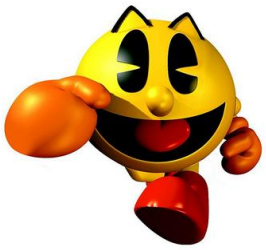

MEMORIA VIDEOJUEGO

- PACMAN –

Ignacio Arriscado García de Blas 12022
Cristina Fernández García 12117
Pablo Medrano Gastañaga 12258

- Roles de los miembros del grupo
- Planteamiento del algoritmo y estructuras de datos/clases utilizadas.
- Descripción general del algoritmo, funciones y estructuras de datos más representativas.
- Guía de uso del programa ejecutable

ROLES DE LOS MIEMBROS DEL GRUPO

- Algoritmo de movimiento Pacman: Ignacio
- Clases del código: Ignacio, Pablo, Cris
- Movimiento de fantasmas: Pablo, Cris
- Rotación cámara: Ignacio
- Subwindow: Pablo
- Salidas por pantalla: Cris
- Depuración código: Ignacio, Pablo, Cris

PLANTEAMIENTO DEL ALGORITMO Y CLASES UTILIZADAS

Comenzaremos por exponer las clases utilizadas en el algoritmo explicando en cada una de ellas los métodos que no se consideran triviales para facilitar su entendimiento.

Vector3D: Es una clase que engloba a 3 variables de tipo float que representan las tres coordenadas del espacio (x, y, z). Esto facilita el trabajo a la hora de trabajar con variables que sean vectores en las 3 dimensiones.

Cubo: La clase cubo tiene como atributos lado color y posición, y métodos para ajustar respectivamente el lado, el color y la posición, así como para dibujarse.
La posición es una variable de tipo Vector3D (relación de agregación entre clases). Los cubos constituirán el laberinto del juego.
En este punto es importante mencionar que hemos creado una matriz global llamada laberinto que contiene un 1 donde haya un cubo y un 0 donde no lo haya, esta matriz como iremos viendo será fundamental para el desarrollo lógico del algoritmo. Las filas y columnas guardan perfectamente analogía con los ejes X e Y utilizados para todas las referencias (el plano en el que se desarrolla el juego es el XY).Es decir, el elemento de laberinto de i=4,j=6 corresponderá con la posición (4,6) de el plano XY.

En el programa principal se instancia con una matriz de cubos llamada cubo, la definición de la misma seria: Cubo cubo[tam][tam].

Esfera: La clase esfera va a servir fundamentalmente para las esferas que tendrán que ir recogiéndose por el usuario, y también para el protagonista del juego, el pacman.
La clase contiene la variable radio de tipo float, un Vector 3D posición, ya explicado anteriormente, y color. Existen una serie de métodos triviales (los métodos con Set y Get que sirven para asignar o que el método nos devuelva lo correspondiente en cada uno de ellos, ya sea radio, posición…) contiene también las funciones 4 funciones para moverse (avanzar, retroceder, derecha e izquierda) a las cuales se les llama en el OnKeyBoardDown cada vez que el usuario pulse las respectivas teclas. Evidentemente estas funciones serán métodos del objeto esfera principal, que es el objeto móvil que se va moviendo por el plano.
El método Mueve, es el encargado de realizar una pequeña oscilación en las esferitas del laberinto que hay que recoger.

Tiene 2 métodos dibuja ya que no son iguales las esferas del laberinto que la esfera principal. Las funciones mayomenxizda y similares son para gestionar el movimiento y se explicarán explícitamente en el apartado de lógica de algoritmos.

La clase es 'friend class' de lista esferas.

Lista Esferas: Esta clase como su propio nombre indica es una clase que servirá como lista dinámica de esferas. Contiene un único atributo clave: una matriz de punteros a objetos tipo esfera. En el constructor se inicializa la lista con el **new** la lista de esferas allá donde no exista un cubo, para ello se le pasa por parámetro la matriz laberinto y se implementa que lista[i][j] apunte a NULL o a new esfera en función de que haya un 1 o un 0 en laberinto.
Contiene 2 métodos: Dibujar que lo que hace es situar(pintar) una esfera donde haya un hueco vacio y Eliminar que por un lado hacemos que el puntero que apuntaba a esa esfera ahora apunte a NULL y por otro eliminamos dinámicamente las esferas, esto ocurre cuando el usuario pase por alguna de ellas.

