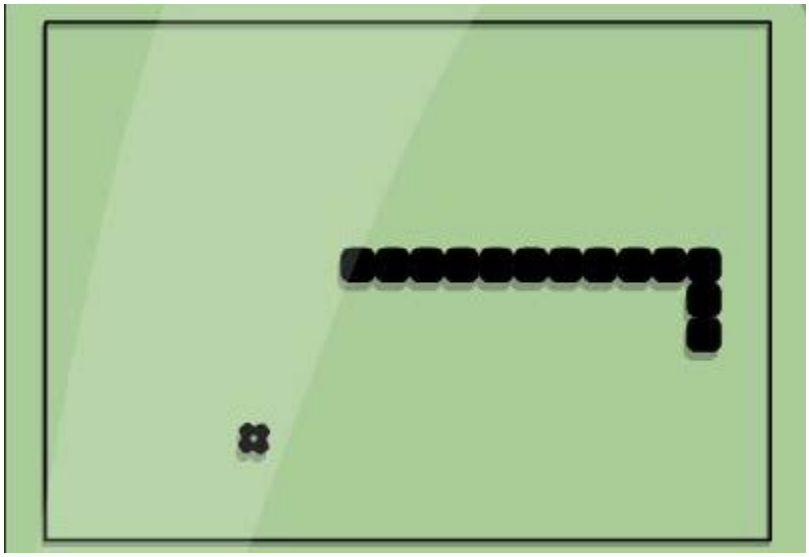
Proyecto final
Juego: SNAKE

Integrantes:
Julio Cesar Ovalle Lara
Carlos Arturo Murcia Andrade



Curso: Programación de computadores
Carrera: Ingeniería de sistemas y computación
Docente: Felipe Restrepo Calle
Universidad Nacional de Colombia

Bogotá D.C.
Fecha: 01/01/2014

## 1. Contenido



## 2. Tabla de ilustraciones

## 3. Introducción

Este proyecto es una imitación del clásico juego Snake.

Dicho proyecto fue hecho en base a la mayoría de los conceptos aprendidos a lo largo del curso de programación de computadores, y de otras funcionalidades que se explicaran a continuación.

Este documento consta de cuatro partes:
La primera aquí desarrollada es una breve introducción acerca del proyecto.
La segunda será una descripción detallada acerca de lo que se desarrollo en el proyecto, como sus funcionalidades, su alcance, sus restricciones, etc.
La tercera parte trata acerca de las conclusiones presentadas derivadas del desarrollo del proyecto.
Y la última parte consiste en las referencias bibliográficas que sirvieron de orientación en pro de la realización del proyecto.

## 4. Descripción del proyecto

### 4.1.Descripción general

Como fue previamente explicado a manera de esbozo en la introducción este proyecto consiste en la emulación del juego Snake o Culebrita…

El juego posee funcionalidades como la capacidad de mover la serpiente en diferentes direcciones; crecimiento de la serpiente a medida de que consume un alimento, por ende un aumento de la puntación y de la velocidad de movimiento de la misma, también la opciones de perdida (que la serpiente se chocara con las paredes o con si misma); la opción de ingresar un nombre de jugador y almacenarla en un documento si rompió el record de puntuación, pantallazos de inicio y final para hacer mas amigable la ejecución.

### 4.2.Alcance

Dicho proyecto pretende mostrar y demostrar lo aprendido en el curso de programación de computadores, temáticas tales como (manejo de funciones, matrices, vectores, cadenas de caracteres, flujos de datos…).

Inicialmente se pretendía que el proyecto pudiese mostrar uso mas avanzado de los conceptos mostrados en la asignatura tales como tipos de datos abstractos y aspectos gráficos pero dada la brevedad del tiempo… no fue posible implantar

dichas funcionalidades, por lo tanto esto es lo que el proyecto no pretende dar a conocer.

## 4.3.Funcionalidades implementadas

Mencionado anteriormente en la introducción, este proyecto posee las siguientes funcionalidades:

- ✓ Capacidad de movimiento de la serpiente.

```
//AQUI SE IMPRIME POR PRIMERA VEZ LA COMIDA DE LA SERPIENTE
gotoxy(pcx,pcy);
cuadro[pcy-4][pcx-28]=comida;
cout<<cuadro[pcy-4][pcx-28];
//IF DIRECCION 1
if(direccion==1){
    //CICLO MOVIMIENTO HORIZONTAL HACIA LA DERECHA
    while(x<79){
        // CONTROL DEL TECLADO
        /*EN ESTE PASO SE USA LA FUNCION KBHIT QUE RETORNA 0 SI NO SE
        HA PRESIONADO NINGUNA TECLA Y RETURNA 1 EN EL CASO CONTRARIO,
        ENTONCES EN CASO DE QUE SE HAYA PRESIONADO ALGUNA TECLA, ENTRARA
        EN EL IF DONDE DE INMEDIATO SE PASA EL VALOR QUE ESTABA EN EL BUFFER
        DEL TECLADO A UNA VARIABLE A POR MEDIO DE UN GETCH Y DESPUES TAMBIEN
        SE LE PASA ESE VALOR A UNA VARIABLE B, ESTO SE HACE PORQUE LAS TECLAS
        ESPECIALES COMO LAS FLECHAS DE DIRECCION, TIENEN UN VALOR DIFERENTE
        AL DE LAS TECLAS COMUNES EN ASCII YA QUE ES MAS LARGO Y NO SE PUEDE
        CAPTURAR EN UN SOLO GETCH, POR LO QUE SE USAN DOS PARA CAPTURAR EL VALOR
        DE LA TECLA.
        */
        if(kbhit()){
            a=getch();
            if(a==-32){
                b=getch();
            }
            if(b==72){
                direccion=4;
                break;
            }
            if(b==80){
                direccion=2;
                break;
            }
        }//FIN CONTROL DEL TECLADO
        //VERIFICACION CHOQUE SERPIENTE
        if(cuadro[y-4][x-28]==cuadrito){
            game=1;
            break;
        }
        if(x==78){
            gotoxy(x,y);
            cuadro[y-4][x-28]=cuadrito;
            cout<<cuadro[y-4][x-28];
            Sleep(velocidad);
            gotoxy(x,y);
            cuadro[y-4][x-28]=32;
            cout<<cuadro[y-4][x-28];
```

