**OTHELLO GAME IN C**

**Abstract:** Othello is a strategy board game for two players, played on an 8×8 uncheckered board. There are sixty-four identical game pieces called *disks* (often spelled "discs"), which are light on one side and dark on the other. Players take turns placing disks on the board with their assigned color facing up. During a play, any disks of the opponent's color that are in a straight line and bounded by the disk just placed and another disk of the current player's color are turned over to the current player's color.

The object of the game is to have the majority of disks turned to display your color when the last playable empty square is filled.

**Code:**

#include<stdio.h>

#include<graphics.h>

#include<string.h>

//number of blocks in a row/column

#define N 8

//top left vertex X

#define X0 100

//top left vertex Y

#define Y0 50

//width of block

#define W 50

//height of block

#define H 50

//radius of circle

#define R 20

//delay between coin flips

#define coinDelay 100

int grid[N][N];

void wait\_for\_char()

{

//Wait for a key press

int in = 0;

while (in == 0) {

in = getchar();

}

}

void flood(int x, int y, int new\_col, int old\_col)

{

// check current pixel is old\_color or not

if (getpixel(x, y) == old\_col) {

putpixel(x, y, new\_col);

flood(x + 1, y, new\_col, old\_col);

flood(x - 1, y, new\_col, old\_col);

flood(x, y + 1, new\_col, old\_col);

flood(x, y - 1, new\_col, old\_col);

}

}

void boundFill(int x, int y, int boundary\_col, int new\_col)

{

if (getpixel(x, y) != boundary\_col && getpixel(x, y) != new\_col) {

putpixel(x, y, new\_col);

boundFill(x + 1, y, boundary\_col, new\_col);

boundFill(x - 1, y, boundary\_col, new\_col);

boundFill(x, y + 1, boundary\_col, new\_col);

boundFill(x, y - 1, boundary\_col, new\_col);

}

}

void placeOneAtIndex(int colour, int x, int y)

{

grid[x][y]=colour;

setcolor(BLUE);

int xc=X0+(x\*W)+W/2,yc=Y0+(y\*H)+H/2;

circle(xc,yc,R);

boundFill(xc,yc,BLUE,colour);

}

void placeOneAtSquare(int colour, char xin, int yin)

{

int y=yin-1;

int x=(int)(xin)-(int)('a');

placeOneAtIndex(colour,x,y);

}

int getOppositeColour(int colour)

{

if(colour==WHITE)

{

return BLACK;

}

else if(colour==BLACK)

{

return WHITE;

}

}

int checkIncentives(int colour, int x, int y)

{

int oppositeColour=getOppositeColour(colour);

int count=0, total\_count=0,i,j;

if(x<N-1)

{

count=0;

i=x+1,j=y;

while(grid[i][j]==oppositeColour)

{

count++;

i++;

}

if(i<N && grid[i][j]==colour)

{

total\_count+=count;

}

}

if(x>0)

{

count=0;

i=x-1,j=y;

while(grid[i][j]==oppositeColour)

{

count++;

i--;

}

if(i>=0 && grid[i][j]==colour)

{

total\_count+=count;

}

}

if(y>0)

{

count=0;

i=x,j=y-1;

while(grid[i][j]==oppositeColour)

{

count++;

j--;

}

if(j>=0 && grid[i][j]==colour)

{

total\_count+=count;

}

}

if(y<N-1)

{

count=0;

i=x,j=y+1;

while(grid[i][j]==oppositeColour)

{

count++;

j++;

}

if(j<N && grid[i][j]==colour)

{

total\_count+=count;

}

}

if(x>0 && y>0)

{

count=0;

i=x-1,j=y-1;

while(grid[i][j]==oppositeColour)

{

count++;

i--;

j--;

}

if(j>=0 && i>=0 && grid[i][j]==colour)

{

total\_count+=count;

}

}

if(x>0 && y<N-1)

{

count=0;

i=x-1,j=y+1;

while(grid[i][j]==oppositeColour)

{

count++;

i--;

j++;

}

if(i>=0 && j<N && grid[i][j]==colour)

