

Universidad Francisco José de Caldas

FACULTAD DE INGENIERÍA DE SISTEMAS

PROYECTO:
GATO Y EL RATON C++

Programación Básica

Autor:

Henry Ricaurte Mora
Daniel Mateo Ballesteros Molina

2022-1

Índice

1. Librerías:	1
1.1. <code>#include <conio.h></code>	1
1.2. <code>#include <graphics.h></code>	1
1.3. <code>#include <vector></code> :	2
1.4. <code>#include <time.h></code> :	2
2. ¿En qué consiste el juego?	3
3. Realización del programa	3
3.1. Principio	4
3.2. <code>Analisis()</code>	5
3.3. <code>Ganoelmejor()</code>	5
3.4. <code>Inicio()</code>	6
3.5. <code>Reglas()</code>	6
3.6. <code>Caras()</code>	8
3.7. <code>Dados()</code>	9
3.8. <code>Escoje()</code>	10
3.9. <code>circulo()</code> — <code>equis2()</code> — <code>equis()</code>	11
3.10. <code>Turno()</code>	11
3.11. <code>TurnoM()</code>	14
3.12. <code>Teclado()</code>	15
3.13. <code>TirarH()</code>	16
3.14. <code>TirarM()</code>	17
3.15. <code>Juego()</code>	23
3.16. <code>Main()</code>	25
4. Resultado final.	26

1. Librerías:

En primera instancia, vamos a definir las siguientes librerías:

```
#include <conio.h>.
```

```
#include <graphics.h>.
```

```
#include <iostream>.
```

```
#include <vector>.
```

```
#include <time.h>.
```

Cada una de ellas conformada por un codio objeto permite hacer una gran variedad de cosas en el programa asignado, a continuacion veremos de que sirve cada una :

1.1. #include <conio.h>

Esta libreria se usa para promover un sistema de entrada y salida por consola.

Funcion	Descripción
kbhit	Determina si una tecla ha sido presionada
getch()	Lee un carácter directamente de la consola sin hacer uso del buffer y sin mostrar salida
getche	Igual que getch, pero muestra la salida
ungetch	Pone un carácter de vuelta al buffer de teclado
cgets	Lee una cadena de texto directamente de la consola
cputs	Escribe una cadena de texto directamente a la consola

1.2. #include <graphics.h>

Esta libreria esta diseñada para implementar un GUI (interfaz grafico de usuario) a c++, con diversas funciones que nos permite hacer, ya sean circulos, elipses lineas, bitmaps, o anidar imagenes, texto fuente, limpiar la pantalla, analizar la posicion del mauese, leer los clicks del mouse, dar color, y muchas mas, aunque es un poco "pobre" debido a que no cuenta con una gran capacidad, aun asi, sirve para crear graficas, y minijuegos.

Funcion	Descripcion
<code>closegraph();</code>	Sirve para poder cerrar las ventanas emergentes y dejar de consumir memoria
<code>initwindow(...);</code> <code>(Xi,Yi,"nombre",largo,ancho)</code>	Es la funcion que nos permite iniciar una ventana para poder incluir ahi los elementos graficos a usar, las imagenes y demas.
<code>readimagefile();</code> <code>("Nombre;mg",Xi,Yi,Xf,Yf)</code>	Sirve para leer las imagenes que estan en la misma carpeta que el main Y ubicarlas en algun espacio en la consola
<code>filellipse();</code>	Se usa para poder crear las elipses que son , en este programa, esencial
<code>setcolor(X);</code>	Se usa para poder darle color a los diferentes dibujos
<code>ismouseclick();</code>	Para leer el mouse, detectar, como <code>getch();</code> o <code>kbhit()</code>
<code>getmouseclick();</code>	Con este podemos obtener la posicion del mouse al hacer click

1.3. `#include <vector>`

La librería include vector funciona para poder crear vectores dinámicos, o simplemente para agarrar funciones que funcionan muy bien en el programa; al igual de poder hacer el uso de foreach y de funciones como `.push_back()` donde nos permite añadir o anidar posiciones a un vector.

