INSTITUTO POLITECNICO NACIONAL

Escuela Superior de Cómputo

Unidad de Aprendizaje: Algoritmos y Estructuras de Datos

Profesor: De Luna Caballero Roberto

Equipo:

Devora Guadarrama MaryJose

Mateo García Alejandra Michelle

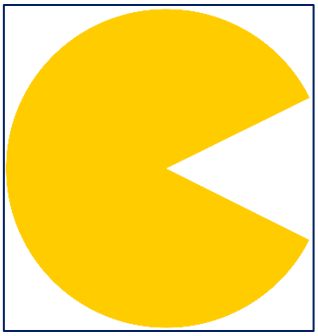
Martínez Jiménez Saul Jacob

Grupo: 2CM5

INDICE:

1. INTRODUCCION: PACMAN

Pacman es aquel juego que muchos en la infancia y adultez jugamos. Es adictivo, entretenido, incluso frustrante en muchas ocasiones.



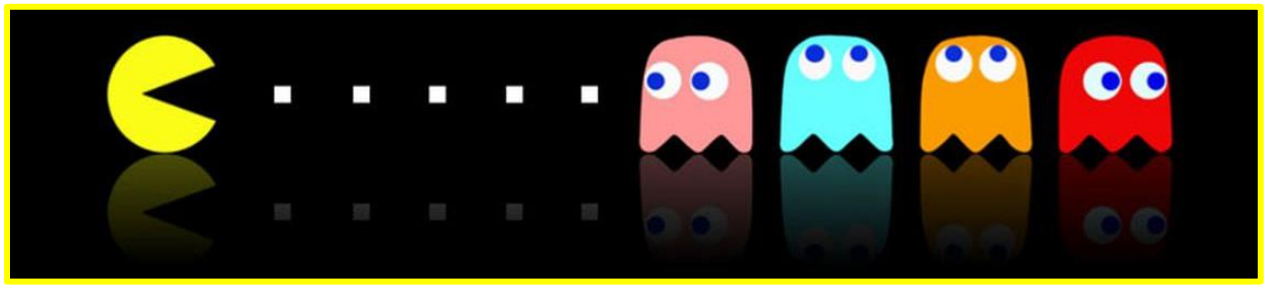
El nacimiento de Pacman se da de manera inesperada, todo fue gracias a una pizza que salió a comer a comer el creador de Pacman con sus amigos y cuando tomó el primer pedazo, fue cuando surgio la ideal del particular muñeco amarillo.

El creador de *Puck-man*, conocido en América con el título de Pacman, es el diseñador *Tōru Iwatani,* que fundó la compañía de software **Namco** en 1977.

El juego consiste en comer el mayor número de puntos sin dejarse atrapar por los fantasmas en un laberinto que se vuelve cada vez más complejo con los cambios de niveles. Un concepto muy sencillo pero adictivo.

Desde que PacMan fue lanzado el 21 de mayo de 1980, fue un éxito. Se convirtió en el primer fenómeno mundial en la industria de los videojuegos, llegó a tener el Récord Guiness del videojuego de arcade más exitoso de todos los tiempos con un total de 293.822 máquinas vendidas de 1981 a 1987.

La dinámica del juego que protagoniza es de sobra conocida y muy sencilla. Debe comerse todos los cocos (pasando sobre ellos) de un panel a modo de laberinto de dos dimensiones con el inconveniente de que cuatro fantasmas le buscan y acaban con él si le pillan. Pac-Man puede morir varias veces pues dispone de dos o tres vidas según la versión de juego. No obstante, se cambian las tornas y es Pac-Man quien puede acabar con ellos durante un breve periodo de tiempo cuando come 4 cocos más grandes que hay en el panel.



La velocidad y la inteligencia de los fantasmas aumenta según el nivel del juego, lo que acarrea mayor dificultad.

Para el desarrollo de este proyecto se ha recopilado mucha información de cómo es su jugabilidad, recopilación de varios programas y su explicación de cómo es su

programación y la realización del videojuego desde cero para poder empezar la programacion y su estructura.

De manera que este proyecto usando estructuras de datos que se han visto a lo largo de nuesto curso de algoritmos y estructuras de datos como los son la recursividad, uso de nodos, usando estructuras nuevas que no conociamos sin embargo tienen su propio funcionamiento como lo son las clases.

Continuando desde las partes de la jugabilidad, su estructura como videojuego, su programación, su seguimiento como programa y sobre todo su presentacion del resultado final de nuestro proyecto. El cuál es la serie de conocimientos adquiridos y aplicados a nuestra unidad de aprendizaje. Temas que a continuación ser irán explicando de mejor manera para entender su aplicación en nuestra era digital.

