Técnicas de Programación de Videojuegos

Banderas

Sirven para cambiar de un estado a otro, supongamos que estamos jugando la batalla naval, y tenemos dos modos de batalla, uno es recorrer el tablero en busca de un barco, el otro es ubicado un barco concentrarse en hundirlo.

Para indicarle al programa en que modo estamos podemos usar una variable que en la jerga informática se conoce como bandera (en ingles flag), básicamente es una variable común y corriente en la cual nosotros definimos los valores y que significan, por ejemplo, en nuestro ejemplo de la batalla naval, podríamos tomar 0 para buscar barcos y 1 para concentrarse en hundir, o declarar una variable char y tomar 'b' para buscar y 'h' para hundir.

Se usaría:

```
int flag;
...
...
if(flag==0)
{
	//rutina buscar barco
	...
	if (barco encontrado) flag=1; //cuando encontramos un barco pasamos
					// al modo hundirlo
}
else
{	// entra por flag==1
	// rutina hundir barco
	...
	if(barco hundido) flag=0; //cuando terminamos de hundir el barco
				// pasamos al modo buscar para encontrar el
				// siguiente
}
```

Las banderas no necesariamente pueden ser de dos estados como la del ejemplo, podríamos tener tres, cuatro o los que se necesiten.

Haciendo mapas

Los mapas en los videojuegos son las estructuras que sirven de base para ubicar los elementos del mismo, una de las estructuras más usadas son los arreglos, y de ellos las matrices.

Por ejemplo un mapa sencillo del juego tres en raya sería una matriz de 3 x 3, de char, en la que cada elemento de la matriz tuviera tres valores posibles, ' ' (espacio) 'X' o 'O', nuestro mapa representaría el estado del

juego en cada momento.

Complicando un poco más la cosa, en la batalla naval se usarían dos mapas (uno para cada jugador) de 10x10, donde estarían indicados los barcos, y los disparos realizados se pueden incluir tanto en este mediante una estructura o generar otra matriz de 10x10 que los registre.

primera solución 2 matrices de 10x10 una que lleve los barcos y otra que lleve los disparos:

Igual que en el caso de el tres en raya, debemos asignar valores para cada tipo de elemento que incluimos en el mapa (y así podemos distinguirlos), como en el anterior podemos usar números o char u otro tipo según conveniencia:

Podemos definir:

0 o ' ' para el agua
1 o 'P' para Portaviones
2 o 'A' para Acorazado
3 o 'C' para Crucero
4 o 'S' para Submarino
5 o 'D' para Destructor

Luego dentro de la matriz hay que colocar 5 unos o Pes seguidos para indicar los cuadrados del portaviones, 4 dos o A seguidos para indicar los cuadrados usados por el acorazado y así para cada tipo de barco, los cuadrados que no tienen barco se los llenan con 0 o espacios.

La segunda matriz es para marcar los disparos según va transcurriendo el juego, en este caso tenemos las posibilidades de: aún no se disparó, agua, tocado y hundido, igual que en el caso anterior podemos usar números o caracteres:

0 o ' ' para cuando aún no se disparó
1 o 'A' para Agua
2 o 'T' para Tocado
3 o 'H' para Hundido

Luego iniciamos esta segunda matriz con 0 o ' ' según corresponda y a medida que trascurre el juego la vamos llenando con los otros valores según corresponda.

Mediante una estructura

Podemos definir una estructura como:

```
typedef struct {
        int barcos;
        int tiros;
} mapa;
```

Y luego declaramos una matriz de 10x10 del tipo mapa:

```
mapa grilla1[10][10],grilla2[10][10]; //grilla1 para jugador 1 y grilla2 para jugador 2
```

Y usamos la misma codificación para los barcos (que en el ejemplo de dos matrices) para asignar en barcos y la misma codificación para marcar los disparos en tiros, por ejemplo queremos poner un Destructor en la posición A1 en horizontal para el jugador 1 sería:

```
grilla1[0][0].barcos=5;
grilla1[0][1].barcos=5;
```

Y por ejemplo si el jugador 2 efectúa un disparo sobre A1.

entonces modificaría su grilla como sigue:

```
grilla2[0][0].tiro=2;
```

Codificación

Una codificación es representar una cosa compleja con algo más simple de manera de poder manejarla mejor, por ejemplo, en nuestro mapa codificamos el agua con un número o una letra.
La codificación permite, hecha de manera inteligente, realizar operaciones más simples sobre los mapas (u otras estructuras), como comparaciones, modificaciones, etc.
La contra que tiene es que cuando hay que ingresar o informar algo, antes hay que hacer un proceso de codificación o decodificación según corresponda, por ejemplo:

Queremos imprimir el mapa que lleva los disparos de la batalla naval, queremos hacer una marca azul para agua, una roja para tocado y una marrón para hundido, entonces hay que traducir los códigos, por ejemplo:

```
for(int n=0;n<10;++n)
   for(int n=0;n<10;++n)
   {
      gotoxy(m+1,n+1);
      switch (grilla1[n][m].tiro){
            case 1:
```

```
            textbackground (BLUE);
            break;
         case 2:
            textbackground (RED);
            break;
         case 3:
            textbackground (BROWN);
            break;
         default:
            textbackground (BLACK);
      }
      cout<<" ";
   }
```

Por supuesto, algo más elaborado quedaría mejor, pero la idea es mostrar que con un if o un switch se puede traducir un código en otra cosa.

Estructuras y mapas

Hasta ahora hemos visto mapas simples, pero a medida que se agregan elementos a los mismos el uso de estructuras es de gran ayuda ya que sino habría que hacer tantas matrices como elementos tengamos:, por ejemplo:

si hiciéramos un mapa de una ciudad podría ser:

```
struct mapa{
      char descripcion[500]; //descripción del sector en la ciudad
      int paso; //bandera para saber si ya si el jugador ya pasó una vez por
                // este sector
      int objeto[10]; //hasta diez objetos diferentes puede haber en esa
                      // sector
};
```

Por supuesto, puede haber una estructura dentro de otra estructura, por ejemplo, los objetos podrían ser:

```
typedef struct{
      int codigo;  //codigo del objeto
      int x;         //coordenada x del objeto dentro del sector
      int y;         //coordenada y del objeto dentro del sector
} obj;

struct mapa{
      char descripcion[500]; //descripción del sector en la ciudad
      int paso; //bandera para saber si ya si el jugador ya pasó una vez por
                // este sector
      obj objeto[10]; //hasta diez objetos diferentes puede haber en ese
                      // sector
};
```