La manera de inicializar un vector es como:

```
#include <vector>
#include <iostream>
using namespace std;
int main(){
vector<int> t; // vector<tipo_de_dato> nombre(tamaño,inicializacion);

t.push_back(6); // anida un 6 a la primera posicion.
}
```

1.4. `#include <time.h>`

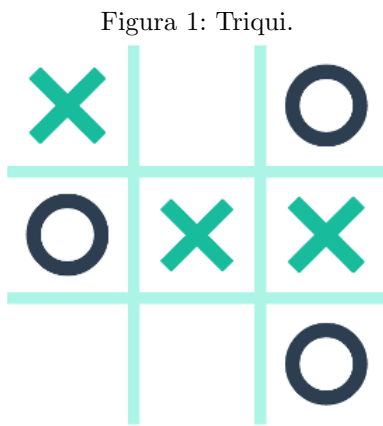
`time.h` relacionado con formato de hora y fecha es un archivo de cabecera de la biblioteca estándar del lenguaje de programación C que contiene funciones para manipular y formatear la fecha y hora del sistema. Usado en el programa para poder dar un numero aleatorio mediante la hora. por ejemplo:

```
#include <iostream>
#include <time.h>
using namespace std;
```

```
int main(){
    srand(time(0));
    int a = rand()%10 ;
}
```

2. ¿En qué consiste el juego?

El juego del Gato consiste mediante dos personas que, a través de un tablero 3x3, tiren por turnos en donde, en cada relevo, el participante coloca su ficha en alguna casilla, el primero en concretar 3 de estas en línea o en diagonal, gana.



Tenemos como finalidad que el programa sea capaz de comprender como ganar, y también como puede denegar el juego del rival, si no es capaz de analizar algún momento "ganador." "denegar", simplemente pone una posición aleatoria.

Las funciones usadas para este programa se dividen en secciones para hacer mucho más fácil el programa, de esta manera podemos organizar un poco más el programa.

3. Realizacion del programa

Tenemos las funciones:

- **analisis();** *tipo bool*
- **ganoelmejor();** *tipo void*
- **inicio();** *tipo void*
- **reglas();** *tipo int*
- **caras();** *tipo void*
- **dados();** *tipo bool*
- **escoje();** *tipo int*

- `circulo();` — `equis2();` — `equis();` *tipo void*
- `turno();` *tipo void*
- `turnoM();` *tipo void*
- `teclado();` *tipo int*
- `tirarH();` *tipo void*
- `tirarM();` *tipo int*
- `juego();` *tipo void*
- `main();` *main.*

A continuación, enfatizaremos cada una de las funciones donde explicaremos el código realizado.

3.1. Principio

Para empezar con el programa debemos hablar de las variables iniciales usadas:

```
#include <conio.h>
#include <graphics.h>
#include <iostream>
#include <vector>
#include <time.h>

using namespace std;

vector<bool> pos(9,false);

int matriz[3][3] = {-1};
```

En esta sección usamos las variables `pos` y `matriz`, las usamos para, en funciones posteriores, hacer que el programa entienda cuales posiciones están ocupadas (`true` para ocupadas), y, además, entender como funciona el triqui mediante la matriz 3 X 3.

3.2. Analisis()

En esta función es simple, si todas las posiciones están llenas, es decir, false, entonces retorna false, de lo contrario retorna true.

```
bool analisis(){
    int cont = 0;
    for(int i = 0 ; i <9; i++){
        if(pos[i] == true){
            cont++;
        }
    }
    if(cont == 9){
        return false;
    }

    return true;
}
```

3.3. Ganoelmejor()

Esta función es de tipo void debido a que solo necesito que me realice un proceso, en este caso, que me mande las diferentes pantallas, cuando gana el usuario, cuando gana la maquina y en el momento en que se quede empatados, todo gracias a la variable que lee al inicio.

```
void ganoelmejor(int a){
    closegraph();
    initwindow(1150,600,"GANADOR",500,200);
    setbkcolor(COLOR(100,399,152));
    cleardevice();
    if(a == 1){
        readimagefile("Imagenes/elmejor.jpg",0,0,1150,600); // colocar imagen
    }
    else if(a ==2 ){
        readimagefile("Imagenes/GANE.jpg",0,0,1150,600); // colocar imagen
    }
    else{
        readimagefile("Imagenes/empate.jpg",0,0,1150,600);
    }
}
```