2. MARCO TEORICO

El nombre original de Pac Man era Puck-Man, pero los ejecutivos de Midway decidieron cambiarlo a Pac Man por miedo a que adolescentes fueran a vandalizar las cabinas del juego, cambiando la P por una F.

El nombre viene de la palabra Paku-Paku, una palabra japonesa que se usa para describir el movimiento y sonido de cómo se abre y cierra una boca al comer.

PACMAN tiene como objetivos comer todas las píldoras que hay en el laberinto sin ser capturado más de n veces. Los fantasmas tienen como objetivo capturar a PACMAN más de n veces. Hay píldoras especiales que hacen a los fantasmas vulnerables y que puedan ser capturados durante unos segundos, PACMAN puede decidir capturar a los fantasmas tras comer una de estas píldoras

Todos los fantasmas tienen tres comportamientos: persecución, asustado y dispersión.

Cuando un fantasma en modo de persecución su objetivo es eliminar a Pac Man; cuando este en modo asustado (activado cuando comes una pastilla de energía y los fantasmas se vuelven comestibles) su objetivo es huir de Pac Man y cuando está en modo dispersión su objetivo es regresar a su esquina de origen.

Conforme avanzabas de nivel los fantasmas pasaban más tiempo en modo persecución y menos tiempo en modo dispersión, el límite alto de esta dificultad era el nivel 21 ya que a partir de ahí ya no había cambios en el comportamiento de los fantasmas.

Cada uno de los fantasmas tiene una personalidad diferente; Blinky (Rojo) es el más agresivo y su objetivo siempre va a ser matar a Pac Man, casi siempre está en modo persecución.

Blinky es el único de los fantasmas cuya velocidad aumenta según vayas avanzando en el nivel. En Japón su personaje se describe como “Oikake” que se podria traducir como “Aquel que persigue”

Pinky (Rosa) siempre busca acorralar a Pac Man; su objetivo siempre es cortarle el camino a Pac Man para que Blinky pueda capturarlo. Incluso cuando está en modo de persecución, su objetivo nunca es comerse a Pac Man.

Esto se puede explotar fácilmente ya que si Pinky y Pac Man se encuentran por chocar de frente Pinky se va a salir del camino de Pac Man para intentar acorralarlo en la siguiente intersección. En Japon su personaje se describe como “machibuse” que se podría describir como “Aquel que hace emboscadas”

Clyde (Naranja), es el fantasma más tranquilo. Él se va a quedar en modo dispersión, dando vueltas en su esquina sin molestar a Pac Man a menos que Pac Man se acerque a menos de 8 cuadros de distancia de él, en ese caso Clyde se pone agresivo y entra

en modo persecucion. En Japon su personaje se describe como Otoboke, que se podria traducir a “Aquel que finge ignorancia”

Por último, Inky (Azul) es el fantasma más peligroso de todos porque es el menos predecible. Él está programado para replicar el comportamiento de alguno de sus hermanos al azar. Esto significa que puede estar tan tranquilo como Clyde y de un momento a otro ponerse agresivo como Blinky. En Japon se describe a su personaje como “Kimagure” que se podria traducir como “Aquel que cambia de humor”

Si lograbas llegar al nivel 256 del juego te encontrabas con un bug en el que la mitad de la pantalla se

veía normal, pero la otra mitad era un caos de letras y colores.

Esto sucedía por que 255 es el número más grande que cabe en un solo Byte y cuando el contador de niveles llegaba a 256 el CPU no sabía qué hacer y generaba una serie de errores que causaban el bug.

Pero lo más impresionante de todo, es que aun podías seguir jugando.

Si lograbas pasar el nivel 256 el juego regresaba al nivel 1 con tu puntuación intacta, pero el comportamiento de los fantasmas era como si hubieras llegado al nivel 21.

Definición de Clases

Una clase se puede definir con una estructura (struct), una unión (union) o una clase (class). En el tipo class todos los miembros son por defecto privados. Las estructuras y uniones son clases en las que todos sus miembros son por defecto public, a no ser que se especifique lo contrario utilizando los especificadores de acceso.

Una clase contiene elementos dato llamado miembros dato, y funciones que manipulan esos datos llamados miembros función o funciones miembro.

Objetos o instancias de una clase.

Los objetos o instancias de una clase se definen así:

nombre_clase instancia_1 //instancia_1 es una instancia de la clase tipo nombre_clase

Componentes.

La definición de una clase consta de dos partes: una declaración y una implementación. La declaración lista los miembros de la clase. La implementación o cuerpo define las funciones de la clase.