Fantasma: La clase fantasma es la encargada de dotar de forma y movimiento a los enemigos de pacman en el juego.
Como nuestros fantasmas son objetos de tipo cono esta clase contiene atributos base y altura, propios de un cono así como el Vector3D posición para situarlo en el espacio.
Como métodos contiene los triviales Set y Get que se explicaron por encima en la clase esfera y que en esta clase funcionan de la misma forma. Un método para dibujar los fantasmas con la posición, color, base y altura indicados. Y por último 4 métodos para el movimiento de cada uno de los cuatro fantasmas que funcionan por tiempo con variables que se van incrementando en el Ontimer.

OpenGL: Esta clase contiene el código necesario para que se pueda visualizar texto por pantalla. Al ser llamado el método de la clase deben pasarse como parámetros, el mensaje, colores y posición en el plano. Debe incluirse al igual que las otras en el main del videojuego.

Funciones: En esta clase lo que hemos hecho ha sido juntar todas las funciones que usamos a lo largo del código. Su funcionalidad es básicamente hacer lo más claro y sintetizado el programa. Las funciones son distancia y distancia2 que lo que hacen es calcular la distancia entre pacman y esferitas y pacman y fantasmas respectivamente para que uno de ellos se elimine si se encuentran en la misma posición, la función todo0 que inicializa a 0 la matriz y la función rotar.

DESCRIPCION GENERAL DEL ALGORITMO, FUNCIONES Y ESTRUCTURAS DE DATOS REPRESENTATIVAS

En este punto nos vamos a centrar en explicar un poco más detalladamente sobre todo el programa principal (pacman.cpp) sin hacer tanto hincapié en las clases de las que ya sabemos su funcionamiento.

En primer lugar, en el encabezado se incluyen todas las clases que se van a usar a lo largo del código así como las librerías .h que vayan a ser necesarias.
Detrás aparecen declaraciones de las funciones (OnDraw, OnTimer, OnKeyboardDown y demás…) y las declaraciones de las variables globales del código y las clases (esfera, cubo, fantasmas…)

Una vez llegamos al "main" tenemos la inicialización del gestor de ventanas de GLUT, a la que le ponemos la dimensiones deseadas. Creamos dos subwindows para poder jugar en tercera y primera persona al mismo tiempo, ambas dibujan lo mismo.

Asignamos atributos al pacman y los fantasmas con los métodos correspondientes de cada una de las clases

```
//se asigna atributos a los fantasmas y a la esfera
esfera.SetColor(255, 255, 0);
esfera.SetRadio(0.1);
esfera.SetPos(12.5, 12.5, 0);

fantasma1.SetColor(230,0,38);
fantasma1.SetBas(0.25);
fantasma1.SetHeight(1.5);
fantasma1.SetPos(1.5,23.5,-1.0);
```

Creamos después los cubos (que conformaran el laberinto) y las esferas condicionando a la lista de esferas que no apunte NULL donde no hay cubo.

Pasamos a la primera funcion OnDraw: en primer lugar se realiza la ubicación del ojo y del punto de vista en GLUT y tras esto con los métodos apropiados se dibujan cada uno de los objetos que hemos creado anteriormente.

```
//se dibujan los fantasmas y las esfera
esfera.Dibuja();
fantasma1.Dibuja();
fantasma2.Dibuja();
fantasma3.Dibuja();
fantasma4.Dibuja();
```

Dibujamos la lista de esferas, evidentemente se dibujaran solo en el caso de que no apunte a NULL y el laberinto haciendo uso de las funciones DibujarV y DibujarH.

```
void DibujarH(int fila, int columnai, int columnaf)
{
    for(int i=columnai; i<=columnaf; i++)
    {
        cubo[i][fila].Dibuja();
        laberinto[i][fila]=1;
    }
}
```

```
void DibujarV(int columna, int filai, int filaf)
{
    for(int i=filai; i<=filaf; i++)
    {
        cubo[columna][i].Dibuja();
        laberinto[columna][i]=1;
    }
}
```

Estas dos funciones tienen un código que funciona de la misma manera, la única diferencia es la dirección en la que se dibuja. Analizaremos uno de ellos a modo de ejemplo: en DibujarH metiendo como parámetros la fila en la que queremos dibujar y desde que columna hasta cual, con un bucle for dibujamos los cubos correspondientes que habían sido creados antes y ya tenían todo sus atributos.

Con este mismo código se realiza el OnDraw2 para la otra subventana que hemos creado.

También hemos creado una tercera función OnDraw3 que cuyo único objetivo es ir alternando cual de los otros dos OnDraw se dibuja en función de una variable bolean que alterna su valor cíclicamente cada 25ms gracias a que se realiza este cambio dentro de la función OnTimer.