**Ilustración 1: Código movimiento serpiente.**

- ✓ Crecimiento de la serpiente a medida de que consuma un alimento.

```
    //IMPRESION DE LA SERPIENTE
    if(puntaje==0){
        cuadro[y-4][x-28]=cuadrito;
        gotoxy(x,y);
        cout<<cuadro[y-4][x-28];
        Sleep(velocidad);
        gotoxy(x,y);
        cuadro[y-4][x-28]=32;
        cout<<cuadro[y-4][x-28];
    }
    else{
        if(puntaant!=puntaje){
            cuadro[y-4][x-28]=cuadrito;
            gotoxy(x,y);
            cout<<cuadro[y-4][x-28];
            Sleep(velocidad);
            posiciones[pos]=x;
            pos++;
            posiciones[pos]=y;
            pos++;
            posiciones[pos]=0;
        }
        else{
            cuadro[y-4][x-28]=cuadrito;
            gotoxy(x,y);
            cout<<cuadro[y-4][x-28];
            Sleep(velocidad);
            posiciones[pos]=x;
            pos++;
            posiciones[pos]=y;
            pos++;
            posiciones[pos]=0;
            int u=0;
            while(posiciones[u]!=0){
                u++;
            }
            gotoxy(posiciones[u-((2*puntaje)+2)],posiciones[u-((2*puntaje)+1)]);
            cuadro[(posiciones[u-((2*puntaje)+1)])-4][(posiciones[u-((2*puntaje)+2)])-28]=32;
            cout<<cuadro[(posiciones[u-((2*puntaje)+1)])-4][(posiciones[u-((2*puntaje)+2)])-28];
        }
    }
    puntaant=puntaje;
    x++;
}//FIN CICLO MOVIMIENTO HORIZONTAL HACIA LA DERECHA
```

**Ilustración 2: Código crecimiento serpiente.**

- ✓ Con base a lo anterior que la velocidad aumente y por ende la dificultad.

```
            //VERIFICACION COMIDA DE LA SERPIENTE
            if((cuadro[y-4][x-28])==comida){
                puntaje++;
                velocidad-=1.2;
                gotoxy(52,2);
                cout<<puntaje;
                pcx=rand()%50+29;
                pcy=rand()%36+5;
                while(cuadro[pcy-4][pcx-28]==cuadrito){
                    pcx=rand()%50+29;
                    pcy=rand()%36+5;
                }
                gotoxy(pcx,pcy);
                cuadro[pcy-4][pcx-28]=comida;
                cout<<cuadro[pcy-4][pcx-28];
            }//FIN VERIFICACION COMIDA SERPIENTE
```

**Ilustración 3: Código aumento velocidad.**

- ✓ Opciones de pérdida.

```
///FIN CONTROL DEL TECLADO
//VERIFICACION CHOQUE SERPIENTE
if(cuadro[y-4][x-28]==cuadrito){
    game=1;
    break;
}
if(x==78){
    gotoxy(x,y);
    cuadro[y-4][x-28]=cuadrito;
    cout<<cuadro[y-4][x-28];
    Sleep(velocidad);
    gotoxy(x,y);
    cuadro[y-4][x-28]=32;
    cout<<cuadro[y-4][x-28];
    if(kbhit()){
        a=getch();
        if(a==-32){
            b=getch();
        }
        if(b==72){
            direccion=4;
            break;
        }
        if(b==80){
            direccion=2;
            break;
        }
    }
    game=1;
    break;
}
//FIN VERIFICACION CHOQUE SERPIENTE
```

**Ilustración 4: Código opciones de perdida.**

- ✓ La opción de ingresar un nombre y almacenarlo junto con la puntuación si rompió el record.

```
    char* lectura=new char[10];
    entrada>>lectura;
    int punthaje;
    entrada>>punthaje;
    if(puntaje>punthaje){
        cout<<"\n\n\n\n\n\n\n\t\t\t\tNUEVO RECORD"<<endl;
        cout<<"\n\n\t\t\t\t"<<nombre_jugador<<" Su Puntaje Fue:"<<puntaje<<"\n\n\t\t\t\tPRESIONE UNA TECLA PARA CONTINUAR";
        ofstream salida("puntajes.txt");
        salida<<nombre_jugador<<"\n"<<puntaje;
        getch();
    }else{
        cout<<"\n\n\n\n\n\n\t\t\t"<<nombre_jugador<<" Su Puntaje Fue:"<<puntaje;
        cout<<"\n\n\t\t\tNO HA PODIDO SUPERAR EL RECORD DE "<<lectura<<" QUE ES DE "<<punthaje;
        cout<<"\n\n\t\t\tPRESIONE UNA TECLA PARA CONTINUAR";
        getch();
    }
```

**Ilustración 5: Código nombre y puntuación.**

- ✓ Pantallazos de inicio y final para hacer mas amigable la ejecución:
  - o Inicio.