Declaración de una clase

```
class Contador {
        long cuenta;
public:
        void LeerValor(long);
        long ObtenerValor( );
```

Implementación de una clase

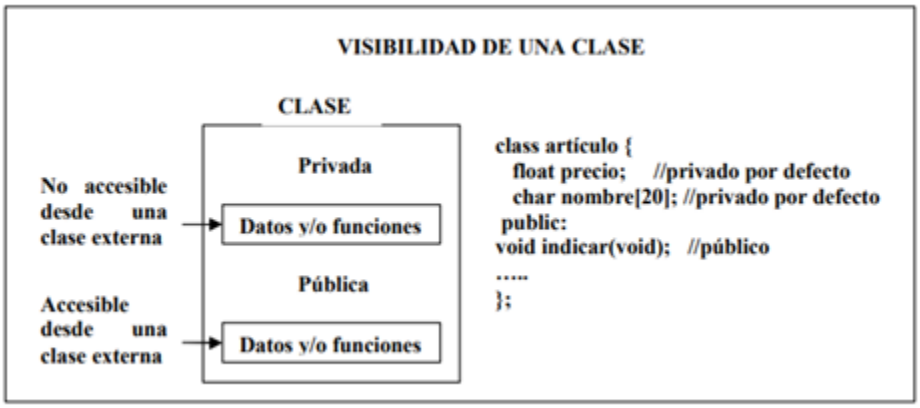
```
void Contador ::LeerValor(long valor)
{
    cuenta=valor;
}
long Contador ::  ObtenerValor( )
{
     return cuenta;
}
```

:: operador de resolución de ámbito, identifica la clase a la que pertenece la función.

Control de acceso a una clase: visibilidad

Una de las características fundamentales de una clase es ocultar tanta información como sea posible. Por consiguiente, es necesario imponer ciertas restricciones en el modo en que se puede manipular una clase y de cómo se puede utilizar los datos y el código dentro de una clase.

Una clase puede contener partes públicas y partes privadas. Por defecto, todos los miembros definidos en la clase son privados. Para hacer las partes de una clase públicas (esto es, accesible desde cualquier parte de su programa) deben declararse después de la palabra reservada public. Todas las variables o funciones definidas después de public son accesibles a las restantes funciones del programa. Especialmente, el resto de su programa accede a un objeto a través de sus funciones y datos públicos.



Función clock.
La función clock determina el tiempo usado del procesador.

Valor de retorno:

La función clock retorna la mejor aproximación por la implementación del tiempo del procesador usado por el programa desde el comienzo de un período, definido según la implementación, relacionado solamente a la invocación del programa. Para determinar el tiempo en segundos, el valor retornado por la función clock debería ser dividido por el valor de la macro CLOCKS_PER_SEC. Si el tiempo usado del procesador no está disponible o su valor no puede ser representado, la función retorna el valor (clock_t)-1.

Sintaxis: clock_t clock(void);

Función pieslice

Esta función es usada para dibujar y rellenar un una cuña circular. La cuña circular está centrada en el punto especificado por los argumentos x e y. La porción circular de la cuña comienza con el ángulo especificado por el argumento comienzo_angulo y se extiende en un sentido contrario a las agujas del reloj al ángulo especificado por el argumento final_angulo. La función pieslice considera este - el eje horizontal a la derecha del centro - como su punto de referencia de 0 grados. El perímetro de la cuña es dibujado con el color actual y es rellenado con la trama y color de relleno actual.

La función pieslice no retorna ningún valor.

void far pieslice(int x, int y, int comienzo_angulo, int final_angulo, int

3.DESARROLO DEL PROGRMA

```
#include<graphics.h>
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>
#include<dos.h>


#define bool     int
#define false    0
#define true     1
#define limite   0.3   //tiempo que se demoran los fanstasmas para pensar
#define sup_time 3.5   //tiempo en el que pacman es invencible cuando come
#define Ncol     15
#define Nfil     15


int px[14]={0,30,60,90,120,150,180,210,240,270,300,330,360,390};
int py[14]={0,30,60,90,120,150,180,210,240,270,300,330,360,390};


//matriz del camino del escenario

int table[Nfil][Ncol]={
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,1,1,1,1,1,1,1,1,1,1,1,1,1,0,
0,1,0,1,0,0,0,1,0,0,0,1,0,1,0,
0,1,0,1,0,0,0,1,0,0,0,1,0,1,0,
0,1,1,1,1,1,1,1,1,1,1,1,1,1,0,
0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,
0,1,0,0,1,1,1,1,1,1,1,0,0,1,0,
0,1,1,1,1,1,1,1,1,1,1,1,1,1,0,
0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,
0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,
0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,
0,1,1,1,1,1,1,1,1,1,1,1,1,1,0,
0,1,0,1,0,0,0,1,0,0,0,1,0,1,0,
0,1,1,1,1,1,1,1,1,1,1,1,1,1,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,};
```

En las primeras 7 líneas del código incluimos las librerías que utilizaremos; nuestro modo grafico se hizo en graphic.h y también incluimos stdio,conio,stdlib,math,time y dos(librería que se explicó en el marco teórico).

