Luces (Lights Out)

Estudiante: Carlos Campos Martín

Asignatura: **Introducción a la programación**

Curso: **2010/2011**

Grupo de prácticas: A1b

Profesor: María Ángeles Mariscal Araujo

Convocatoria (febrero, junio, septiembre): Febrero

Índice de contenido

1. Introducción

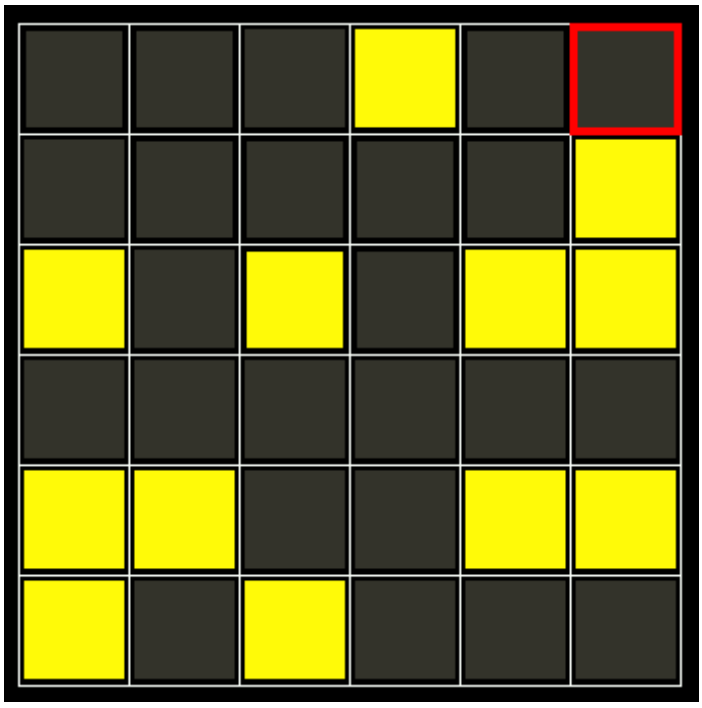
La versión del proyecto Lights Out que se entrega está basado en un juego popular en el que se parte de un tablero de dimensiones cuadradas con celdas cuyo contenido representa una bombilla que puede estar encendida o apagada. El fichero entregado posee, además del juego como tal, el juego con las ampliaciones realizadas.

1.1. Breve descripción sobre el funcionamiento del juego

El usuario puede marcar una posición de una celda del tablero y la bombilla que ocupa dicha posición y sus adyacentes cambiaran su estado (encendido por apagado y viceversa). El juego consta de dos padrones, los cuales pueden cambiarse en el transcurso de la partida. Según el patrón seleccionado cambiara el estado de las celdas arriba, abajo, derecha o por consiguiente cambiara el estado de toda la fila y columna donde se encuentra la celda marcada por usuario.

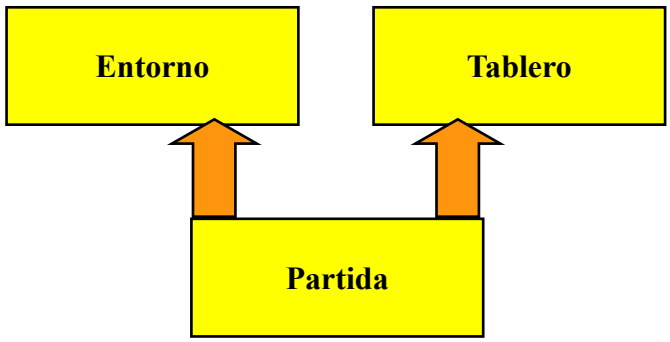
1.2. Objetivo

Partiendo de un tablero inicial con bombillas encendidas, el objetivo es ir seleccionando casillas hasta conseguir que se apaguen todas antes de agotar los intentos restantes.



2. Análisis y diseño

2.1. Diagrama modular



2.2. TAD Tablero

Encargado de gestionar la información del tablero: Inicialización, modificación y estado en cada momento de la partida.

2.2.1. Descripción del tipo de datos Tablero

Ttablero contiene:

-Una matriz de booleanos para alojar tantas casillas como se puedan definir con el máximo del tablero y que guardan la información de encendido y apagado .

-Un entero que se corresponde con el tamaño real del tablero.

