DOCUMENTACIÓN

Clase: Programación II	Práctica: Final (Proyecto Individual)
Alumno: Miguel Rodríguez Gallego	

Resumen: Título del juego y temática

Título: **Torre de Batalla**

Es una torre de batalla en la cual el usuario puede escoger un personaje con el cual pelear contra los diferentes oponentes que se encuentre en el camino.

Fase previa

Motivación del trabajo

La principal motivación para desarrollar este trabajo fue hacer un trabajo simple que recogiese todas las listas hechas hasta el momento para así ponerlas a prueba en un proyecto real.

Referencias

Solamente la de la primera propuesta proporcionada por el docente.

Análisis de la aplicación

Relaciones entre la implementación y los contenidos trabajados durante el curso

En el juego logré implementar los contenidos abordados en programación orientada a objetos que aparecieron en el 2º cuatrimestre:

- **Listas**:
  - Pilas: Para guardar las puntuaciones de los jugadores.
  - Colas: Para la creación de los enemigos a los que se enfrentará el jugador.

Interfaz de usuario

Cree un menú por consola con 3 opciones:

```
¿Qué quieres hacer?

0 - Jugar partida.
1 - Cargar partida guardada.
2 - Salir.
```

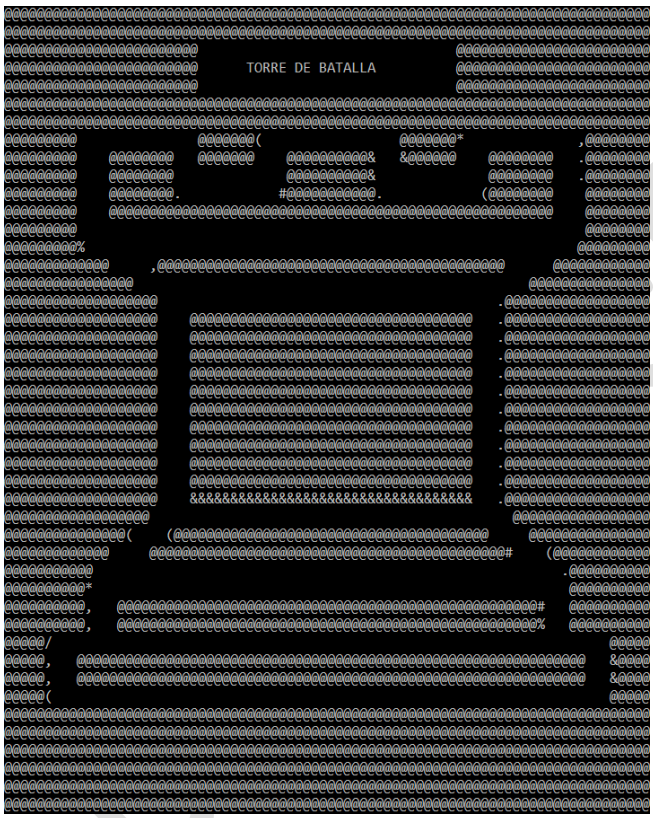
- Jugar ➔ Más menús:

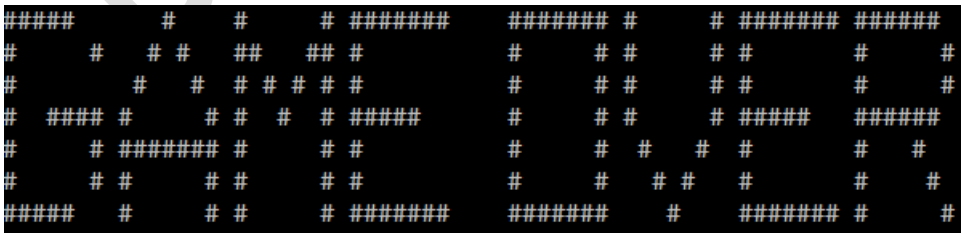
```
Con cuantos enemigos quieres luchar (de 1 a 10):
1
```

```
Te vas a enfrentar a Principe veloz

¿Qué quieres hacer?

0 - Atacar.
1 - Lanzar un proyectil mágico.
2 - Analizar enemigo.
3 - Descansar y recuperar energía.
4 - Guardar la partida actual.
5 - Salir del juego.
```

- Cargar Partida
- Salir

Además, usé convertidores de imágenes a arte ASCII para adornar el inicio del juego y por si el jugador pierde.