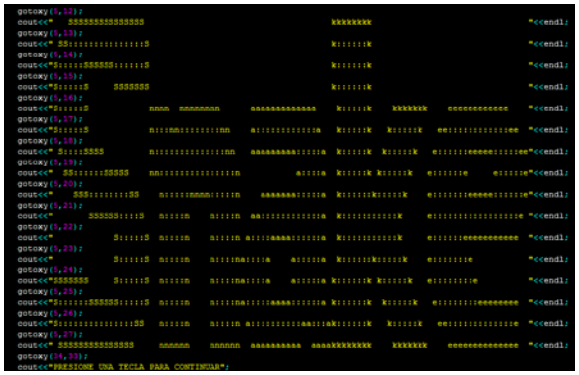


**Ilustración 6: Código pantalla inicial.**

- Final:

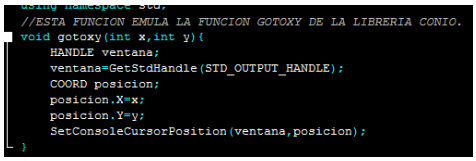
```
cout<<"\n\n\n";
cout<<"\t   gggggggg   ggggg aaaaaaaaaaaaa      mmmmmm    mmmmmm       eeeeeeeeeeee    "<<endl;
cout<<"\t  g:::::::::ggg::::g a:::::::::::a    mm:::::::m  m:::::::mm   ee::::::::::::ee  "<<endl;
cout<<"\t g:::::::::::::::::g aaaaaaaaa:::::a m::::::::::mm::::::::::m e::::::eeeee:::::ee"<<endl;
cout<<"\tg::::::ggggg:::::gg          a::::a m::::::::::::::::::::::::me::::::e     e:::::e"<<endl;
cout<<"\tg:::::g     g:::::g    aaaaaaa:::::a m::::mmm::::::mmm::::me:::::::eeeee:::::::e"<<endl;
cout<<"\tg:::::g     g:::::g  aa::::::::::::a m::::m   m::::m   m::::me:::::::::::::::::e "<<endl;
cout<<"\tg:::::g     g:::::g a::::aaaa::::::a m::::m   m::::m   m::::me::::::eeeeeeeeeee  "<<endl;
cout<<"\tg::::::g    g:::::ga::::a    a:::::a m::::m   m::::m   m::::me:::::::e           "<<endl;
cout<<"\tg:::::::ggggg:::::ga::::a    a:::::a m::::m   m::::m   m::::me::::::::e          "<<endl;
cout<<"\t g::::::::::::::::ga::::aaaa::::::a m::::m   m::::m   m::::m e::::::::eeeeeeee  "<<endl;
cout<<"\t  gg::::::::::::::g a::::::::::aa:::am::::m   m::::m   m::::m  ee::::::::::::e  "<<endl;
cout<<"\t    gggggggg::::::g  aaaaaaaaaa  aaaammmmmm   mmmmmm   mmmmmm    eeeeeeeeeeeeee  "<<endl;
cout<<"\t            g:::::g                                                              "<<endl;
cout<<"\tgggggg      g:::::g     ooooooooooo vvvvvvv           vvvvvvv eeeeeeeeeeee    rrrrr   rrrrrrrrr   "<<endl
cout<<"\tg:::::gg   gg:::::g   oo:::::::::::oov:::::v         v:::::vee::::::::::::ee  r::::rrr:::::::::r  "<<endl
cout<<"\tg::::::gg  gg:::::g  o:::::::::::::::ov:::::v       v:::::ve::::::eeeee:::::eer:::::::::::::::::r "<<endl
cout<<"\t g:::::::ggg:::::::g  o:::::ooooo:::::o v:::::v     v:::::ve::::::e     e:::::err::::::rrrrr::::::r"<<endl
cout<<"\t  gg:::::::::::::g   o::::o     o::::o  v:::::v   v:::::v e:::::::eeeee::::::e r:::::r     r:::::r"<<endl
cout<<"\t    ggg::::::ggg     o::::o     o::::o   v:::::v v:::::v  e:::::::::::::::::e  r:::::r     rrrrrrr"<<endl
cout<<"\t       gggggg        o::::o     o::::o    v:::::v:::::v   e::::::eeeeeeeeeee   r:::::r             "<<endl
cout<<"\t\t\t     o::::o     o::::o      v:::::::::v    e:::::::e            r:::::r            "<<endl;
cout<<"\t\t\t     o::::ooooo::::o       v:::::::v     e::::::::e           r:::::r            "<<endl;
cout<<"\t\t\t     o:::::::::::::o        v:::::v       e::::::::eeeeeeee   r:::::r            "<<endl;
cout<<"\t\t\t      oo:::::::::::oo        v:::v         ee::::::::::::e   r:::::r            "<<endl;
cout<<"\t\t\t        ooooooooooo           vvv            eeeeeeeeeeeeee   rrrrrrr            "<<endl;
```

Ilustración 7: Código pantalla final.