2.2.2. Descripción de las operaciones

TAD tablero es :

PRE: MINTAMANIO <= Dimension <= MAXTAMANIO,

POST: Inicia el tablero de juego (de Dimension x Dimension) con todas las casillas apagadas (0).

Devuelve: Nada.

Complejidad: O(n²)

IniciarTodoApagado

PRE: MINTAMANIO <= Dimension <= MAXTAMANIO, 0<=Encendidas<=(Dimension x Dimension)

POST: Inicia el tablero de juego (de Dimension x Dimension) con un numero de casillas encendidas aleatoriamente.

Devuelve: Nada

Complejidad: O(n²)

IniciarTableroAleatorio,

PRE: MINTAMANIO <= Dimension <= MAXTAMANIO,

POST: Inicia el tablero de juego (de Dimension x Dimension) con una matriz leída desde un fichero de configuración externo.

Devuelve: Nada

Complejidad: O(n²)

IniciarTableroDeFichero,

PRE: Tablero iniciado correctamente. La casilla debe contener una valor correcto (booleano)

POST: Cambia el estado de la casilla[Fila][Columna]. Si es 0 a 1 y si es 1 a 0

Devuelve: Nada

Complejidad: O(1)

AccionCasilla

PRE: Tablero iniciado correctamente. La casilla debe contener una valor correcto (booleano)

POST: Comprueba el estado de la casilla[Fila][Columna].

Devuelve:

TRUE: Si la casilla esta encendida(1);

FALSE: Si la casilla esta apagada(0);

Complejidad: O(1)

ValorCasilla

PRE: Tablero iniciado correctamente.

POST: Comprueba el estado de la casilla[Fila][Columna].

Devuelve: Entero. La dimensión real del tablero de juego

Complejidad: O(1)

ObtenerDimensionReal

PRE: Tablero iniciado correctamente

POST: Modifica el tablero. Se activan las casillas que rodean a la casilla[Fila] [Columna] (arriba, abajo, izquierda, derecha)

Devuelve: Nada

Complejidad: O(1)

CambiarSegunPatron0

PRE: Tablero iniciado correctamente

POST: Modifica el tablero. Se activa la fila y la columna donde esta casilla[F][C]

Devuelve: Nada

Complejidad: O(1)

CambiarSegunPatron1

PRE: Tablero iniciado correctamente. Patron contiene un valor de 0 a 1 (a 2 con ampliación)

POST: Dependiendo del patrón cambia el estado de las casilla[fila][columna]:

-Si(0) las que están contiguas: arriba, abajo, derecha e

izquierda.

-Si(1) se cambia el estado en todas las casillas correspondientes con su fila y columna

-(Ampliacion) Si(2) se cambia el estado en todas las casillas correspondientes con diagonal a fila y columna

Devuelve: Entero. La dimensión real del tablero de juego.

Complejidad: O(1)

CambiarEstadoFichas

PRE: Tablero iniciado correctamente.

POST: Comprueba si el tablero esta totalmente apagado(0).

Devuelve:

TRUE: Si el tablero esta completamente apagado

FALSE: Si el tablero no esta completamente apagado

Complejidad: $O(n^2)$

ComprobarTodoApagado

2.2.3. Juego de pruebas

Se ha implementado un juego de pruebas por cada operación del TAD Tablero.

Las pruebas implementadas se corresponden con:

–Iniciar un tablero con todas sus casillas apagadas

```
void PIniciarTodoApagado(){
      Ttablero Tablero;
      Tablero.DimensionReal=5;
      cout << "El resultado es una matriz de Dimension 5 con todo a
0:" << endl << endl;
      IniciarTodoApagado(Tablero,Tablero.DimensionReal);
      EscribirTablero(Tablero,Tablero.DimensionReal);

}
```

–Iniciar un tablero de dimensión determinada y con un numero de casillas encendidas.

```
void PIniciarTableroAleatorio(){
      Ttablero Tablero;
      Tablero.DimensionReal=7;
      int Encendidas=5;
      cout << "El resultado es una matriz de Dimension 7 con 5
casillas a 1 puestas aleatoriamente:" << endl << endl;

IniciarTableroAleatorio(Tablero,Tablero.DimensionReal,Encendidas);
      EscribirTablero(Tablero,Tablero.DimensionReal);
}
```