Guías

Guía de usuario

Como ya expliqué un poco antes, el proceso del usuario al entrar en el juego sería muy simple, ya que nada más ejecutar el programa aparecen las 3 principales opciones en un menú por consola.

```
¿Qué quieres hacer?

0 - Jugar partida.
1 - Cargar partida guardada.
2 - Salir.
```

- Jugar ➔ Empezar una partida nueva
  - Al entrar se le pide al usuario contra cuantos enemigos quiere jugar la siguiente torre

```
Con cuantos enemigos quieres luchar (de 1 a 10):
1
```

  - Al escoger el N.º, se le aparece el primero y sale el menú de acciones

```
Te vas a enfrentar a Principe veloz

¿Qué quieres hacer?

0 - Atacar.
1 - Lanzar un proyectil mágico.
2 - Analizar enemigo.
3 - Descansar y recuperar energía.
4 - Guardar la partida actual.
5 - Salir del juego.
```

Según la acción que escojamos saldrá un mensaje correspondiente a la acción y sus consecuencias en el rival o el jugador por pantalla, pudiéndose siempre observar los resultados de los daños en la vida de nuestro personaje en el apartado de analizar rival, donde también se muestran las estadísticas de nuestro personaje.

```
Jugador:  fuerza = 5 | poder magico = 7 | defensa = 2 | energia = 12 | barrera magica = 4 | vida = 18

Troll veloz: fuerza = 4 | poder magico = 8 | defensa = 1 | energia = 19 | barrera magica = 5 | vida = 17
```

  - Al derrotar a un enemigo saldrá el siguiente mensaje

```
Has derrotado a Gladiador veloz. Ahora a por el siguiente...
```

  - Si quedan más enemigos aparecerá de nuevo el menú de acciones contra el siguiente oponente.

```
¿Qué quieres hacer?

0 - Atacar.
1 - Lanzar un proyectil mágico.
2 - Analizar enemigo.
3 - Descansar y recuperar energía.
4 - Guardar la partida actual.
5 - Salir del juego.

2

Jugador:  fuerza = 5 | poder magico = 7 | defensa = 2 | energia = 12 | barrera magica = 4 | vida = 18

Troll veloz: fuerza = 4 | poder magico = 8 | defensa = 1 | energia = 19 | barrera magica = 5 | vida = 17

¿Qué quieres hacer?

0 - Atacar.
1 - Lanzar un proyectil mágico.
2 - Analizar enemigo.
3 - Descansar y recuperar energía.
4 - Guardar la partida actual.
5 - Salir del juego.
```

- Si no, pueden pasar 2 cosas:

  + Que el jugador haya ganado: Se muestran las anteriores puntuaciones recogidas en la pila y un mensaje.

```
Decision ataque enemigo : ataque
15/05/2021 18:19:13 - 804 puntos.
15/05/2021 18:18:49 - 659 puntos.
15/05/2021 21:07:19 - 590 puntos.
15/05/2021 21:16:54 - 590 puntos.
15/05/2021 20:42:42 - 524 puntos.
15/05/2021 20:48:54 - 521 puntos.
15/05/2021 20:31:25 - 520 puntos.
15/05/2021 21:01:28 - 516 puntos.


Has ganado la partida. Conseguiste derrotar a todos los enemigos. Pulse una tecla para salir.
```