{

total\_count+=count;

}

}

if(x<N-1 && y<N-1)

{

count=0;

i=x+1,j=y+1;

while(grid[i][j]==oppositeColour)

{

count++;

i++;

j++;

}

if(i<N && j<N && grid[i][j]==colour)

{

total\_count+=count;

}

}

if(x<N-1 && y>0)

{

count=0;

i=x+1,j=y-1;

while(grid[i][j]==oppositeColour)

{

count++;

i++;

j--;

}

if(i<N && j>=0 && grid[i][j]==colour)

{

total\_count+=count;

}

}

return total\_count;

}

void flipCoins(int colour, int x, int y)

{

int oppositeColour=getOppositeColour(colour);

int i,j;

if(x<N-1)

{

i=x+1,j=y;

while(grid[i][j]==oppositeColour)

{

i++;

}

if(i<N && grid[i][j]==colour)

{

i=x+1;

j=y;

while(grid[i][j]==oppositeColour)

{

placeOneAtIndex(colour,i,j);

delay(coinDelay);

i++;

}

}

}

if(x>0)

{

i=x-1,j=y;

while(grid[i][j]==oppositeColour)

{

i--;

}

if(i>=0 && grid[i][j]==colour)

{

i=x-1;

j=y;

while(grid[i][j]==oppositeColour)

{

placeOneAtIndex(colour,i,j);

delay(coinDelay);

i--;

}

}

}

if(y>0)

{

i=x,j=y-1;

while(grid[i][j]==oppositeColour)

{

j--;

}

if(j>=0 && grid[i][j]==colour)

{

i=x,j=y-1;

while(grid[i][j]==oppositeColour)

{

placeOneAtIndex(colour,i,j);

delay(coinDelay);

j--;

}

}

}

if(y<N-1)

{

i=x,j=y+1;

while(grid[i][j]==oppositeColour)

{

j++;

}

if(j<N && grid[i][j]==colour)

{

i=x,j=y+1;

while(grid[i][j]==oppositeColour)

{

placeOneAtIndex(colour,i,j);

delay(coinDelay);

j++;

}

}

}

if(x>0 && y>0)

{

i=x-1,j=y-1;

while(grid[i][j]==oppositeColour)

{

i--;

j--;

}

if(i>=0 && j>=0 && grid[i][j]==colour)

{

i=x-1,j=y-1;

while(grid[i][j]==oppositeColour)

{

placeOneAtIndex(colour,i,j);

delay(coinDelay);

i--;

j--;

}

}

}

if(x>0 && y<N-1)

{

i=x-1,j=y+1;

while(grid[i][j]==oppositeColour)

{

i--;

j++;

}

if(i>=0 && j<N && grid[i][j]==colour)

{

i=x-1,j=y+1;

while(grid[i][j]==oppositeColour)

{

placeOneAtIndex(colour,i,j);

delay(coinDelay);

i--;

j++;

}

}

}

if(x<N-1 && y<N-1)

{

i=x+1,j=y+1;

while(grid[i][j]==oppositeColour)

{

i++;

j++;

}

if(i<N && j<N && grid[i][j]==colour)

{

i=x+1,j=y+1;

while(grid[i][j]==oppositeColour)

{

placeOneAtIndex(colour,i,j);

delay(coinDelay);

i++;

j++;

}

}

}

if(x<N-1 && y>0)

{

i=x+1,j=y-1;

while(grid[i][j]==oppositeColour)

{

i++;

j--;

}

if(i<N && j>=0 && grid[i][j]==colour)

{

i=x+1,j=y-1;

while(grid[i][j]==oppositeColour)

{

placeOneAtIndex(colour,i,j);

delay(coinDelay);

i++;

j--;

}

}

}

}

int checkGrid(int colour, int x, int y)

{

if(grid[x][y]!=WHITE && grid[x][y]!=BLACK && checkIncentives(colour,x,y))

{

return 1;

}

return 0;

}

int oneTurn(int colour)