Después hicimos uso de definiciones para asignarles un valor y después usar esa palabra y no el valor numérico. Esto se hizo para hacerlo más entendible.

Después px y py son declaraciones de una serie de declaraciones de coordenadas que usaremos más adelante.

En la línea 44 y a partir de ella se hace lo mismo que se hizo con el tablero, se realizó con 0,1 y 2, cada uno representa la coordenada de galletas(que es lo que se come el pac-man) los 1 son galletas convencionales y los 2 son galletas grandes que convierten al pac-man invencible.

En la línea 61, declaramos unas variables de tipo entero y usamos otras variables de Clock que usaremos más adelante.

Aquí se presenta el primer Class, que como lo estudiamos es una declaración, esta declaración es de pacman que va a contener unas direcciones de enteros y se dibujó.

```
76  direccion dir;
77  pacman(int xx,int yy,direccion d){x=xx;y=yy;dir=d;}
78  void dibujar();
79  };
80
81  pacman pac(7,1,derecha);
82
83  void pacman::dibujar()
84  {
85   if(!indefensos){setcolor(14);setfillstyle(SOLID_FILL,14);} //esto es para que cambie de color
86   else{if(cc){setcolor(1);setfillstyle(SOLID_FILL,1);cc=false;}
87        else {setcolor(3);setfillstyle(SOLID_FILL,3);cc=true;}
88       }
89   switch(dir)
90        {case arriba:
91          pieslice(px[x]+15,py[y]+15,0,60,10);
92          pieslice(px[x]+15,py[y]+15,120,360,10);
93          break;
94      case abajo:
95          pieslice(px[x]+15,py[y]+15,0,240,10);
96          pieslice(px[x]+15,py[y]+15,300,360,10);
97          break;
98      case izquierda:
99          pieslice(px[x]+15,py[y]+15,0,150,10);
100         pieslice(px[x]+15,py[y]+15,210,360,10);
101         break;
102     case derecha:
103         pieslice(px[x]+15,py[y]+15,30,330,10);
104         break;
105       }
106 }
```

Ahora sigue la función void para dibujar pacman, en esta se llevan a cabo simplemente funciones del modo gráfico en el que dependiendo la dirección que tome el pacman se dibujará. También se debe tomar en cuenta que, como regla del juego, cuando el pacman come una galleta grande se vuelve invencible.

```
106
107
108 class fantasma{
109 public:
110 int x,y;
111 int color;
112 bool levantado; //si pacman es invencible y lo levanto
113 fantasma(int xx,int yy,int c){x=xx;y=yy;color=c;levantado=false;}
114 void direccionar();
115 void randomizar_();
116 void dibujar();
117 };
118
119 typedef fantasma* pfan;
120 pfan fan[7]; //array de punteros a objeto
121
122 void fantasma::dibujar()
123 {
124  if(levantado)
125     {  setfillstyle(SOLID_FILL,color);
126     bar(px[x]+11,py[y]+11,px[x]+14,py[y]+14);
127     bar(px[x]+16,py[y]+11,px[x]+19,py[y]+14);
128     }
129  else{  setfillstyle(SOLID_FILL,color);
130     bar(px[x]+10,py[y]+8,px[x]+20,py[y]+17);
131     bar(px[x]+6,py[y]+15,px[x]+11,py[y]+20);
132     bar(px[x]+13,py[y]+15,px[x]+16,py[y]+20);
133     bar(px[x]+18,py[y]+15,
134     px[x]+23,py[y]+20);
135     setfillstyle(SOLID_FILL,BLACK);
136     bar(px[x]+12,py[y]+11,px[x]+14,py[y]+14);
137     bar(px[x]+16,py[y]+11,px[x]+18,py[y]+14);
138     }
139 }
```