  + Que el jugador haya perdido

```
######      #    #     # #######    ######## #     # ####### ######
#     #    # #   ##   ## #          #     # #     # #       #     #
#          #   #  # # # # #          #     # #     # #       #     #
#  #### #     # #  #  # #####      #     # #     # #####   ######
#     # ######## #     # #          #     #  #   #  #       #   #
#     # #     # #     # #          #     #   # #   #       #    #
######   #     # #     # #######    ########    #    ####### #     #

Has perdido la partida. Pulse una tecla para salir.
```

- Cargar Partida ➔ Cargar una partida guardada con anterioridad

  En tal caso saldría el menú de acciones de combate con los datos de la anterior partida

- Salir ➔ Salir del juego

  Simplemente te saca del juego.

Guía de posibles ampliaciones

Se podrían implementar más cosas como:

- Más personajes y su elección al inicio mediante una lista circular.

Implementación

Componentes

En el juego encontramos 9 componentes:

- Cola ➔ Creación de los enemigos
- Enemigo ➔ Estadísticas de los enemigos
- Jugador ➔ Estadísticas del personaje del jugador
- Nodo ➔ Creación del Nodo para trabajar con la pila y cola
- Pila ➔ Pila ordenada para la recogida de puntuaciones de los usuarios tras ganar la partida
- Program ➔ Menú principal
- Puntuacion ➔ Controlo la recogida de puntuación
- TorreDeBatalla ➔ Donde se desarrollan las funcionalidades básicas de la partida
- Utilidades ➔ Utilidad para hacer diferentes acciones dentro del programa