3.4. Inicio()

La función inicio es bastante simple, lo único que hace es realizar la presentación de la primera pantalla, la inicial, en teoría iba a llevar una pequeña animación sin embargo se salía un poco de las manos debido a que se tenía que colocar un bitmap, guardarlo en una variable dinámica y con el uso de las funciones getimage(); putimage(); colocarla, sin embargo, generaba problemas a la hora de la lectura, tenía retrasos ya que la lectura del kbhit(), se realizaba cada movimiento y no constantemente.

```
void inicio(){
initwindow(1150,600,"INICIO",500,200);
cleardevice();
readimagefile("Imagenes/1.jpg",0,0,1150,600);
}
```

3.5. Reglas()

Dentro de la función reglas hacemos usos de dos for y varias condicionales en donde nos asegura la victoria de alguno de los dos mandos, cuando gane el humano retornara 1, cuando gane la maquina retornara 2 y si no gana nadie retornara 3, de esta forma aseguramos más adelante anidar esta función con otra.

```
int reglas(){ // reglas de la vida y de la muerte de triqui   ejeje
int contv = 0; // contador verticals
int conth= 0; // contador horizontal
int contvmachine = 0; // contador verticals maquina
int conthmachine = 0; // contador horizontal maquina

for(int i = 0; i < 3; i ++){

    // esto evalua las diagonales
    if(((matriz[2][0] == 2 && matriz[1][1] == 2 && matriz[0][2] == 2) || (matriz[0][0]
    == 2 && matriz[1][1] == 2 && matriz[2][2] == 2))){
//gano humano
getch();
ganoelmejor(1);
getch();
return 1;
// break;
}

    if(((matriz[2][0] == 1 && matriz[1][1] == 1 && matriz[0][2] == 1) || (matriz[0][0]
    == 1 && matriz[1][1] == 1 && matriz[2][2] == 1))){
//gano maquina
getch();
```



```

    ganoelmejor(2);
    getch();
    return 2;
//break;
}

for(int j = 0; j <3; j++){
    if (matriz[i][j] == 2){ // analiza los verticales humano
        contv++;
    }
    if (matriz[j][i] == 2){ // analiza los horizontales humano
        conth++;
    }
    //////////////////////////////////maquina //////////////////////////////////
    if (matriz[i][j] == 1){ // analiza los verticales
        contvmachine++;
    }
    if (matriz[j][i] == 1){ // analiza los horizontales
        conthmachine++;
    }
    //////////////////////////////////EVALUA GANADOR //////////////////////////////////
    if(contv == 3 || conth == 3){ // cualquiera de las dos gana
        getch();
ganoelmejor(1);
getch();
return 1;
    }
    if(contvmachine == 3 || conthmachine == 3){ // cualquiera de las dos gana
        getch();
        ganoelmejor(2);
        getch();
        return 2;
    }
}

contv = 0;
conth = 0;
contvmachine = 0;
conthmachine = 0;
}

if(! analisis()){
    ganoelmejor(3);
    getch();
}

return 3;
}