## 4.4.Restricciones

Durante el desarrollo del proyecto se encontraron las siguientes restricciones:

- La función para acceder a una posición de la consola gotoxy(), puede ser usada directamente en algunos compiladores, desafortunadamente en Code:Blocks y Dev C++ no funciona dicha función por la cual es necesario crear una función análoga que la emule, siendo esa la siguiente.



```
using namespace std;
//ESTA FUNCION EMULA LA FUNCION GOTOXY DE LA LIBRERIA CONIO.
void gotoxy(int x,int y){
    HANDLE ventana;
    ventana=GetStdHandle(STD_OUTPUT_HANDLE);
    COORD posicion;
    posicion.X=x;
    posicion.Y=y;
    SetConsoleCursorPosition(ventana,posicion);
}
```

Ilustración 8: Código función gotoxy().

- Ya que el programa fue diseñando bajo una configuración visual distinta se necesita, cambiar la configuración visual de la consola del compilador bajo los siguientes parámetros; buffer ancho = 113, alto = 43, tamaño ventana = 111, alto = 42, posición = (0,0), deseleccionar “sistema ubica ventana, fuente = mapa de bits, tamaño = 12*16.


Ilustración 9: Pantalla sin configuración visual adecuada.

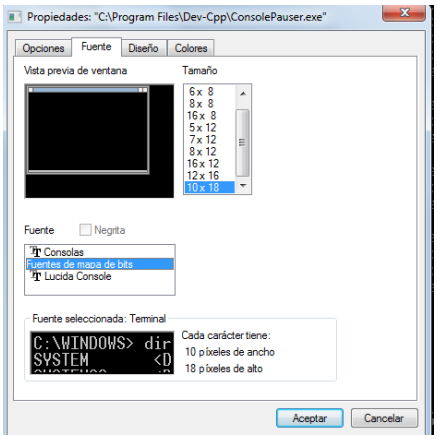



Ilustración 10: Configuración consola.

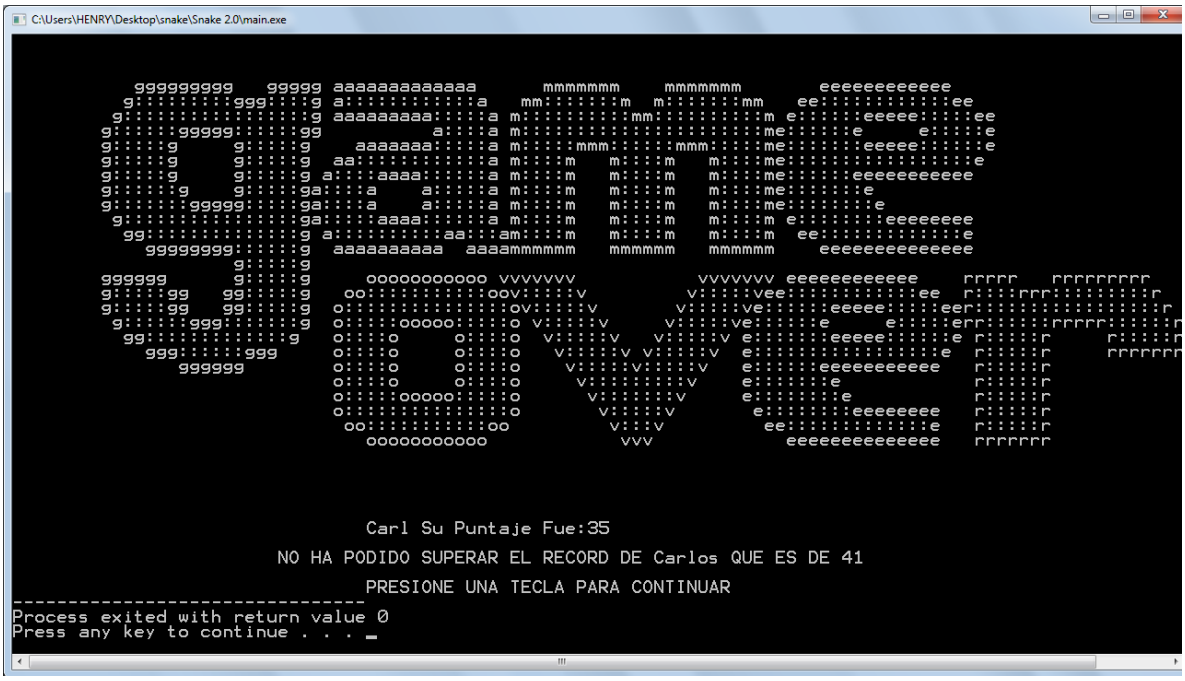



**Ilustración 11: Ventana con configuración visual recomendada.**

## 4.5.Suposiciones

Durante el desarrollo del programa surgieron las siguientes suposiciones:

- Para realizar el movimiento de la serpiente se podría utilizar una variable para el movimiento de la serpiente y capturar el movimiento con un getch(), sin embargo dicha suposición fue errónea ya que al ser las flechas de dirección un código ASCII no imprimible y además mas largo, entonces se procedió a usar la función kbhit() que retorna 0 si no se ha presionado alguna tecla y 1 en el caso contrario, entonces se manda la variable de movimiento a un getch() e inmediatamente a otra variable cuyo valor ira a un getch() para capturar la posición de la dirección.

```
// CONTROL DEL TECLADO
/*EN ESTE PASO SE USA LA FUNCION KBHIT QUE RETORNA 0 SI NO SE
HA PRESIONADO NINGUNA TECLA Y RETURNA 1 EN EL CASO CONTRARIO,
ENTONCES EN CASO DE QUE SE HAYA PRESIONADO ALGUNA TECLA, ENTRARA
EN EL IF DONDE DE INMEDIATO SE PASA EL VALOR QUE ESTABA EN EL BUFFER
DEL TECLADO A UNA VARIABLE A POR MEDIO DE UN GETCH Y DESPUES TAMBIEN
SE LE PASA ESE VALOR A UNA VARIABLE B, ESTO SE HACE PORQUE LAS TECLAS
ESPECIALES COMO LAS FLECHAS DE DIRECCION, TIENEN UN VALOR DIFERENTE
AL DE LAS TECLAS COMUNES EN ASCII YA QUE ES MAS LARGO Y NO SE PUEDE
CAPTURAR EN UN SOLO GETCH, POR LO QUE SE USAN DOS PARA CAPTURAR EL VALOR
DE LA TECLA.
*/
if(kbhit()){
    a=getch();
    if(a==-32){
        b=getch();
    }
    if(b==72){
        direccion=4;
        break;
    }
    if(b==80){
        direccion=2;
        break;
    }
```

**Ilustración 12: Código captura de las flechas de dirección.**

- Para hacer crecer la serpiente se pensó en crear un vector que almacenara cada vez que consumiera el alimento y que al moverse la ultima componente seria la primera, siendo dicha suposición correcta.

```
else{
    cuadro[y-4][x-28]=cuadrito;
    gotoxy(x,y);
    cout<<cuadro[y-4][x-28];
    Sleep(velocidad);
    posiciones[pos]=x;
    pos++;
    posiciones[pos]=y;
    pos++;
    posiciones[pos]=0;
    int u=0;
    while(posiciones[u]!=0){
        u++;
    }
    gotoxy(posiciones[u-((2*puntaje)+2)],posiciones[u-((2*puntaje)+1)]);
    cuadro[(posiciones[u-((2*puntaje)+1)])-4][(posiciones[u-((2*puntaje)+2)])-28]=32;
    cout<<cuadro[(posiciones[u-((2*puntaje)+1)])-4][(posiciones[u-((2*puntaje)+2)])-28];
}
```

**Ilustración 13: Código vector crecimiento.**

## 4.6.Dependencias

Este proyecto depende principalmente del juego original Snake y en las versiones de Nokia, a manera de ejemplo, hubo un caso presentado en las condiciones de choque de la serpiente, que cuando la serpiente crezca y quiera cambiar de dirección primero crece y luego con base a la variable tomada cambiaba su orientación generando un problema de que si la serpiente tomaba el alimento en la ultima casilla antes de la pared crecía hacia la misma sin poder cambiar de posición, para ello se soluciono aplicando una sentencia para cuando la serpiente estaba en el interior del tablero, y otra para cuando la serpiente estaba en los bordes.

```
//VERIFICACION CHOQUE SERPIENTE
if(cuadro[y-4][x-28]==cuadrito){
    game=1;
    break;
}
if(x==78){
    gotoxy(x,y);
    cuadro[y-4][x-28]=cuadrito;
    cout<<cuadro[y-4][x-28];
    Sleep(velocidad);
    gotoxy(x,y);
    cuadro[y-4][x-28]=32;
    cout<<cuadro[y-4][x-28];
    if(kbhit()){
        a=getch();
        if(a==-32){
            b=getch();
        }
        if(b==72){
            direccion=4;
            break;
        }
        if(b==80){
            direccion=2;
            break;
        }
    }
    game=1;
    break;
}
```

**Ilustración 14: Dependencia choque de la serpiente.**

Otro caso se presenta para el cambio de dirección cuando la serpiente se moviliza hacia por ejemplo la izquierda solo puede girar hacia arriba o hacia abajo, no puede dar reversa.

## 4.7.Ejemplo de funcionamiento

En esta sección se mostrara el funcionamiento del juego:

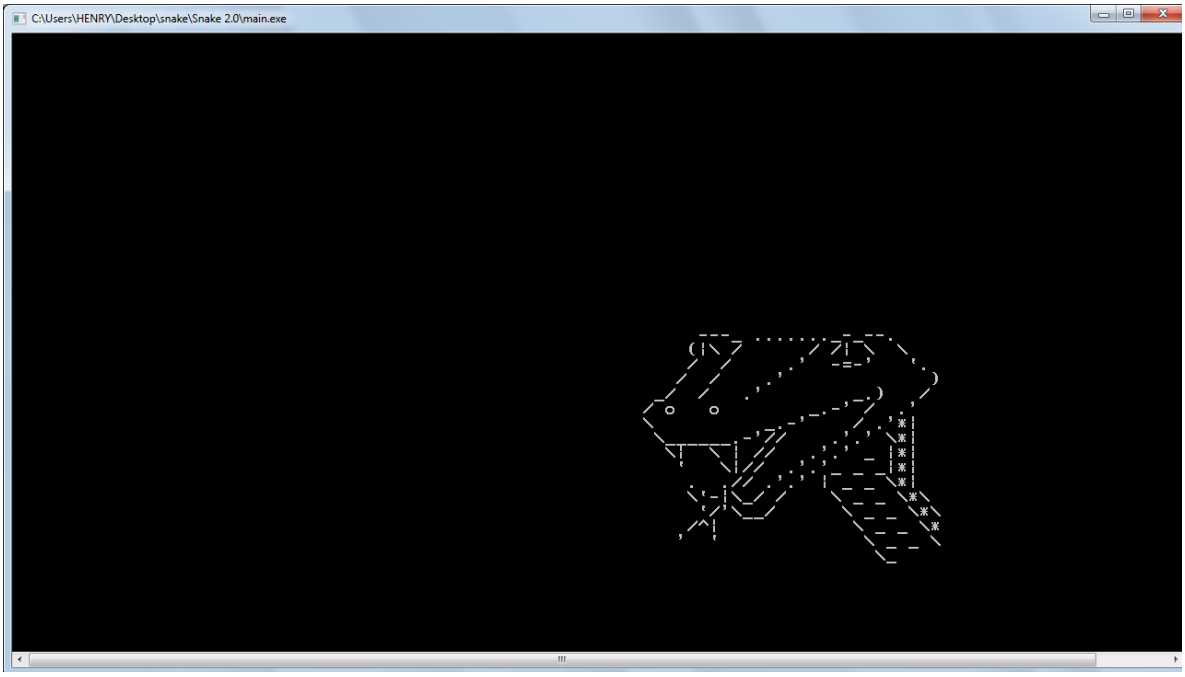



Ilustración 15: Pantallazo inicial 1.

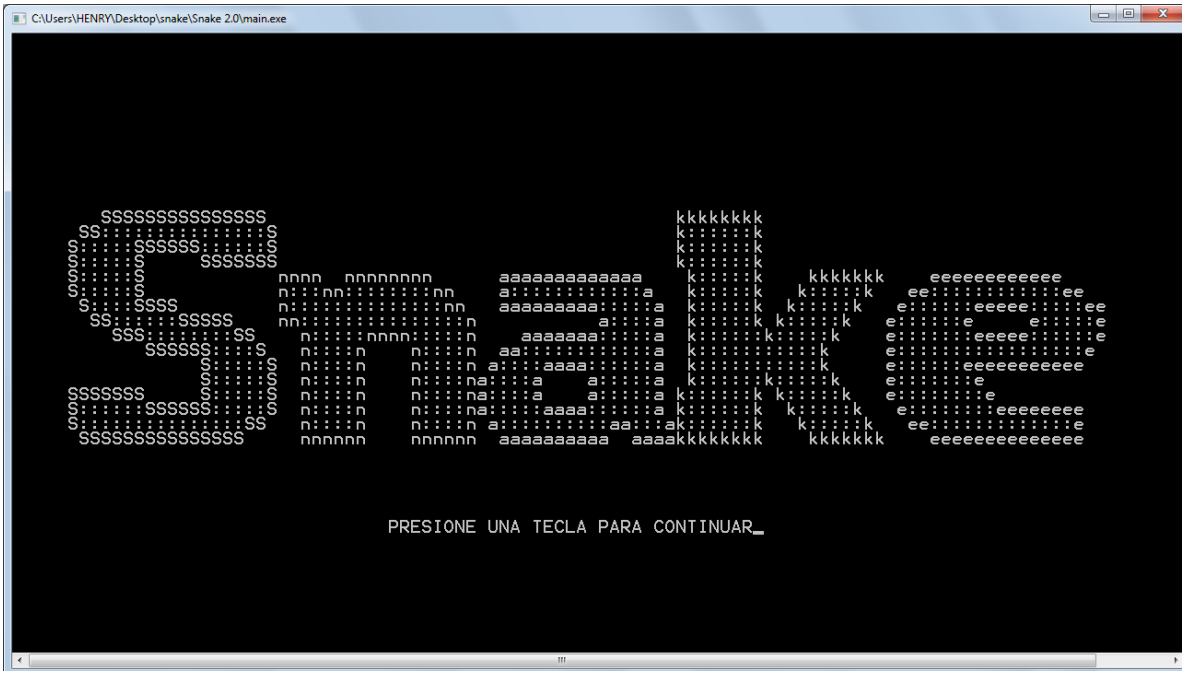



Ilustración 16: Pantallazo inicial 2.