Código

Cola

```
// JuegoTorreDeBatalla
// Miguel Rodríguez Gallego
// 15/05/2021
// Uso una cola para la creación de los enemigos

using System;

namespace TorreDeBatalla
{
    class Cola
    {
        // Atributos
        private Nodo primero;
        private Nodo ultimo;

        // Constructor
        public Cola()
        {
            primero = null;
            ultimo = null;
        }

        // Insertar los nums en una lista enlazada
        public void Insertar(Enemigo enemigo)
        {
            // Variables auxiliares
            Nodo nuevoNodo;

            // Creamos nuestro nuevo nodo
```

```
            nuevoNodo = new Nodo();
            // Meto el dato que me pasan en una variable
            nuevoNodo.enemigo = enemigo;

            // Si la lista está vacía se iguala "primero" y "ultimo" al nuevo nodo
            if (Vacia())
            {
                nuevoNodo.siguiente = primero;
                primero = nuevoNodo;
                ultimo = nuevoNodo;
            }
            // Sino añadimos el "nuevoNodo" al final de la cola
            else
            {
                ultimo.siguiente = nuevoNodo;
                ultimo = nuevoNodo;
            }
        }

        /* Resumen método "Leer":
         * Hacer un nodo guia el cual cuando llegue a un número que no me interesa lo salte con la
variable
         * cabeza para eliminar (ya que en c#) los valores se eliminan automaticamente al saltarlos
         */
        public Nodo Leer()
        {
            // Creo un nuevo nodo para almacenar el primero
            Nodo n = new Nodo();
            n = primero;

            // Si la lista está vacía lo muestro por pantalla
            if (Vacia())
            {
                Console.WriteLine("Lista vacía");
            }
            // Si la lista solo tiene un elemento, lo elimino poniendo el "primero" y "ultimo" nodos
como nulos
            else if (primero == ultimo)
            {
                primero = null;
                ultimo = null;
            }
            // Si la lista tiene datos elimino el primero
            else
            {
                primero = primero.siguiente;
            }
            // Devuelvo el primer nodo de la cola
            return n;
        }

        // Método para comprobar si hay datos en la lista
        public bool Vacia()
        {
            if (primero == null)
            {
                return true;
            }
            else
            {
                return false;
            }
```

```
        }

        // Muestro por pantalla todos los datos
        public void Mostrar()
        {
            // Creo el nodo "indice"
            Nodo indice;
            // Abro un "if" para mostrar mediante el método "ListaVacia" un msg d lista vacía si
no hay datos/se borraron todos
            if (Vacia())
            {
                Console.WriteLine("Lista vacía");
            }
            // Creo un else para mostrar los datos si los hay
            else
            {
                // Creo la variable i, la cual ira subiendo segun el enemigo que toque
                int i = 1;
                indice = primero;
                // Escribo por pantalla los enemigos que tocaron y sus estadísticas
                Console.WriteLine("Enemigo: " + i + " - " + indice.enemigo.Nombre + " fuerza "+
indice.enemigo.Fuerza +
                    " poder magico " + indice.enemigo.Poder_magico + " defensa " +
indice.enemigo.Defensa + " energia " + indice.enemigo.Energia);

                // Con un bucle escribo por pantalla los enemigos que van después del 1º hasta que
no quede ninguno
                while (indice.siguiente != null)
                {
                    i++;
                    indice = indice.siguiente;
                    Console.WriteLine("Enemigo: " + i + " - " + indice.enemigo.Nombre + " fuerza
" + indice.enemigo.Fuerza +
                    " poder magico " + indice.enemigo.Poder_magico + " defensa " +
indice.enemigo.Defensa + " energia " + indice.enemigo.Energia);

                }
            }
        }
        public int Tamanio()
        {
            // Creo el nodo "indice"
            Nodo indice;
            // Si la lista está vacía devuelvo 0
            if (Vacia())
            {
               return 0;
            }
            // En el otro caso, calculo el tamaño de la cola
            else
            {
                // Creo la variable i para hacer de contador
                int i = 1;
                // Situo el índice en el 1er nodo
                indice = primero;

                // Vamos avanzando por la cola hasta el último nodo
                while (indice.siguiente != null)
                {
                    i++;
                    indice = indice.siguiente;
                }
```

```
                // Devuelvo el tamaño de la cola
                return i;
            }
        }

        // Este método pasa todos los enemigos de la cola a un array y lo devuelve.
        public Enemigo[] ColaToArray()
        {
            // Creo el nodo "indice" y un array en el que guardar los enemigos
            Nodo indice;
            Enemigo[] enemigos = new Enemigo[this.Tamanio()];

            // Si la lista está vacía devuelvo 0
            if (!Vacia())
            {
                // Creo la variable i para hacer de contador
                int i = 0;
                // Situo el índice en el 1er nodo
                indice = primero;

                // Vamos avanzando por la cola hasta el último nodo
                while (indice != null)
                {
                    enemigos[i] = indice.enemigo;
                    i++;
                    indice = indice.siguiente;
                }
            }
            return enemigos;
        }
    }
}
```