En la función de la línea 141, regresa un dato de tipo boleano, true si es que un fantasma se puede mover de un punto a otro y false si no puede. Este va de la mano con la función randomizar, que traza la trayectoria que un fantasma invencible sigue. Usa la función es posible para indicarle si puede moverse, se le suma o resta valores de x y y.

```
140
141  bool es_posible(int xx,int yy) //dice si un fantasma puede moverse al punto
142  {                              //indicado
143   if(table[yy][xx]==0)return false;
144   else{
145        for(register int i=0;i<Nfan;i++)
146       {if(fan[i]->x==xx&&fan[i]->y==yy)return false;
147       }
148       }
149   return true;
150  }
151
152  //cuando pacman es invencible el fantasma sigue esta rutina
153
154  void fantasma::randomizar_()
155  {
156   if(es_posible(x,y+1))y++;
157   else if(es_posible(x,y-1))y--;
158   else if(es_posible(x-1,y))x--;
159   else if(es_posible(x+1,y))x++;
160  }
161
162
```

Así como declaramos el tipo de dato pacman, definimos el tipo fantasma que declara el funcionamiento del fantasma el cual tiene unas coordenadas x,y que lleva en esta declaración, colores, si es que este no puede matar al pacman(porque este es invencible), también lleva unas funciones de direccionar que cambian el sentido de los fantasmas, otra que los hace random, y otra que los dibuja. Todas están funciones se explicarán más adelante cuando aparezcan en el código.

Las siguientes líneas es la función dibujar fantasma que dibuja en la consola y les asigna colores.

```
162
163  //los fantasmas siguen a pacman
164  void fantasma::direccionar()
165  {
166   if(x-pac.x>0)
167       {
168        if(y-pac.y>0)
169        {if(y-pac.y>x-pac.x)
170             {if(es_posible(x,y-1))y--;
171              else if(es_posible(x-1,y))x--;
172              else if(es_posible(x,y+1))y++;
173              else if(es_posible(x+1,y))x++;
174             }
175            else{if(es_posible(x-1,y))x--;
176              else if(es_posible(x,y-1))y--;
177              else if(es_posible(x,y+1))y++;
178              else if(es_posible(x+1,y))x++;
179             }
180        }
181        else{if(abs(y-pac.y)>x-pac.x)
182             {if(es_posible(x,y+1))y++;
183              else if(es_posible(x-1,y))x--;
184              else if(es_posible(x,y-1))y--;
185              else if(es_posible(x+1,y))x++;
186             }
187            else{if(es_posible(x-1,y))x--;
188              else if(es_posible(x,y+1))y++;
189              else if(es_posible(x,y-1))y--;
190              else if(es_posible(x+1,y))x++;
191             }
192        }
193       }
194   else{
195        if(y-pac.y>0)
196        {if(y-pac.y>abs(x-pac.x))
197             {if(es_posible(x,y-1))y--;
198              else if(es_posible(x+1,y))x++;
199              else if(es_posible(x,y+1))y++;
200              else if(es_posible(x-1,y))x--;
```

Ahora hacemos la función que moverá a los fantasmas en función de cómo se mueve el pacman, esta función evalúa si es posible moverse a ese lugar y los manda en ese camino, que están programados para seguir la dirección del pacman y hacer más difícil que te escapes.

[*] Pacman1.cpp

```
196         {if(y-pac.y>abs(x-pac.x))
197             {if(es_posible(x,y-1))y--;
198           else if(es_posible(x+1,y))x++;
199           else if(es_posible(x,y+1))y++;
200           else if(es_posible(x-1,y))x--;
201             }
202          else{if(es_posible(x+1,y))x++;
203           else if(es_posible(x,y-1))y--;
204           else if(es_posible(x,y+1))y++;
205           else if(es_posible(x-1,y))x--;
206             }
207         }
208         else{if(abs(y-pac.y)>abs(x-pac.x))
209             {if(es_posible(x,y+1))y++;
210           else if(es_posible(x+1,y))x++;
211           else if(es_posible(x,y-1))y--;
212           else if(es_posible(x-1,y))x--;
213             }
214          else{if(es_posible(x+1,y))x++;
215           else if(es_posible(x,y+1))y++;
216           else if(es_posible(x,y-1))y--;
217           else if(es_posible(x-1,y))x--;
218             }
219         }
220         }
221 }
222
223 void iniciar_modografico()
224 {
225  int gdriver = DETECT, gmode, errorcode;
226  initgraph(&gdriver, &gmode, "");
227  errorcode = graphresult();
228  if (errorcode != grOk)
229   {
230    printf("Graphics error: %s\n", grapherrormsg(errorcode));
231    printf("Press any key to halt:");
232    getch();
233    exit(1);
234   }
```

Encontramos unas simples funciones para iniciar el modo grafico en la función que lleva este nombre.