–Iniciar un tablero con la configuración establecida de un fichero externo.

```
void PIniciarTableroDeFichero(){
        Ttablero Tablero;
        int          movs_restantes;
        int          comodin;
        int          patron;
        int          aleatorio;
      bool Carga=false;
      // Ponemos la matriz totalmente apagada.
      IniciarTodoApagado(Tablero,Tablero.DimensionReal);




      //Cargamos las variables

Carga=TEntornoCargarConfiguracion(Tablero.DimensionReal,patron,movs_re
stantes,comodin,aleatorio,Tablero.Matriz);

      cout << "El resultado es igual al fichero externo siempre que
aleatorio(5ª linea del fichero) sea 0:" << endl << endl;

      cout << "Dimension: " << Tablero.DimensionReal << endl;
      cout << "Patron: " << patron << endl;
      cout << "Movs Restantes: " << movs_restantes << endl;
      cout << "Comodin: "<< comodin << endl;
      cout << "Aleatorio: "<< aleatorio << endl;
      cout << "Matriz: " << endl;
      EscribirTablero(Tablero,Tablero.DimensionReal);
      // La matriz sera nula si la 5ª linea del fichero es 0.
```

```
	// Cuando es 0<aleatorio<MAXTAMANIO se genera una matriz
aleatoria pero que no es parte de la prueba de este modulo
	cout << "Si aleatorio != 0 se genera todo igual al fichero
excepto la matriz que es nula: " << endl;
}
```

–Cambiar el estado de una casilla del tablero

```
void PAccionCasilla(){
	Ttablero Tablero;
	int Dim=4;
	IniciarTodoApagado(Tablero,Dim);

	cout << "Se inicio tablero de dim 4 con todo apagado(0):" <<
endl;
	AccionCasilla(Tablero,3,2);
	cout << "Se acciono casilla[3][2]:" << endl;
	EscribirTablero(Tablero,Dim);
	cout << endl;
	AccionCasilla(Tablero,2,3);
	cout << "Se acciono casilla[2][3]:" << endl;
	EscribirTablero(Tablero,Dim);
	cout << endl;

	AccionCasilla(Tablero,2,3);
	cout << "Se acciono casilla[3][2]:" << endl;
	EscribirTablero(Tablero,Dim);
}
```

–Conocer el valor de una casilla del tablero

```
void PValorCasilla(){
	Ttablero Tablero;
	int Dim=4;
	IniciarTodoApagado(Tablero,Dim);
	cout << "Se inicio tablero de dim 4: "<< endl;
	AccionCasilla(Tablero,3,2);
	cout << "Se acciono casilla[3][2]:" << endl;
	EscribirTablero(Tablero,Dim);
	cout << endl;

	cout << "Verificando casilla[3][2]:" << endl;


	if(ValorCasilla(Tablero,3,2)==1)
		cout << "La casilla (3,2) esta encendida" << endl;
	else
		cout << "La casilla (3,2) esta apagada" << endl;

	cout << "Verificando casilla[1][2]:" << endl;

	if(ValorCasilla(Tablero,1,3)==1)
		cout << "La casilla (1,3) esta encendida" << endl;
	else
		cout << "La casilla (1,3) esta apagada" << endl;
}
```

–Conocer la dimensión del tablero

```
void PObtenerDimensionReal(){
	Ttablero Tablero;
	Tablero.DimensionReal=4;
	IniciarTodoApagado(Tablero,Tablero.DimensionReal);
	cout << "Verificando dimension real para un tablero con Dim=4:"
<<endl;
	if(ObtenerDimensionReal(Tablero)==4){
		cout << "El resultado es correcto: " <<
ObtenerDimensionReal(Tablero) << endl;
```

```
	}
	else{
		cout << "El resultado es erroneo:" <<
ObtenerDimensionReal(Tablero) << endl;
	}
}
```