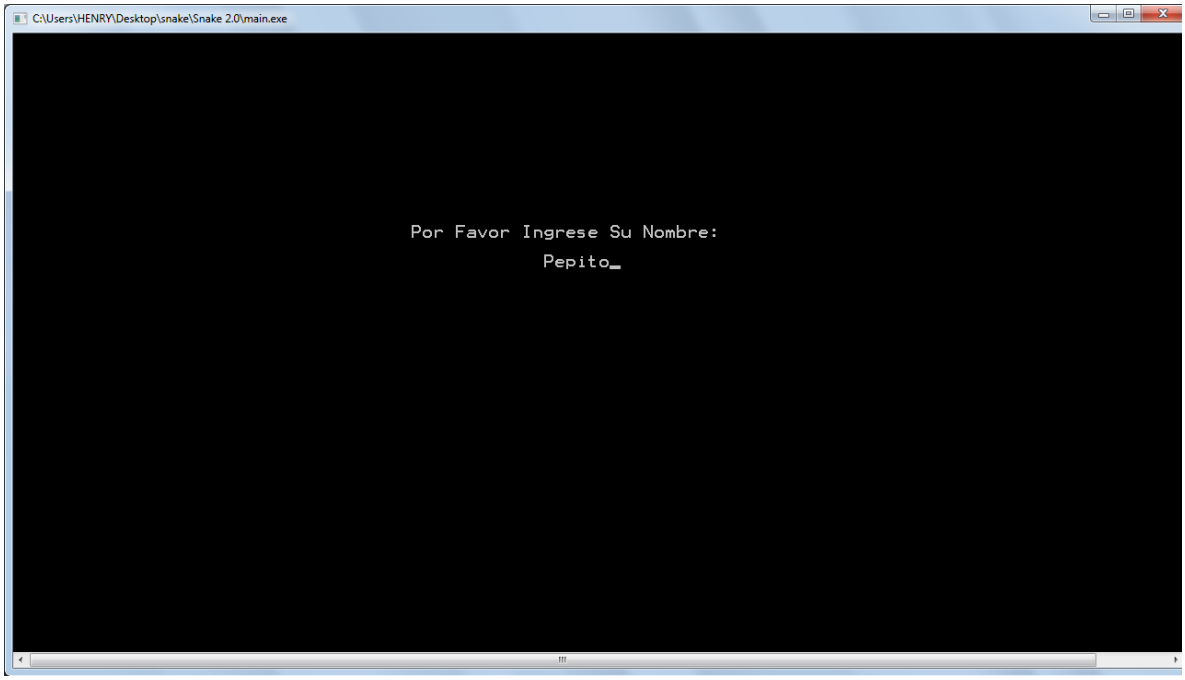



Ilustración 17: Ingreso de nombre.


Ilustración 9: Pantallazo inicial 3.

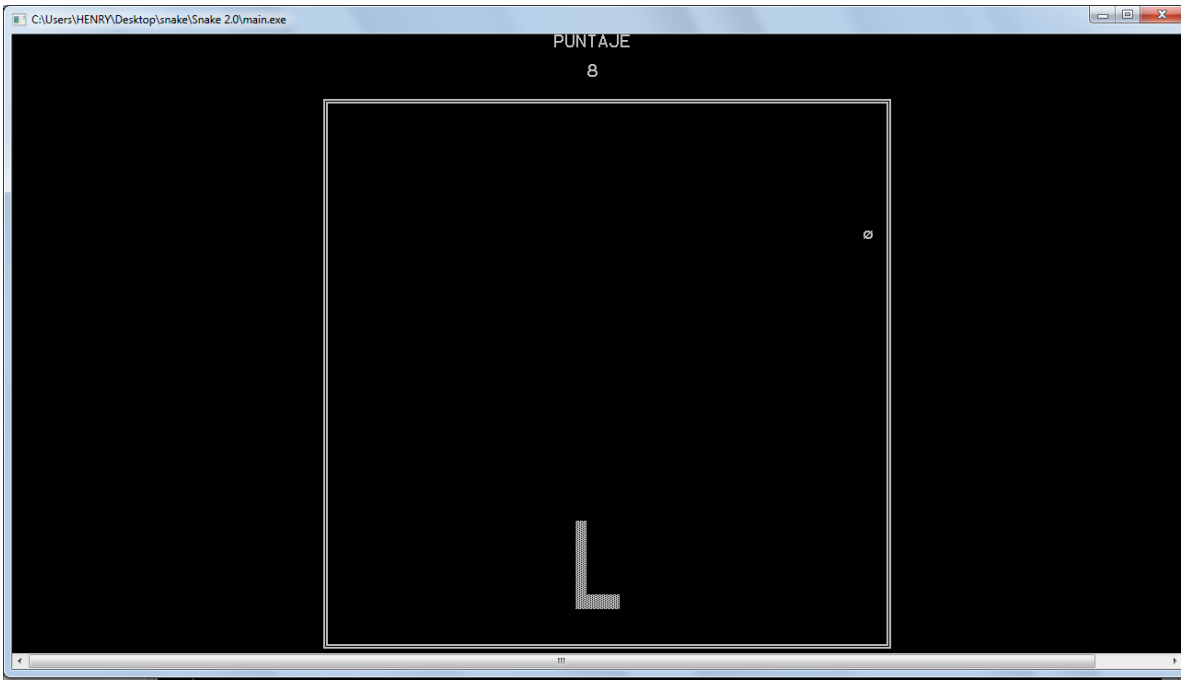



Ilustración 10: Juego en acción.