```

3.6. Caras()

Esta funcion es una de las funciones de diseño, en donde simplemente se crea el "objeto" de las caras de los dados, donde, dependiendo de 3 terminos nos genera uno para el bot y uno para el humano(interfaz).

```
void caras(int x, int xc1,int yc)
{
    int xc2=600;

    setcolor(WHITE);
    bar(xc1,yc,xc1+300,yc+300);
    int trama;
    trama = SOLID_FILL;
    if(xc1 < 300){
        setfillstyle( trama, BLACK );
    }
    else{
        setfillstyle( trama, WHITE);
    }

    switch(x){
case 1:
    fillellipse(xc1+150,yc+150,30,30); // uno
break;
case 2:
    fillellipse(((300))/3 +xc1,((300))/3 +yc, 30,30);
    fillellipse(((300)*2)/3 +xc1,((300*2))/3 +yc, 30,30);
break;
case 3:
    fillellipse(((300))/3 +(xc1-30),((300))/3 +(yc-30), 30,30);
    fillellipse(xc1+150,yc+150,30,30);
    fillellipse(((300)*2)/3 +(xc1+30),((300*2))/3 +(yc+30), 30,30);
break;
case 4:
    fillellipse(((300))/3 +xc1,((300))/3 +yc, 30,30);
    fillellipse(((300)*2)/3 +xc1,((300*2))/3 +yc, 30,30);
    fillellipse(((300)*2)/3 +xc1,((300))/3 +yc, 30,30);
    fillellipse(((300))/3 +xc1,((300*2))/3 +yc, 30,30);
break;
case 5:
    fillellipse(((300))/3 +(xc1-30),((300))/3 +(yc-30), 30,30);
    fillellipse(((300)*2)/3 +(xc1+30),((300))/3 +(yc-30), 30,30);
    fillellipse(((300)*2)/3 +(xc1+30),((300)*2)/3 +(yc+30), 30,30);
```

```

    fillellipse(((300))/3 +(xc1-30),((300*2))/3 +(yc+30), 30,30);
    fillellipse(xc1+150,yc+150,30,30);
break;
case 6:
    fillellipse(((300))/3 +(xc1-30),((300))/3 +(yc-30), 30,30);
    fillellipse(((300)*2)/3 +(xc1+30),((300))/3 +(yc-30), 30,30);
    fillellipse(((300)*2)/3 +(xc1+30),((300)*2)/3 +(yc+30), 30,30);
    fillellipse(((300))/3 +(xc1-30),((300*2))/3 +(yc+30), 30,30);
    fillellipse(xc1+70,yc+150,30,30);
    fillellipse(xc1+230,yc+150,30,30);
break;
}
getch();
}

```

3.7. Dados()

En esta función generamos la aleatoriedad de los dados, uno para la maquina y otro para el usuario.

```

bool dados(){
char r;
srand(time(0));
int a = 1;    // Dado mio
int a2 = 0;   // Dado maquina

initwindow(1000,600,"Dados",500,200);
cleardevice();
readimagefile("Imagenes/dado.jpg",0,0,1000,600);
getch();
do{
a = rand()%6 +1;
caras(a,100,200);
a2 = rand()%6 +1;
caras(a2,600,200);
if(a== a2){
    // poner imagen de ups, son iguales
}
}while(a==a2);
if(a> a2){
    closegraph();
return true; //true humano
}
else{

```

```

        closegraph();
        return false; // false maquina porque es falsa
    }

    return 0;
}

```

3.8. Escoje()

Esta función crea una pantalla , lee la imagen en donde podemos escojer un color, y gracias a la función ismouseclick() y getmouseclick(), podemos leer la posición y el click del mouse.

```

int escoje(){
int pete = 9;
initwindow(1000,600,"ESCOJE",500,200);
readimagefile("Imagenes/Escojer.jpg",0,0,1000,600);
int xm,ym; // declaramos las posiciones del mause
do{
if(ismouseclick(WM_LBUTTONDOWN)){ // para presionar el boton
    getmouseclick(WM_LBUTTONDOWN,xm,ym);

    if(xm>0 && xm<500){
        pete = 1;
        closegraph();
        return 1;
    }
    else{
        pete =1 ;
        closegraph();
        return 2;
    }
}
}while(pete!=1);
return 0;
}

```

3.9. círculo() — equis2() — equis()

Dentro de las funciones círculo, equis2, equis”, ponemos los dos colores disponibles para el usuario y el color predeterminado del bot, usamos diferentes colores para este, tambien leemos dos enteros en cada funcion para determinar la posición en x y en y dentro del programa.

```
void Ciculo(int x, int y){ ///////////////ciruclito para el humano -- azul
    int trama, color;
    trama = SOLID_FILL;
    color = 3; // 9
    setfillstyle( trama, color );

    fillellipse( x, y, 80, 80 );
}

void equis2(int x, int y){ // simbolo del pc -- Amarillo
    int trama, color;
    trama = SOLID_FILL;
    color = 6; // voy en el 12 verificando
    setfillstyle( trama, color ); // la forma, meh
    fillellipse( x, y, 80, 80 );
}

void equis(int x, int y){ // simbolo del pc -- VERDE
    int trama, color;
    trama = SOLID_FILL;
    color = 2;
    setfillstyle( trama, color );

    fillellipse( x, y, 80, 80 );
}
```