Enemigo

```
// JuegoTorreDeBatalla
// Miguel Rodríguez Gallego
// 15/05/2021
// Estadísticas de los enemigos

using System;
using System.Collections.Generic;
using System.Text;

namespace TorreDeBatalla
{
    // Declaro las variables
    class Enemigo
    {
        private string nombre = "";
        private int vida = 0;
        private int fuerza = 0;
        private int defensa = 0;
        private int poder_magico = 0;
        private int barrera_magica = 0;
        private int energia = 0;

        // Generador de tamaño de las estadísticas de cada enemigo según el nº total de enemigos
        public Enemigo()
        {
            this.Vida = Utilidades.NumAleatorio(15, 20);
            this.Fuerza = Utilidades.NumAleatorio(4, 6);
            this.Defensa = Utilidades.NumAleatorio(1, 3);
```

```
            this.Poder_magico = Utilidades.NumAleatorio(7, 9);
            this.Barrera_magica = Utilidades.NumAleatorio(4, 6);
            this.Energia = Utilidades.NumAleatorio(15, 20);
            this.Nombre = this.generarNombre();
        }

        // Métodos para cambiar los valores de las variables privadas (getters y setters)
        public string Nombre { get => nombre; set => nombre = value; }
        public int Vida { get => vida; set => vida = value; }
        public int Fuerza { get => fuerza; set => fuerza = value; }
        public int Defensa { get => defensa; set => defensa = value; }
        public int Poder_magico { get => poder_magico; set => poder_magico = value; }
        public int Barrera_magica { get => barrera_magica; set => barrera_magica = value; }
        public int Energia { get => energia; set => energia = value; }

        // Generador aleatorio de nombres para los enemigos
        public string generarNombre()
        {
            int numPersonaje = Utilidades.NumAleatorio(0, 9);
            int numAdjetivo = Utilidades.NumAleatorio(0, 9);

            string[] personajes = new string[] { "Brujo", "Guerrero", "Arquero", "Caballero",
"Gladiador", "Principe", "Troll", "Gigante", "Enano", "Mendigo" };
            string[] adjetivos = new string[] { "loco", "peludo", "piojoso", "fuerte", "rápido",
"veloz", "sucio", "cojo", "tuerto", "despiadado" };

            return personajes[numPersonaje] + " " + adjetivos[numAdjetivo];
        }
    }
}
```

Jugador

```
// JuegoTorreDeBatalla
// Miguel Rodríguez Gallego
// 15/05/2021
// Estadísticas del personaje del jugador

using System;
using System.Collections.Generic;
using System.Text;

namespace TorreDeBatalla
{
    class Jugador
    {
        // Declaro las variables
        private int vida = 0;
        private int fuerza = 0;
        private int defensa = 0;
        private int poder_magico = 0;
        private int barrera_magica = 0;
        private int energia = 0;

        // Constructor
        public Jugador()
        {
        }

        // Generador de tamaño de las estadísticas del jugador según el nº total de enemigos
        public Jugador(int numEnemigos)
        {
            this.Vida = Utilidades.NumAleatorio(15*numEnemigos, 20*numEnemigos);
```

```
            this.Fuerza = Utilidades.NumAleatorio(4, 6);
            this.Defensa = Utilidades.NumAleatorio(1, 3);
            this.Poder_magico = Utilidades.NumAleatorio(7, 9);
            this.Barrera_magica = Utilidades.NumAleatorio(4, 6);
            this.Energia = Utilidades.NumAleatorio(15*numEnemigos, 20*numEnemigos);
        }

        // Métodos para cambiar los valores de las variables privadas (getters y setters)
        public int Vida { get => vida; set => vida = value; }
        public int Fuerza { get => fuerza; set => fuerza = value; }
        public int Defensa { get => defensa; set => defensa = value; }
        public int Poder_magico { get => poder_magico; set => poder_magico = value; }
        public int Barrera_magica { get => barrera_magica; set => barrera_magica = value; }
        public int Energia { get => energia; set => energia = value; }
    }
}
```

Nodo

```
// JuegoTorreDeBatalla
// Miguel Rodríguez Gallego
// 15/05/2021
// Creación del Nodo

using System;
using System.Collections.Generic;
using System.Text;

namespace TorreDeBatalla
{
    // Creo el nodo
    class Nodo
    {
        public Enemigo enemigo;
        public Nodo siguiente;
    }
}
```

