[i].x;

if(vetPontos[i].x > fimX)

fimX = vetPontos[i].x;

if(vetPontos[i].y < inicioY)

inicioY = vetPontos[i].y;

if(vetPontos[i].y > fimY)

fimY = vetPontos[i].y;

}

printf("inicioX: %d",fimX);

printf("Informe uma cor(numero != 0)");

scanf("%d",&cor);

for(i = inicioX; i <= fimX; i++){

flag = 0;

for(j = inicioY; j <= fimY; j++){

if(matriz[i+10][j+10] == 1){

flag = !flag;

if(i != fimX && j != fimY)

matriz[i+10][j+10] = 4;

}

else{

if(flag)

matriz[i+10][j+10] = cor;

}

printf("\ni: %d j: %d ponto = %d flag = %d",i,j,matriz[i][j], flag);

}

}

/\*mostra a matriz\*/

for(i = 0; i < 20; i++){

printf("\n");

for(j = 0; j < 20; j++){

printf(" %d", matriz[j][19-i]);

}

}

}

1. A
2. A
3. Regra da paridade: A paridade é indicada por um bit, que pode ser 0(par) ou 1(ímpar). Iniciando com paridade par, a cada intersecção encontrada o bit de paridade é invertido. Neste caso então os próximos bits vão sendo preenchidos(coloridos) enquanto a paridade for ímpar. Por este método podemos escanear cada linha que possui pixels do objeto em questão e verificar quais pixels se encontram dentro do polígono. Esta abordagem foi utilizada nos algoritmos acima, pois é simples e eficiente.