3.10. Turno()

En la función turno encontramos lo que es la lectura de los datos por parte del usuario, es decir, cuando digite “1”, por ejemplo, que use la matriz y el vector para leer y posicionar el dato que el usuario ha querido poner, en este caso, “1”. *Esto mediante un switch case que lee cada caso*

```
void turno(int tiro, int e){ // escojer
switch (tiro) {
case 1:
    switch (e){
    case 1:
```

```

        Ciculo(300,200);
        break;
    case 2:
        equis2(300,200);
        break;
    }

    pos[0] = true;
    matriz[0][0] = 2;        // para decir que ya se uso
break;
case 2:
    switch (e){
    case 1:
        Ciculo(500,200);
        break;
    case 2:
        equis2(500,200);
        break;
    }
    pos[1] = true;
    matriz[1][0] = 2;
break;
case 3:
    switch (e){
    case 1:
        Ciculo(700,200);
        break;
    case 2:
        equis2(700,200);
        break;
    }
    pos[2] = true;
    matriz[2][0] = 2;
break;
case 4:
    switch (e){
    case 1:
        Ciculo(300,400);
        break;
    case 2:
        equis2(300,400);
        break;
    }
    pos[3] = true;

```

```

    matriz[0][1] = 2;
break;
case 5:
    switch (e){
    case 1:
        Ciculo(500,400);
        break;
    case 2:
        equis2(500,400);
        break;
    }
    pos[4] = true;
    matriz[1][1] = 2;
break;
case 6:
    switch (e){
    case 1:
        Ciculo(700,400);
        break;
    case 2:
        equis2(700,400);
        break;
    }
    pos[5] = true;
    matriz[2][1] = 2;
break;
case 7:
    switch (e){
    case 1:
        Ciculo(300,600);
        break;
    case 2:
        equis2(300,600);
        break;
    }
    pos[6] = true;
    matriz[0][2] = 2;
break;
case 8:
    switch (e){
    case 1:
        Ciculo(500,600);
        break;
    case 2:

```

```

    equis2(500,600);
    break;
}

pos[7] = true;
matriz[1][2] = 2;
break;
case 9:

    switch (e){

        case 1:
            Ciculo(700,600);
            break;
        case 2:
            equis2(700,600);
            break;
    }

    pos[8] = true;
    matriz[2][2] = 2;

break;
}
}

```

3.11. TurnoM()

La funcion turnoM() tiene la misma funcionalidad que la funcion turno(), la diferencia es que en esta no lee la entrada del usuario, sino que lee la entrada del bot, que en una funcion mas adelante sabremos como se usa, y tambien esta la diferencia de que se usa el .equis” del bot.

```

void turnoM(int tiro){    // Los switch case son para colocar los simbolitos
switch (tiro) {
case 1:
    equis(300,200);
    pos[0] = true;
    matriz[0][0] = 1;
break;
case 2:
    equis(500,200);
    pos[1] = true;
    matriz[1][0] = 1;
break;
case 3:
    equis(700,200);
    pos[2] = true;

```



```

    matriz[2][0] = 1;
break;
case 4:
    equis(300,400);
    pos[3] = true;
    matriz[0][1] = 1;
break;
case 5:
    equis(500,400);
    pos[4] = true;
    matriz[1][1] = 1;
break;
case 6:
    equis(700,400);
    pos[5] = true;
    matriz[2][1] = 1;
break;
case 7:
    equis(300,600);
    pos[6] = true;
    matriz[0][2] = 1;
break;
case 8:
    equis(500,600);
    pos[7] = true;
    matriz[1][2] = 1;
break;
case 9:
    equis(700,600);
    pos[8] = true;
    matriz[2][2] = 1;
break;
}
}