Pila

```
// JuegoTorreDeBatalla
// Miguel Rodríguez Gallego
// 15/05/2021
// Pila ordenada para la recogida de puntuaciones de los usuarios tras ganar la partida

using System;

namespace TorreDeBatalla
{
    class Pila
    {
        // Declaro la clase Nodo con sus respectivos atributos para trabajar con ella en la Pila
        class Nodo
        {
            public Puntuacion puntuacion;
            public Nodo siguiente;
        }
        private Nodo cabeza;

        // Constructor
        public Pila()
        {
            cabeza = null;
        }
```

```
        // Insertar los nums en una lista enlazada ordenada
        public void Insertar(Puntuacion puntos)
        {
            // Variables auxiliares
            Nodo nuevoNodo;
            Nodo anterior;

            // Creamos nuestro nuevo nodo
            nuevoNodo = new Nodo();
            // Meto el dato que me pasan en una variable
            nuevoNodo.puntuacion = puntos;
            anterior = null;

            // Si la lista está vacía o queremos insertar en la cabecera. (Suponemos q la función
"ListaVacía()" ya está implementada).
            if (ListaVacia() || cabeza.puntuacion.Puntos <= puntos.Puntos)
            {
                nuevoNodo.siguiente = cabeza;
                cabeza = nuevoNodo;
            }
            else
            {
                anterior = cabeza;
                // Buscamos el nodo d valor menor a v
                while (anterior.siguiente != null && anterior.siguiente.puntuacion.Puntos >=
puntos.Puntos)
                {
                    anterior = anterior.siguiente;
                }

                // Insertamos el nuevo nodo después del nodo anterior
                nuevoNodo.siguiente = anterior.siguiente;
                anterior.siguiente = nuevoNodo;
            }
        }
        // Método para comprobar si hay datos en la lista
        public bool ListaVacia()
        {
            if (cabeza == null)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
        // Muestro por pantalla todos los datos
        public void Mostrar()
        {
            // Creo el nodo "indice"
            Nodo indice;
            // Abro un "if" para mostrar mediante el método "ListaVacia" un msg d lista vacía si
no hay datos/se borraron todos
            if (ListaVacia())
            {
                Console.WriteLine("Lista vacía");
            }
            // Creo un else para mostrar los datos si los hay
            else
            {
```

```
                // Creo la variable i para hacer de contador al lado de los nºs que muestre por
pantalla a su izquierda
                int i = 1;
                indice = cabeza;
                // Muestro por pantalla la fecha y puntuación del primer jugador que se encuentra
en la Pila ordenada
                Console.WriteLine(indice.puntuacion.Fecha + " - " + indice.puntuacion.Puntos + "
puntos.");

                // Vamos avanzando por la lista hasta el último elemento
                while (indice.siguiente != null)
                {
                    i++;
                    indice = indice.siguiente;
                    // Muestro por pantalla la fecha y puntuación del resto de jugadores que se
encuentran en la Pila ordenada
                    Console.WriteLine(indice.puntuacion.Fecha + " - " + indice.puntuacion.Puntos
+ " puntos.");
                }
            }
        }
    }
}
```