Se tiene previsto los mensajes de salida en caso de que el grafico fallara y sea necesario salir, esta función la usaremos en el main para inicializar.

```
238 //se grafica el tablero segun la matriz de caminos
239 void tablerito()
240 {setcolor(3);
241  int i,j;
242  for(j=0;j<Nfil;j++)
243     {for(i=0;i<Ncol;i++)
244     {if(table[j][i]==1)
245        {
246         if(table[j][i-1]==0)line(i*30,j*30,i*30,(j+1)*30);
247         if(table[j][i+1]==0)line((i+1)*30,j*30,(i+1)*30,(j+1)*30);
248         if(table[j-1][i]==0)line(i*30,j*30,(i+1)*30,j*30);
249         if(table[j+1][i]==0)line(i*30,(j+1)*30,(i+1)*30,(j+1)*30);
250        }
251     }
252     }
253  setcolor(15);
254  outtextxy(500,100,"PUNTOS");
255  outtextxy(500,150,"TIEMPO");
256 }
257
258 //convierte los putajes y el tiempo a cadenas y las imprime
259
260 void actualizar_puntaje()
261 {
262  char tim[5];
263  char pun[5];
264  itoa((int)total,tim,10);
265  itoa(puntaje,pun,10);
266  setfillstyle(1,0);
267  bar(500,120,550,130);
268  bar(500,170,550,180);
269  setcolor(15);
270  outtextxy(500,120,pun);
271  outtextxy(500,170,tim);
272 }
```

Ahora dibujaremos un tablerito que se llenará respecto a la matriz de caminos y que este mostrará en pantalla el resultado final del juego, se muestra al momento que perdiste y te muestra el tiempo y los puntos (galletas que comiste). Ahora, para que esta

Información sea actualizada y se le manden los datos correctos y totales de los puntajes se utiliza la siguiente función, itoa convierte de un dato entero a una cadena de caracteres que es el que veremos reflejados en nuestras pantallas.

[*] Pacman1.cpp

```
273
274  //dibuja galleta pequeña
275  void galletica(int xx,int yy)
276  {setcolor(BROWN);
277   circle(px[xx]+15,py[yy]+15,2);
278   setfillstyle(SOLID_FILL,BROWN);
279   floodfill(px[xx]+15,py[yy]+15,BROWN);
280  }
281
282  //dibuja galleta grande
283  void galletica_p(int xx,int yy)
284  {
285   setcolor(BROWN);
286   circle(px[xx]+15,py[yy]+15,5);
287   setfillstyle(SOLID_FILL,BROWN);
288   floodfill(px[xx]+15,py[yy]+15,BROWN);
289  }
290
291  //borra un cuadro del juego
292  void borra(int xx,int yy)
293  {setfillstyle(SOLID_FILL,0);
294   bar(px[xx]+2,py[yy]+2,px[xx]+28,py[yy]+28);
295  }
296
297
298
299  void leer_teclado()
300  {char tecla;
301   tecla=getch();
302   if(tecla==27)terminar=true;
303   else if(!tecla)
304      {
305       tecla=getch();
306       if(tecla==80&&table[pac.y+1][pac.x]==1){pac.y++;pac.dir=abajo;}
307       else if(tecla==72&&table[pac.y-1][pac.x]==1){pac.y--;pac.dir=arriba;}
308       else if(tecla==75&&table[pac.y][pac.x-1]==1){pac.x--;pac.dir=izquierda;}
309       else if(tecla==77&&table[pac.y][pac.x+1]==1){pac.x++;pac.dir=derecha;}
310      }
```

Las siguientes dos funciones, dibujan en el mapa unas galletas de dos tamaños diferentes, una de ellas que es galletica, que dibuja una galleta grande en las esquinas del mapa, que como he mencionado, estas funcionan como escudo que convierten el pacman invencible. Y luego están las galletas pequeñas que simplemente sirven para darle puntajes al pacman.

Ahora en la línea 299 empezamos a dar las instrucciones para leer del teclado, usaremos las flechas de arriba, abajo, izquierda y derecha, esta función lee los movimientos que hagamos en el buffer de teclado.

```
312
313  void perdio()
314  {
315      setcolor(15);
316      settextstyle(0,0,5);
317      outtextxy(100,100,"PERDISTE");
318  }
319
320  void gano()
321  {int col=0;
322   for(int i=0;i<60;i++)//Esto es para que cambie de color
323       {col++;
324        if(col==16)col=0;
325        delay(50);
326        setcolor(col);
327        settextstyle(0,0,5);
328        outtextxy(100,100,"GANASTE!");
329       }
330  }
331  //busca si un fantasma es atrapado por pacman invencible
332  void fan_muertos()
333  {for(int i=0;i<Nfan;i++)
334      {if(fan[i]->x==pac.x&&fan[i]->y==pac.y)
335         {fan[i]->levantado=true;
336      fan[i]->x=5+i;
337      fan[i]->y=6;
338      puntaje+=20;
339         }
340      }
341  }
342  //busca si pacman esta muerto
343  void buscar_ataque()
344  {for(int i=0;i<Nfan;i++)
345      {if(fan[i]->x==pac.x&&fan[i]->y==pac.y)
346         {perdio();
347      terminar=true;
348      break;
349         }
350      }
351  }
```

Las funciones PERDISTE Y GANASTE hacen básicamente lo mismo, las dos le asignan un colora, unas coordenadas x,y y un outtextxy que imprimirá estas palabras en el tablerito que hicimos más arriba.