```

3.12. Teclado()

La funcion teclado es utilizada para poder leer el teclado, es decir, a la hora de usar el boton '1', se coloque la posicion 1 del triqui.

```

int teclado(char r){ // para que me lea cosas de teclado, en casos
int aux;

switch(r){

```

```

case '1':
    aux =1 ;
break;
case '2':
    aux =2 ;
break;
case '3':
    aux =3;
break;
case '4':
    aux = 4;
break;
case '5':
    aux = 5;
break;
case '6':
    aux = 6;
break;
case '7':
    aux = 7;
break;
case '8':
    aux = 8;
break;
case '9':
    aux = 9;
break;
default:
    return 0;
break;
}
return aux;
}

```

3.13. TirarH()

Aquí es donde nosotros leemos lo que el usuario quiere hacer, y mediante la función turno() y teclado(), además del uso del vector y la matriz, hacemos que se haga realidad ese deseo.

```

void tirarH(int n){ ///////////////Tirar humano
    char r;
    int ala;
    do{

```

```

r = getch();
ala = teclado(r);
    // lee la posicion// verifica si esta o no ocupado

}while(pos[ala-1]==true);
    pos[ala-1]==true;    /////para ocupar ya el espacio
    turno(ala,n); // lo lee
    return;
}

```

3.14. TirarM()

Esta función es demasiado larga debido a que contiene los condicionales necesarios para que la máquina pueda pseudopensar de manera correcta, simplemente denegando cada que el humano vaya a ganar y ganando cada que esta tiene la oportunidad, y cuando no puede ni denegar ni ganar, crea un número aleatorio. En principio solo creaba un número aleatorio que ocupaba 10 líneas de espacio, sin embargo la pseudo-IA, requiere muchos condicionales para poder hacer un intento de la misma.

```

int tirarM(){ // Tirar maquina
int aux = 0;
int pensar = 0; // piensa  ejeje
int contador = 0; //contada cuantos cuadros quedan disponibles
int contador2 = 0;
int contador3 = 0;
srand(time(NULL));
///quitar diagonales
for(int i =0; i <3; i++){
    if(matriz[i][i] == 2){
        contador3++;
    }
    if(contador3 == 2){
        if(matriz[0][0] == -1){
            turnoM(1);
            return 1;
        } else if(matriz[1][1] == -1){
            turnoM(5);
            return 5;
        }
        else if(matriz[2][2] == -1){
            turnoM(9);
            return 9;
        }
        else{break;}
    }
}

```

```

    }
}
contador3 = 0;

//2,0 1,1 ,0,2

for(int i =0; i <3 ; i++){
    for(int j = 0; j < 3 ; j++){

        if(matriz[j][i] == 1){ // izq a derecha
            contador++;
            // cout<<" columnas: "<<contador<<endl;
        }
        if(matriz[i][j] == 1){ // arriba abajo 123/456/789
            contador2++;
            // cout<<" filas: "<<contador2<<endl;
        }

        ////////////analizamos las horizontales.
        if(contador == 2){ // analiza los contadores de las verticales
            switch(i){ // analisamos la i
            case 0:
                if(matriz[0][i] == -1 ){
                    turnoM(1);
                    return 1;
                }else if(matriz[1][i] == -1){
                    turnoM(2);
                    return 2;
                }else if(matriz[2][i] == -1){
                    turnoM(3);
                    return 3;
                }
                else{break;}
            break;
            case 1:
                if(matriz[0][i] == -1){
                    turnoM(4);
                    return 4;
                }
                else if(matriz[1][i] == -1 && !pos[6]){
                    turnoM(5);
                    return 5;
                }
                else if(matriz[2][i] == -1 ){