{

char xin;

int yin,x,y;

if(colour==BLACK)

printf("black: ");

if(colour==WHITE)

printf("white: ");

scanf("%c%d",&xin,&yin);

x=(int)(xin)-(int)('a');

y=yin-1;

if(x>=0 && x<N && y>=0 && y<N)

{

if(checkGrid(colour,x,y))

{

placeOneAtIndex(colour,x,y);

flipCoins(colour,x,y);

return 1;

}

}

printf("(invalid)\n");

return 0;

}

void setupBoard()

{

int i;

setbkcolor(BLACK);

line(X0,Y0+(N\*H),X0+(N\*W),Y0+(N\*H));

line(X0+(N\*W),Y0,X0+(N\*W),Y0+(N\*H));

line(X0,Y0,X0+(N\*W),Y0);

line(X0,Y0,X0,Y0+(N\*H));

flood(X0+(N\*W)/2,Y0+(N\*H)/2,GREEN,BLACK);

for(i=1; i<N; i++)

{

line(X0,Y0+(i\*H),X0+(N\*W),Y0+(i\*H));

line(X0+(i\*W),Y0,X0+(i\*W),Y0+(N\*H));

char index[2],alpha[2];

sprintf(index, "%d", i);

sprintf(alpha, "%c", (int)('a')+i-1);

outtextxy(X0-15,Y0+(i\*H)-H/2,index);

outtextxy(X0+(i\*W)-W/2,Y0-15,alpha);

delay(500);

}

char index[2],alpha[2];

sprintf(index, "%d", N);

sprintf(alpha, "%c", (int)('a')+N-1);

outtextxy(X0-15,Y0+(N\*H)-H/2,index);

outtextxy(X0+(N\*W)-W/2,Y0-15,alpha);

}

void main()

{

int gd = DETECT, gm;

int i=0,j=0;

for(i=0;i<N;i++)

{

for(j=0;j<N;j++)

{

grid[i][j]=1;

}

}

initgraph(&gd,&gm,"");

setupBoard();

placeOneAtIndex(WHITE,N/2-1,N/2-1);

delay(250);

placeOneAtIndex(WHITE,N/2,N/2);

delay(250);

placeOneAtIndex(BLACK,N/2,N/2-1);

delay(250);

placeOneAtIndex(BLACK,N/2-1,N/2);

int count=0, success=0;

while(count<N\*N-4)

{

if(count%2)

{

success=oneTurn(BLACK);

}

else

{

success=oneTurn(WHITE);

}

if(success)

{

count++;

}

}

//display winner

int bcnt=0, wcnt=0;

for(i=0;i<N;i++)

{

for(j=0;j<N;j++)

{

if(grid[i][j]==BLACK)

{

bcnt++;

}

else if(grid[i][j]==WHITE)

{

wcnt++;

}

}

}

if(bcnt>wcnt)

{

outtextxy(X0+N/4\*W, Y0/2-10, "Black Wins!");

}

else if(wcnt>bcnt)

{

outtextxy(X0+N/4\*W,Y0/2-10, "White Wins!");

}

else

{

outtextxy(X0+N/4\*W,Y0/2-10, "It is a Tie!");