–Cambiar casillas dado uno de los patrones

```
void PCambiarSegunPatron0(){
	Ttablero Tablero;
	Tablero.DimensionReal=5;
	int Fila;
	int Columna;

	IniciarTodoApagado(Tablero,Tablero.DimensionReal);

	//PATRON 0   Cruz desde la posición de la casilla
	Fila=1;
	Columna=4;
	cout << "Activada casilla[1][4] con patron 0:" << endl;
	CambiarSegunPatron0(Tablero,Fila,Columna);
	EscribirTablero(Tablero,Tablero.DimensionReal);
	cout << endl;

	Fila=4;
	Columna=4;
	cout << "Activada casilla[4][4] con patron 0:" << endl;
	IniciarTodoApagado(Tablero,Tablero.DimensionReal);
	CambiarSegunPatron0(Tablero,Fila,Columna);
	EscribirTablero(Tablero,Tablero.DimensionReal);
	cout << endl;

	Fila=3;
	Columna=3;
	cout << "Activada casilla[3][3] con patron 0:" << endl;
	IniciarTodoApagado(Tablero,Tablero.DimensionReal);
	CambiarSegunPatron0(Tablero,Fila,Columna);
	EscribirTablero(Tablero,Tablero.DimensionReal);
	cout << endl;
}

void PCambiarSegunPatron1(){
	Ttablero Tablero;
	Tablero.DimensionReal=5;
	int Fila;
	int Columna;

	IniciarTodoApagado(Tablero,Tablero.DimensionReal);

	//    PATRON 1   Se acciona toda la fila y columna respecto de la
posición de casilla

	Fila=0;
	Columna=3;
	cout << "Activada casilla[0][3] con patron 1:" << endl;
	CambiarSegunPatron1(Tablero,Fila,Columna);
	EscribirTablero(Tablero,Tablero.DimensionReal);
	cout << endl;


	Fila=3;
	Columna=4;
	IniciarTodoApagado(Tablero,Tablero.DimensionReal);
	cout << "Activada casilla[3][4] con patron 1:" << endl;
```

```
	CambiarSegunPatron1(Tablero,Fila,Columna);
	EscribirTablero(Tablero,Tablero.DimensionReal);
	cout << endl;

	Fila=0;
	Columna=0;
	IniciarTodoApagado(Tablero,Tablero.DimensionReal);
	cout << "Activada casilla[0][0] con patron 1:" << endl;
	CambiarSegunPatron1(Tablero,Fila,Columna);
	EscribirTablero(Tablero,Tablero.DimensionReal);
	cout << endl;
}

//(AMPLIACION)
void PCambiarSegunPatron2(){
	Ttablero Tablero;
	Tablero.DimensionReal=5;
	int Fila;
	int Columna;

	Fila=3;
	Columna=2;
	//    PATRON 2   Se accionan las diagonales respecto de la
posicion de casilla
	IniciarTodoApagado(Tablero,Tablero.DimensionReal);
	cout << "Activada casilla[3][2] con patron 2:" << endl;
	CambiarSegunPatron2(Tablero,Fila,Columna);
	EscribirTablero(Tablero,Tablero.DimensionReal);
	cout << endl;

	Fila=0;
	Columna=0;
	IniciarTodoApagado(Tablero,Tablero.DimensionReal);
	cout << "Activada casilla[0][0] con patron 2:" << endl;
	CambiarSegunPatron2(Tablero,Fila,Columna);
	EscribirTablero(Tablero,Tablero.DimensionReal);
	cout << endl;

	Fila=0;
	Columna=4;
	IniciarTodoApagado(Tablero,Tablero.DimensionReal);
	cout << "Activada casilla[0][4] con patron 2:" << endl;
	CambiarSegunPatron2(Tablero,Fila,Columna);
	EscribirTablero(Tablero,Tablero.DimensionReal);
	cout << endl;

	Fila=2;
	Columna=4;
	IniciarTodoApagado(Tablero,Tablero.DimensionReal);
	cout << "Activada casilla[2][4] con patron 2:" << endl;
	CambiarSegunPatron2(Tablero,Fila,Columna);
	EscribirTablero(Tablero,Tablero.DimensionReal);
	cout << endl;
}

void PCambiarEstadoFichas(){
	Ttablero Tablero;
	int Patron,Fila,Columna;
	Tablero.DimensionReal=5;

	cout << "Estado Inicial del tablero: apagado (todo a 0) con
Dim=5:" << endl;

	Patron=0;
```

```
	Fila=1;
	Columna=0;
	cout << "Estado para casilla[1][0] con patron 0" << endl;
	IniciarTodoApagado(Tablero,Tablero.DimensionReal);
	CambiarEstadoFichas(Tablero,Patron,Fila,Columna);
	EscribirTablero(Tablero,Tablero.DimensionReal);

	cout << endl;

	Patron=1;
	Fila=3;
	Columna=1;
	cout << "Estado para casilla[3][1] con patron 1" << endl;
	IniciarTodoApagado(Tablero,Tablero.DimensionReal);
	CambiarEstadoFichas(Tablero,Patron,Fila,Columna);
	EscribirTablero(Tablero,Tablero.DimensionReal);

	cout << endl;
}
```