```

```

        turnoM(6);
        return 6;
    }else{break;}
break;
case 2:
    if(matriz[0][i] == -1 ){
        turnoM(7);
        return 7;
    }
    else if(matriz[1][i] == -1 ){
        turnoM(8);
        return 8;
    }
    else if(matriz[2][i] == -1){
        turnoM(9);
        return 9;
    }else{break;}
break;
    default:

        break;

}

}

if(contador2 == 2){ // analiza los contadores de las verticales
switch(i){ // analisamos la i
case 0:
    if(matriz[i][0] == -1 ){
        turnoM(1);
        return 1;
    }
    else if(matriz[i][1] == -1){
        turnoM(4);
        return 4;
    }
    else if(matriz[i][2] == -1 ){
        turnoM(7);
        return 7;
    }else{break;}
break;
    case 1:
        if(matriz[i][0] == -1){

```

```

        turnoM(2);
        return 2;
    }
    /*else if(matriz[i][2] == -1){
        turnoM(5);
        return 5;
    }*/
    else if(matriz[i][2] == -1){
        turnoM(8);
        return 8;
    }else{break;}
break;
    case 2:
        if(matriz[i][0] == -1){
            turnoM(3);
            return 3;
        }
        else if(matriz[i][1] == -1){
            turnoM(6);
            return 6;
        }
        else if(matriz[i][2] == -1){
            turnoM(9);
            return 9;
        }else{break;}
break;
    default:

        break;

}
}
}
contador = 0;
contador2 = 0;
}
for(int i =0; i <3 ; i++){
    for(int j = 0; j < 3 ; j++){

        if(matriz[j][i] == 2){ // izq a derecha
            contador++;
            // cout<<" columnas: "<<contador<<endl;
        }
        if(matriz[i][j] == 2){ // arriba abajo 123/456/789

```

```

        contador2++;
        //    cout<<" filas: "<<contador2<<endl;
    }

    ////////////analizamos las horizontales.
    if(contador == 2){    // analiza los contadores de las verticales

switch(i){ // analisamos la i
case 0:
    if(matriz[0][i] == -1 ){
        turnoM(1);
        return 1;
    }else if(matriz[1][i] == -1){
        turnoM(2);
        return 2;
    }else if(matriz[2][i] == -1){
        turnoM(3);
        return 3;
    }
    else{break;}
break;
    case 1:
    if(matriz[0][i] == -1){
        turnoM(4);
        return 4;
    }
    else if(matriz[1][i] == -1 && !pos[6]){
        turnoM(5);
        return 5;
    }
    else if(matriz[2][i] == -1 ){
        turnoM(6);
        return 6;
    }else{break;}
break;
    case 2:
    if(matriz[0][i] == -1 ){
        turnoM(7);
        return 7;
    }
    else if(matriz[1][i] == -1 ){
        turnoM(8);
        return 8;
    }
}

```

```

        else if(matriz[2][i] == -1){
            turnoM(9);
            return 9;
        }else{break;}
    break;

    default:

        break;

}

}

if(contador2 == 2){ // analiza los contadores de las verticales
switch(i){ // analisamos la i
case 0:
    if(matriz[i][0] == -1 ){
        turnoM(1);
        return 1;
    }
    else if(matriz[i][1] == -1){
        turnoM(4);
        return 4;
    }
    else if(matriz[i][2] == -1 ){
        turnoM(7);
        return 7;
    }else{break;}
break;
case 1:
    if(matriz[i][0] == -1){
        turnoM(2);
        return 2;
    }
    /*else if(matriz[i][2] == -1){
        turnoM(5);
        return 5;
    }*/
    else if(matriz[i][2] == -1){
        turnoM(8);
        return 8;
    }else{break;}
break;
case 2:

```



```

        if(matriz[i][0] == -1){
            turnoM(3);
            return 3;
        }
        else if(matriz[i][1] == -1){
            turnoM(6);
            return 6;
        }
        else if(matriz[i][2] == -1){
            turnoM(9);
            return 9;
        }else{break;}
    }break;
    default:

        break;

    }
}

}

}

contador = 0;
contador2 = 0;
}

////////////////////////////////////

do{        // verifica si esta o no ocupado
    aux =rand()%9+1;

}while(pos[aux-1]); //si estan usados se replantea otro numero random
    turnoM(aux);

return aux;
}

```

3.15. Juego()

En esta funcion define quien va primero, quien juega, como juega, y lee las demas funciones, tambien determina el ganador y si hay empate de manera compacta, es decir, como dato.

```

void juego(bool sociedad, int n){
    initwindow(1000,800,"JUEGO",500,200);
    setbkcolor(4);