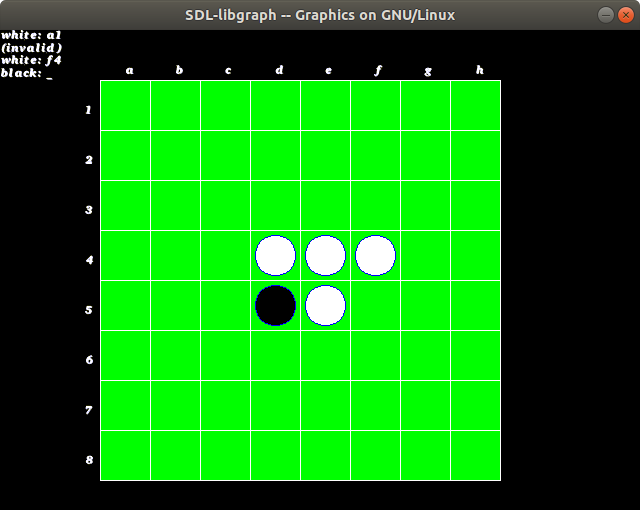
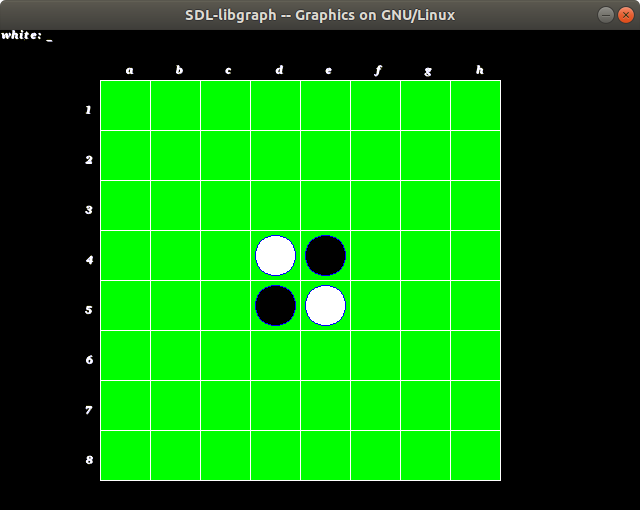
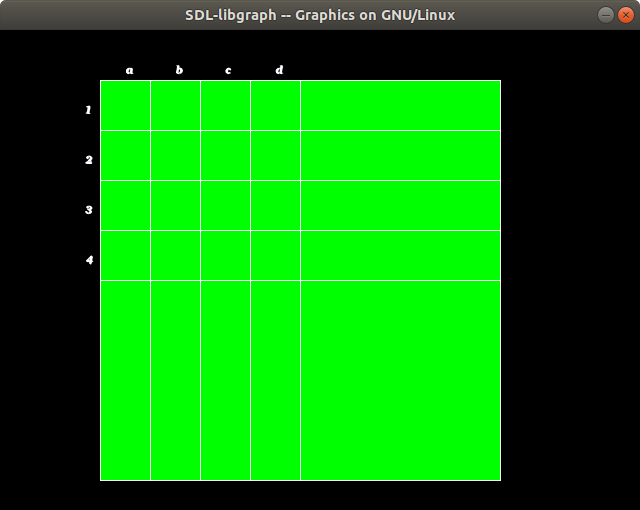
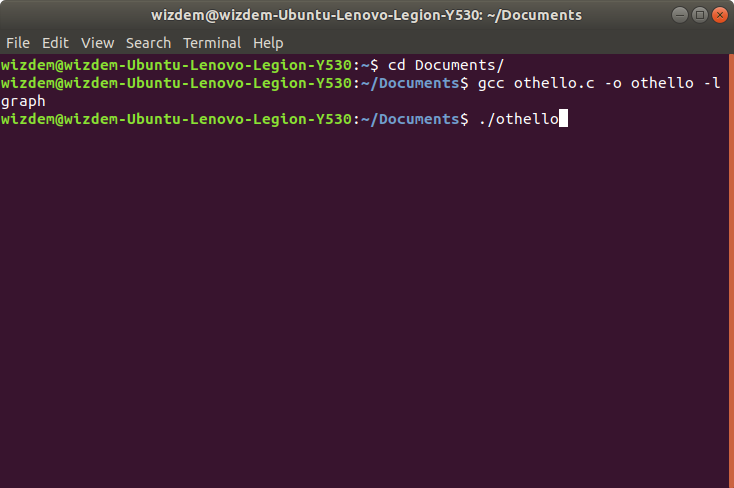
}

wait\_for\_char();

closegraph();

}

**Output:**

****

**Outcome of the Project:**

The project uses basic graphics library in C to create a super lightweight yet immersive game to provide entertainment. It is small enough to be ported to the internet as an in browser game or to play when there’s no internet available.

**Project By:**

Ananya Malik SAP ID: 60004170009

Ankit Gupta SAP ID: 60004170012

Arnav Parekhji SAP ID: 60004170014

**Date: 15th April, 2019**