–Comprobar si las casillas del tablero están apagadas.

```
void PComprobarTodoApagado(){
	Ttablero Tablero;
	Tablero.DimensionReal=5;
	IniciarTodoApagado(Tablero,Tablero.DimensionReal);
	cout << "Estado Inicial del tablero apagado (todo a 0) con
Dim=5:" << endl;
	cout << "Comprobando si todo apagado: " << endl;

	if(ComprobarTodoApagado(Tablero)==true){
		cout << "RESULTADO: El tablero esta con las luces apagadas"
<< endl;
		EscribirTablero(Tablero,Tablero.DimensionReal);
	}
	else{
		cout << "RESULTADO: Error, existen luces encendidas" <<
endl;
		EscribirTablero(Tablero,Tablero.DimensionReal);
	}

	cout << "Activando casilla[3][2]: " << endl;
	AccionCasilla(Tablero,3,2);
	cout << "Comprobando si todo apagado: " << endl;

	if(ComprobarTodoApagado(Tablero)==true){
		cout << "RESULTADO: El tablero esta con las luces apagadas"
<< endl;
		EscribirTablero(Tablero,Tablero.DimensionReal);}
	else{
		cout << "RESULTADO: Error, existen luces encendidas" <<
endl;
		EscribirTablero(Tablero,Tablero.DimensionReal);
	}
}
```

2.3. TAD Partida

Se encarga de gestionar la partida a traves del tablero y de actualizar el entorno grafico.

2.3.1. Descripción de tipo de datos Partida

Tpartida contiene:

- Tablero: a través del cual se gestiona el tablero y se actualiza el entorno
- Aleatorio: se corresponde las casillas encendidas con las que contara el tablero.
- Movs_Restantes: denota en cuantos movimientos debe hacerse la partida para ganar

-Comodín: (Ampliación), Su valor es recogido del fichero externo. Coincide con el numero de ayudas posibles para finalizar el juego correctamente.

Ejemplo: Si se cuenta con tres ayudas, el usuario podrá usarlas en cualquier momento (pulsando la tecla X) y se apagara o encenderá la bombilla seleccionada en ese momento. Según convenga o no al usuario le interesará utilizar ayudas para completar el juego.

-Patrón: La activación de las casillas depende de si es 0 o 1 ,(2 con ampliación).

2.3.2. Descripción de las operaciones

TAD Partida es:

PRE:

POST: Refresca la pantalla cuando se ha realizado alguna gestion sobre el tablero.

Devuelve: Nada.

Complejidad: $O(n^2)$

ActualizarPantalla

PRE:

POST: Inicia el tablero de juego (de Dimension x Dimension) con un numero de casillas encendidas(1) leidas de fichero o cargadas aleatoriamente.

Devuelve: Nada

Complejidad: O(1)

CargarConfiguracionExterna

PRE:

POST: Desde aqui se interacciona el juego con el usuario. El modulo gestiona los movimientos y cambios del patron, la actualizacion de la pantalla y si el usuario gana o no la partida .

Devuelve: Nada

Complejidad: O()

RealizarGestion

2.3.3. Juego de pruebas

–Implementadas: Cargar una partida desde el fichero externo.

```
void PCargarConfiguracionExterna(){

	Tpartida P;

	cout << "El resultado es igual que al fichero externo siempre 
que aleatorio(5ª linea del fichero) sea 0:" << endl << endl;
	CargarConfiguracionExterna(P);

	cout << "Dimension: " << P.Tablero.DimensionReal << endl;
	cout << "Patron: " << P.Patron << endl;
	cout << "Movs Restantes: " << P.Movs_restantes << endl;
	cout << "Comodin: "<< P.Comodin << endl;
	cout << "Aleatorio: "<< P.Aleatorio << endl;
	cout << "Matriz: " << endl;
	EscribirTablero(P.Tablero,P.Tablero.DimensionReal);

	// La matriz sera nula si la 5ª linea del fichero es 0.
	// Cuando es 0<aleatorio<MAXTAMANIO se genera una matriz 
aleatoria con los demás parámetros del fichero inalterados

	cout << "Si aleatorio != 0 se habrá generado una matriz 
aleatoria con los demás parámetros iguales al fichero: " << endl;

}
```

–No Implementadas: Refrescar la pantalla y Realizar gestiones

Los módulos no implementados requieren la inicialización del entorno. El proceso de pruebas no se ha realizado dado que visualmente debe ocurrir cualquier cambio realizado por el usuario en la partida desde su inicio.