```

```

cleardevice();
readimagefile("Imagenes/Fondo.jpg",0,0,1000,800);

// Barras verticales
setcolor( 4 );
bar(200,100,210,700);
bar(400,100,410,700);
bar(600,100,610,700);
bar(800,100,810,710);
// Barras horizontales
bar(200,100,800,110);
bar(200,300,800,310);
bar(200,500,800,510);
bar(200,700,800,710);
int cont = 0;
if(sociedad == true){
    do{
        tirarH(n);
        if(reglas() == 1 || reglas()==2 || ! analisis()){
            return;
        }
        tirarM();
        if(reglas() == 1 || reglas()==2 || ! analisis()){
            return;
        }
    }while(true);

}else {
//maquina
do{
    tirarM();
    if(reglas() == 1 || reglas()==2 || ! analisis()){
        return;
    }
    tirarH(n);
    if(reglas() == 1 || reglas()==2 || ! analisis()){
        return;
    }
}while(true);
}
}

```

3.16. Main()

Ya en el main podemos encontrar de manera compacta como funcionan todas las funciones anteriores, y con un do-while(true); hacemos que el programa se repita hasta que el usuario se salga.

```
int main(){
    //-----
    char m;
    bool p;
    //-----
    do{
        for(int i = 0; i < 9; i++){
            pos[i] = false;
        }

        for (int i = 0; i < 3 ; i ++){
            for(int j = 0; j < 3; j++){
                matriz[i][j] = -1; //cambiar a 0 para humano 1 maquina -1 y sumarlos
            }
        }
        closegraph();
        inicio();
        //bienvenido
        getch();
        closegraph();
        bool sociedad = dados();
        int n = escoje();
        juego(sociedad,n);
        //getch();
        //ganoelmejor(reglas());
    }while(true);
}
```

4. Reusltado final.

Como entrada principal tenemos:



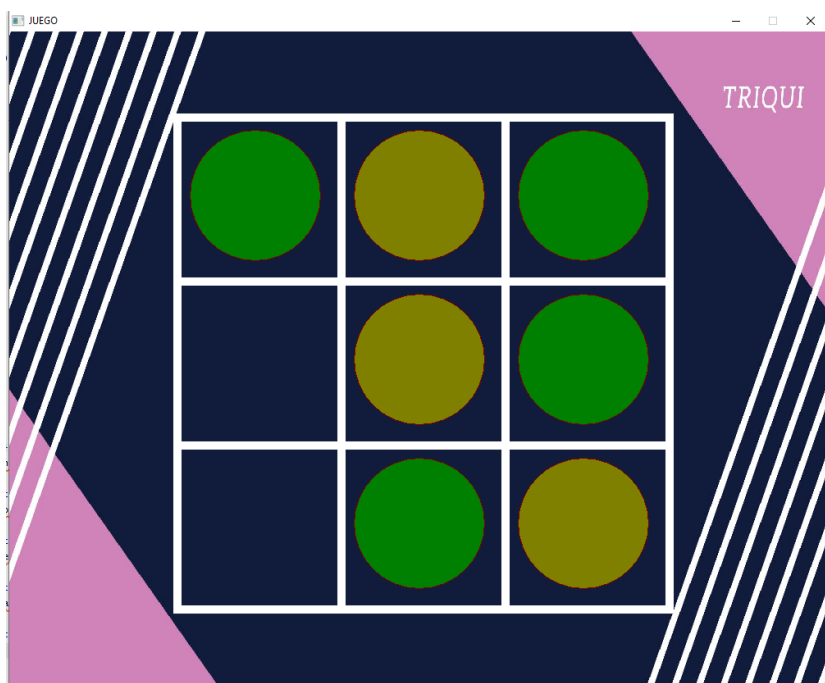
Luego, el interfaz de dados:



Para el interfaz de escoger un color:



Luego nos lleva al interfaz del juego, ya pudiendo jugar:



Para empate tenemos:



A la hora de perder:



Y a la hora de Ganar