En las siguientes líneas, primero buscamos si un fantasma es atrapado por el pacman invencible porque esto nos da puntaje extra (línea 338).

Si el pacman está muerto, porque un fantasma lo comio, será buscado por buscar_ataque() que si está muerto termina el juego.

```
353
354  //mueve todos los fantasmas
355  void actualizar_fantasmas()
356  {
357   for(int i=0;i<Nfan;i++)
358      {borra(fan[i]->x,fan[i]->y);
359       if(galleta[fan[i]->y][fan[i]->x]==1)galletica(fan[i]->x,fan[i]->y);
360       else if(galleta[fan[i]->y][fan[i]->x]==2)galletica_p(fan[i]->x,fan[i]->y);
361       if(!indefensos)fan[i]->direccionar();
362       else fan[i]->randomizar_();
363       fan[i]->dibujar();
364      }
365  }
366
```

Actualizar fantasmas es otra función que sirve para mover a todos los fantasmas, dibujarlos, direccionarlos y randomizarlos, siendo hasta esta función donde unimos las anteriores y hacemos que los 5 fantasmas lleven una trayectoria.

```
401  do{  //ciclo principal
402     inicio=clock();
403     do{
404        if(kbhit())//busca si hay algo en el buffer
405           {borra(pac.x,pac.y);
406         leer_teclado();
407         pac.dibujar();
408         if(!indefensos)buscar_ataque();
409         else fan_muertos();
410         if(terminar)break;
411         if(galleta[pac.y][pac.x]==1)
412           {
413            galleta[pac.y][pac.x]=0;
414            Ngalletas--;
415            puntaje+=5;
416           }
417         if(galleta[pac.y][pac.x]==2)
418           {
419            galleta[pac.y][pac.x]=0;
420            Ngalletas--;
421            puntaje+=10;
422            indefensos=true;
423            ini_sup=clock();
424           }
425           }
426        fin=clock();
427        if(indefensos)
428         {if(((fin-ini_sup)/CLK_TCK)>sup_time)
429           {indefensos=false;
430            for(i=0;i<Nfan;i++)fan[i]->levantado=false;
431           }
432         }
433        total=(fin-primer)/CLK_TCK;
434     }while((fin-inicio)/CLK_TCK<limite);
435     if(terminar)break;
436     actualizar_puntaje();
437     actualizar_fantasmas();
438     if(!indefensos)buscar_ataque();
439     else fan_muertos();
440     if(terminar)break;
441  }while(Ngalletas!=0);
442  if(Ngalletas==0&&!terminar)gano();
443  getch();
444  delay(3000);
445  for(i=0;i<Nfan;i++)delete fan[i]; //Se libera la memoria de los fantasmas
446  closegraph();
447 }
448
449
```

Inicializamos el do principal, este do tiene otro do anidado que lo primero que hace es buscar si hay algo en el buffer de teclado, dibuja el pacman, busa los ataques y también busca el momento en el que pac man está muerto con ayuda de ifs.

Hacemos funciones if para indicar que los pacman han comido galletas y que es necesario aumentar en el puntaje. También considera cuando se ha convertido invencible o que ha comido la galleta grande.

Termina el primer do y se debe actualizar fantasmas y puntaje, el do principal se termina cuando Ngalletas=0;

Se libera la memoria de los fantasmas y se cierra el gráfico.

en este caso

al final de haber

necesarias

s fantasmas de tipo

s diferentes y que

nte

que será una

man de juego. El cual

consola.

queño y dibujamos

correspondiente.

alizamos el juego por

lock.

4. CONCLUSIONES.

El proyecto, fue un trabajo de mucha investigación, ya que ocupábamos estructuras básicas de lo visto, además de nuevas funciones que encontramos, fueron varias horas de prueba de erros desde saber como hacer un mapa, la movilización, asi cpmo lo fueron las clases, las librerias y las funciones de time.h, buscamos la manera de que fuera implementado de la manera más clara y que nos fuer eficiente en implementar.