Tras las funciones OnDraw tenemos la función Ontimer, que se ejecuta cada 25 ms y contiene los movimientos de los fantasmas que como ya hemos comentado su movimiento es a base de tiempos de recorrido del laberinto y estos se van incrementando en el Ontimer.
También aquí tenemos las funciones de distancia que ya hemos explicado y que se actualizan cada 25 ms para comprobar si se debe comer esferitas el pacman, o los fantasmas al pacman.

Por último tenemos el OnKeyBoardDown que se encarga del movimiento del usuario dentro del laberinto. Por un lado tenemos la lógica de rotación y por otro la lógica de movimiento.

Lógica de la rotación: Esta implementada mediante dos bucles if cada uno asociado a una tecla k, si es rotación a la derecha y j, si es rotación a la izquierda.

Básicamente hay una variable global d que sirve para saber la orientación en la que está en cada instante.
Así, al pulsar la tecla de rotar derecha o izquierda se rota el valor de la variable entre 1, 2, 3 y 4.
En cada estado al pulsar tecla se llama a una función `rotar' en la que se modifican referencia dos variables globales x1,y1 las cuales sirven de enlace para en el glookat sumar al parámetro que indica el punto al que miras el valor de x1 e y1, así se pasa de manera discreta de una orientación a otra.
Se crea entonces una especie de ciclo en el que hay 4 posibilidades. Cuando avanzas o retrocedes, se condiciona la actuación a como se esté orientado en ese instante, de este modo si pulsas por ejemplo avanzar hacia delante y está orientado hacia delante irá hacia delante, pero si está orientado hacia la derecha será avanzar 'hacia la derecha' ya que siempre se usan los mismos ejes para el movimiento. Así se condiciona de manera análoga en el resto de teclas de movimiento y se consigue así que para el usuario pulsar avanzar siempre sea avanzar, independientemente de hacia donde estemos mirando.

Además, en cada instante hay que estar comprobando si hay intersección con los cubos, ya que en ese caso no se deberá poder avanzar.
Al ser el movimiento continuo no pasamos de un cubo a otro sino que podemos estar cerca, o no, del cubo siguiente hacia donde queramos avanzar, de forma que cada celda en la que estemos se divide en una franja en la que siempre nos podremos mover en cualquier dirección, y otras 4 en las que no podremos movernos en todas direcciones.
Esto se consigue con 4 variables globales c1,c2,c3,c4 a las que se le asignan los métodos Mayomenxizda() Mayomenxdcha() ;Mayomenyarriba() Mayomenyabajo(). Estas funciones informan de si estás cerca o lejos de la siguiente celda.

Además, por ejemplo, si quieres moverte a la izquierda y estás orientado hacia delante podrás siempre que en a la izquierda no haya un cubo, esto es que haya un 0 una posición a la izquierda en la matriz laberinto. Utilizando dos variables globales que nos indican la posición en la que estamos a y b, que al ser enteras se logra discretizar el espacio, así, en cada acción de movimiento se condiciona a que haya un 0 en la dirección donde nos queremos mover, y en caso de que haya un 1 se añade la condición de que c4 sea 1 para poder avanzar (estamos cerca de la celda siguiente). Por tanto, si queremos avanzar hacia delante y hay un 1 delante si hemos sobrepasado la franja de 0.8L la variable c3 tomará el valor de 0 ya que el método Mayomenarriba comenzará a valer 0 en esa franja,por lo que no se entra al if que hay en cada implementación de el movimiento:

```
if(key=='w')
{
    if (d==1)
    {
        if(laberinto[a][b+1]==0)
        {
            esfera.Avanzar();
        }
        if((laberinto[a][b+1]==1)&&(c4==1))
        {
           esfera.Avanzar();
        }
    }
```

Se implementa de manera análoga para el resto de casos y se logra de esta manera el movimiento gracias al uso de variables globales que conectan unas situaciones con otra.

GUIA DE USO DEL PROGRAMA EJECUTABLE

El videojuego está basado en el conocido juego Pacman con la principal diferencia de que el juego se desarrolla en primera persona. Las instrucciones son sencillas: el usuario debe recorrer todo el mapa mientras recoge las esferitas prestando en todo momento atención a no ser comido por los fantasmas. El jugador gana si recoge todas y cada una de las esferitas, y pierde en el momento de que un fantasma le 'coma'. Las teclas que se deberán utilizar son:

- **d**→ mover derecha
- **a**→ mover izquierda
- **w**→ avanzar
- **s**→ retroceder
- **k**→ girar cámara 90 grados a la derecha
- **j**→ girar cámara 90 grados a la izquierda