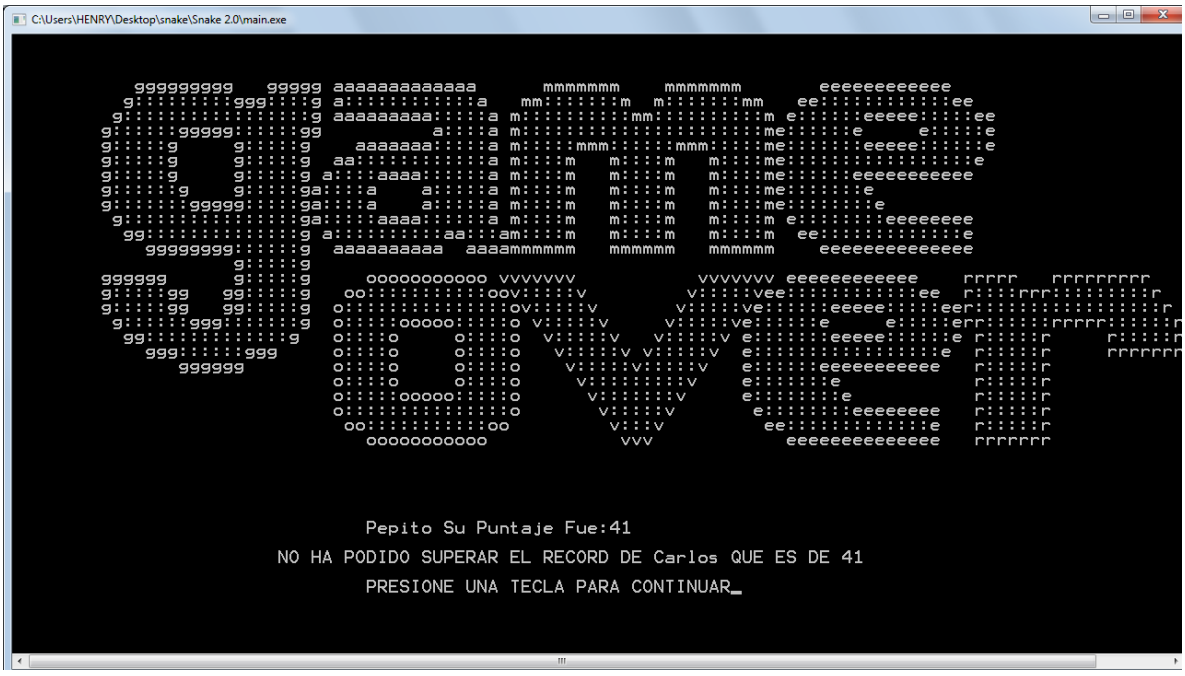



Ilustración 11: Juego terminado.

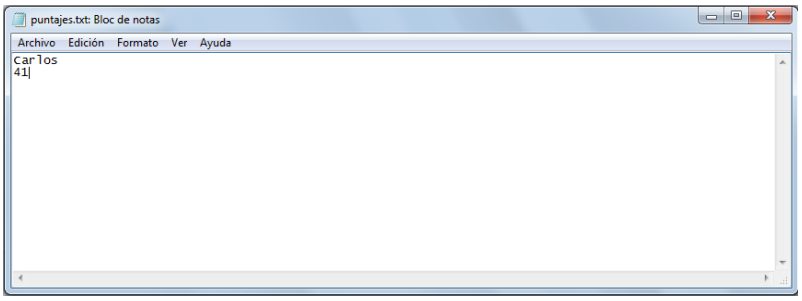



Ilustración 10: Puntaje generado.

## 4.8.Puntos fuertes

Este proyecto tiene los siguientes puntos fuertes:

- Posee una buena fundamentación teórica en cuanto lo aprendido en el curso.
- Tiene una interfaz amigable y manejable.
- Posee una documentación apropiada y fácil de entender para el usuario tanto en el programa, como en el texto mismo.

## 4.9.Puntos débiles

Este proyecto análogamente posee los siguientes puntos débiles:

- No posee elementos de tipo grafico, por ejemplo, inclusión de librerías de tal estilo.
- Para configurar la visualización de consola por primera vez se debe ingresar a las propiedades en tiempo de ejecución lo cual podría ser tedioso.

## 4.10. Ideas para futuras versiones

Si existiera la posibilidad de desarrollar una futura versión se tomarían en cuanta los siguientes aspectos:

- Incluir mejoras graficas.
- Incluir niveles de dificultad.
- Mostrar las mejores 10 puntuaciones y mostrarlas.
- Incluir otros juegos, o niveles de bonificación.

## 5. Conclusiones: Reflexión sobre el trabajo realizado

Durante la realización de dicho trabajo surgieron dificultades que impedían el progreso del proyecto, pero aun así con paciencia y esmero se ha logrado sacar adelante, por ende se puede concluir que.

La culminación de este trabajo ha dejado como resultado una profundización en el aprendizaje de los conceptos aprendidos en el curso, y se espera de igual forma incentivar nuestro deseo e interés de seguir con las demás materias en el transcurso de esta materia.

## 6. Referencias bibliográficas

Las siguientes fuentes sirvieron como base y complemento para la realización del proyecto:


© RESTREPO CALLE, Felipe, Estructuras de programación cíclicas, http://disi.unal.edu.co/~ferestrepoca/files/2014-2/programacion/presentacion-cap9_Ciclos_FRC.pdf

© RESTREPO CALLE, Felipe, Vectores o arreglos unidimensionales, http://disi.unal.edu.co/~ferestrepoca/files/2014-2/programacion/presentacion-cap10_vectores_FRC.pdf

© RESTREPO CALLE, Felipe, Cadenas de caracteres, http://disi.unal.edu.co/~ferestrepoca/files/2014-2/programacion/presentacion-cap11_cadenas_FRC.pdf

© RESTREPO CALLE, Felipe, Matrices o arreglos bidimensionales, http://disi.unal.edu.co/~ferestrepoca/files/2014-2/programacion/presentacion-cap12_matrices_FRC.pdf

© Crear la función gotoxy() en Dev C++, http://micodigocpp.blogspot.com/2013/11/crear-la-funcion-gotoxy-en-dev-c.html

© Sleep function, http://msdn.microsoft.com/en-us/library/windows/desktop/ms686298%28v=vs.85%29.aspx

© Sleep command in C++, http://www.dreamincode.net/forums/topic/16391-sleep-command-in-c/

© PREDOMO, RODRIGUEZ, CUBIDES, La ciencia de programar,