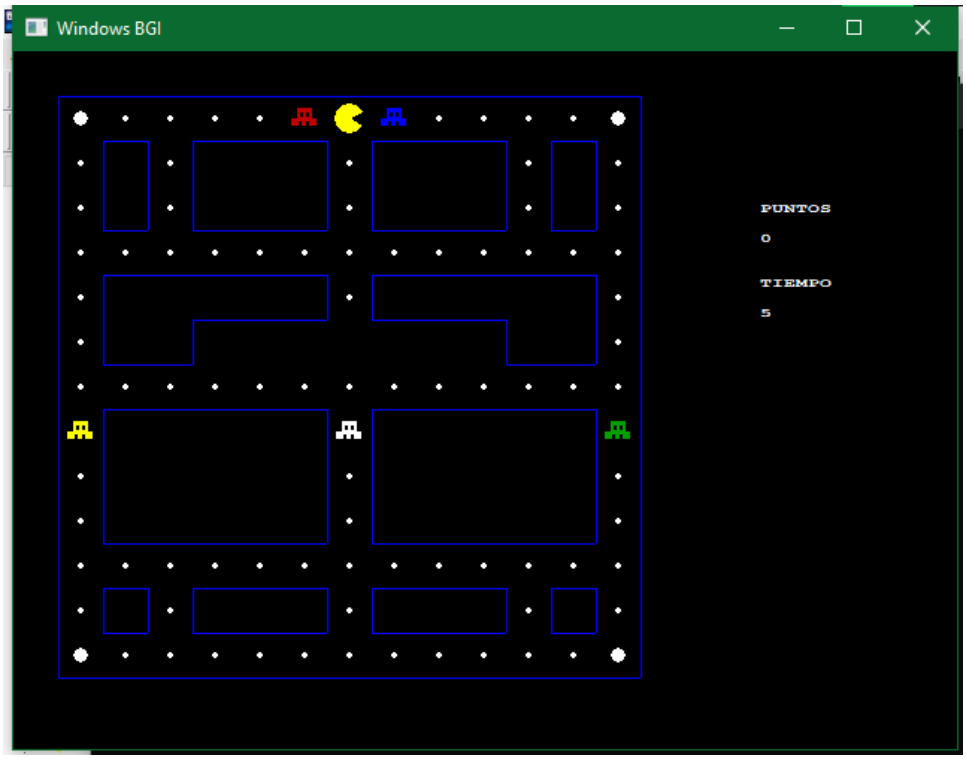
Realizar este programa que es un juego en consola fue un reto bastante interesante que tiene como objetivo que el pacman que nosotros creamos y dibujamos se coma todas las galletas sin ser atrapado por los fantasmas. Esto fue un reto por varias razones, la primera es que fue un trabajo en equipo, la comunicación y organización fue clave para realizar este programa de manera correcta porque a pesar de actualmente hay bastantes aplicaciones y maneras de trabajar a la distancia, sigue siendo un reto.

Aprendimos nuevas maneras de declarar estructuras, como class, que a pesar de que tomamos el riesgo de usarlas funcionaron bastante bien y nos ayudaron a declarar el tipo fantasma y pacman.

El cómo usar los datos de una manera diferente, ya que fue confuso por algunas cosas, al final todo fue teniendo sentido ya que estudiábamos mas el algoritmo e investigando, dando como resultado el icono de los videojuegos.

Nuestro aprendizaje se complementa hasta aquí y cierra con nuestro curso, mostrando el uso de las estructuras, los algoritmos y mucho más, comprendimos durante este curso para que se usan estas estructuras y lo fundamnetales que son para el resto de nuestras carreras.

Considero de mucha importancia que cada vez vayamos desarrollando proyectos de este tipo porque se acercan a como será la trabjarar con mas personas en la vida laboral, a pesar de que uno se buenos para ciertas cosas y otros sean buenos para otras, es justo la diversidad e


Windows BGI
PUNTOS
0
TIEMPO
5

1. **BIBLIOGRAFIA**

https://panoramacultural.com.co/tecnologia/3350/la-historia-de-pacman-el-primer-gran-exito-en-el-mundo-de-los-videojuegos

https://www.redbull.com/mx-es/5-cosas-que-no-sabias-de-pac-man

http://www.juegosdb.com/pac-man-el-comecocos-pc-ps3-xbox-360-wii-u-ps-vita-nintendo-3ds-moviles/

http://docs.mis-algoritmos.com/c.funcion.clock.html

http://conclase.net/c/librerias/time/clock

http://www.frlp.utn.edu.ar/materias/paradigmas/ApunteCmasmas.pdf

http://conclase.net/c/borland/graphics/pieslice

https://www.lawebdelprogramador.com/codigo/C-Visual-C/525-PacMan.html