2.4. Programa principal

El programa principal es muy simple. Se ha definido una variable del tipo Partida y a partir de esta se ha cargado la configuración del fichero externo para posteriormente empezar a jugar. Además incluye todas las llamadas a los módulos de prueba, realizados, desactivados inicialmente y cuya activación si se requiere debe hacerse manualmente.

2.5. Ampliaciones

El juego cuenta con dos modificaciones:

1)Un nuevo patrón que cambia el estado de las bombillas que están en las diagonales de la casilla[f][c] pulsada.
Para ello se implemento el siguiente modulo y se modifico otro que requiere su utilización en el TAD Tablero.

-Modulo creado

```
void CambiarSegunPatron2(Ttablero &Tablero,int &Fila,int &Columna){
	int i,j,Dim;
	i=Fila;
	j=Columna;

	Dim=ObtenerDimensionReal(Tablero);
	AccionCasilla(Tablero,i,j);
	while(j>=0){
		AccionCasilla(Tablero,i,j);
		i--;
		j--;
	}

	i=Fila;
	j=Columna;
	while(j<Dim){
		AccionCasilla(Tablero,i,j);
		i--;
		j++;
	}

	i=Fila;
	j=Columna;
	while(i<Dim){
		AccionCasilla(Tablero,i,j);
		i++;
		j--;
	}

	i=Fila;
	j=Columna;
	while(i<Dim){
		AccionCasilla(Tablero,i,j);
		i++;
		j++;
	}

}
```

-Modulo modificado

```
void CambiarEstadoFichas(Ttablero &Tablero,int Patron,int Fila, int Columna){
	switch(Patron){
		case 0:		CambiarSegunPatron0(Tablero,Fila,Columna);
	break;

		case 1:		CambiarSegunPatron1(Tablero,Fila,Columna);
	break;

		//Se añadio un nuevo caso correspondiente con el nuevo patron
		case 2:		CambiarSegunPatron2(Tablero,Fila,Columna);
	break;
	}
}
```

2)Dar funcionalidad a la Tecla X(TX) del modulo RealizarGestion. Su funcion es ayudar al usuario a terminar el juego, apagando todas las luces.

Por no poner todo el código del modulo se presenta el case TC modificado.

```
case TC:
	if(P.Patron==2)
		P.Patron=0;
	else
		P.Patron++;
		sprintf(msg, "Patrón actual: %d", P.Patron);
		TEntornoMostrarMensaje(Zona1, msg);
		sprintf(msg, "Tecla: C");
		TEntornoMostrarMensaje(Zona3, msg);
break;
```

3. Tareas

El desarrollo del proyecto completo ha demorado unas 14 horas. Durante este tiempo se citan las siguientes tareas realizadas:

–Realización del juego de pruebas del entorno (4 horas)

–Lectura de la documentación del proyecto (30 mins)

–Búsqueda en internet sobre las posibles versiones que se pedían (30 mins)

–Estructuración previa del proyecto (Análisis) (15 mins)

–Diseño del Tadtablero (45 mins)

–Batería de pruebas del Tadtablero (1 hora)

–Diseño del TadPartida (30 mins)

–Batería de pruebas del TadPartida (30 mins)

–Adecuación de los módulos y modificaciones (3 horas)

–Redacción de la documentación (3 horas)

–Ampliaciones (2 horas)

4. Conclusiones y principales problemas

La implementación del proyecto no me ha resultado costosa, los módulos mínimos que se requerían para su implementación son acordes con el nivel que presenta la asignatura y no me ha resultado difícil estructurar cada una de las partes del proyecto.

Sin embargo he encontrado dificultades para montarlo todo (Realizar el montaje del propio juego con la implicación de todos los TADs), pero finalmente con ayuda de alguna tutoría ha sido posible su entendimiento y finalización.

Por ultimo constatar que también hubo algunos problemas al implementar el modulo CambiarSegunPatron2,. El problema se resolvió haciendo por partes (cuadrantes) y dependiendo de la zona donde se accionara la casilla 4 bucles para accionar (o no) las diagonales de cada cuadrante.

Proyecto realizado por Carlos Campos Martín
1º Grado Ingeniería del Software