Program

```
// JuegoTorreDeBatalla
// Miguel Rodríguez Gallego
// 15/05/2021
// Menú principal

using System;

namespace TorreDeBatalla
{
    class Program
    {
        enum MenuPartida
        {
            // Pide un valor para empezar a jugar
            jugar_partida,      // 0
            // pide un valor para cargar la partida guardada
            cargar_partida,     // 1
            // salir del juego
            salir,              // 2
        }

        static void Main(string[] args)
        {
            // Enseño la imagen ASCII al inicio del juego
            SplashScreen();

            int valor;
            int numEnemigos = 0;
            //Cola enemigos;
            MenuPartida opcion;
            TorreDeBatalla torreDeBatalla;

            while (true)
            {
                // Muestro el menú al usuario
                Console.WriteLine(" ");
                Console.WriteLine("¿Qué quieres hacer?");
```

```
                Console.WriteLine(" ");
                Console.WriteLine("0 - Jugar partida.");
                Console.WriteLine("1 - Cargar partida guardada.");
                Console.WriteLine("2 - Salir.");
                Console.WriteLine(" ");
                // Pido al usuario un valor para escoger una opción del menú
                try
                {
                    valor = Convert.ToInt32(Console.ReadLine());
                }
                catch (Exception)
                {
                    valor = 9;
                }

                opcion = (MenuPartida)valor;

                // Introduzco la opción del usuario en un switch con diferentes opciones
                switch (opcion)
                {
                    // Jugar al juego
                    case MenuPartida.jugar_partida:
                        // Pido el nº de enemigos y los creo
                        Console.WriteLine("Con cuantos enemigos quieres luchar (de 1 a 10): ");
                        numEnemigos = Convert.ToInt32(Console.ReadLine());
                        Console.WriteLine(" ");

                        if (numEnemigos < 1 || numEnemigos > 10)
                        {
                            Console.WriteLine("Error. Solo se puede elegir entre 1 y 10 enemigos.
Has elegido: " + numEnemigos);
                        } else
                        {
                            // Creo una partida
                            torreDeBatalla = new TorreDeBatalla(numEnemigos);
                            // Arranco la partida
                            torreDeBatalla.Jugar();
                        }
                        break;

                    // Cargar una partida anteriormente guardada
                    case MenuPartida.cargar_partida:
                        // Creo una partida a partir de los datos cargados
                        torreDeBatalla = new TorreDeBatalla();
                        // Cargo la partida guardada.
                        torreDeBatalla.CargarPartida();
                        // Arranco la partida
                        torreDeBatalla.Jugar();
                        break;

                    // Salir del juego
                    case MenuPartida.salir:
                        Environment.Exit(0);
                        break;

                    // Error si el usuario pulsa otra tecla del menú
                    default:
                        Console.WriteLine("Has pulsado una tecla errónea.");
                        break;
                }
            }
        }
```

```
        // Icono ASCII para mostrar al inicio del juego
        public static void SplashScreen()
        {

    Console.WriteLine("@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
    @@@@");

    Console.WriteLine("@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
    @@@@");
            Console.WriteLine("@@@@@@@@@@@@@@@@@@@@@@@@
    @@@@@@@@@@@@@@@@@@@@@@@@");
            Console.WriteLine("@@@@@@@@@@@@@@@@@@@@@@@@      TORRE DE BATALLA
    @@@@@@@@@@@@@@@@@@@@@@@@");
            Console.WriteLine("@@@@@@@@@@@@@@@@@@@@@@@@
    @@@@@@@@@@@@@@@@@@@@@@@@");

    Console.WriteLine("@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
    @@@@");

    Console.WriteLine("@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
    @@@@");
            Console.WriteLine("@@@@@@@@@               @@@@@@@(                 @@@@@@@*
    ,@@@@@@@@");
            Console.WriteLine("@@@@@@@@@    @@@@@@@@   @@@@@@@    @@@@@@@@@@&   &@@@@@@
    @@@@@@@@   .@@@@@@@@");
            Console.WriteLine("@@@@@@@@@    @@@@@@@@              @@@@@@@@@@&
    @@@@@@@@   .@@@@@@@@");
            Console.WriteLine("@@@@@@@@@    @@@@@@@@.            #@@@@@@@@@@@.
    (@@@@@@@@    @@@@@@@@");
            Console.WriteLine("@@@@@@@@@
    @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    @@@@@@@@");
            Console.WriteLine("@@@@@@@@@
    @@@@@@@@");
            Console.WriteLine("@@@@@@@@@%
    @@@@@@@@@");
            Console.WriteLine("@@@@@@@@@@@@@     ,@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
    @@@@@@@@@@@@");
            Console.WriteLine("@@@@@@@@@@@@@@@@
    @@@@@@@@@@@@@@@");
            Console.WriteLine("@@@@@@@@@@@@@@@@@@@
    .@@@@@@@@@@@@@@@@@@");
            Console.WriteLine("@@@@@@@@@@@@@@@@@@@    @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
    .@@@@@@@@@@@@@@@@@@");
            Console.WriteLine("@@@@@@@@@@@@@@@@@@@    @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
    .@@@@@@@@@@@@@@@@@@");
            Console.WriteLine("@@@@@@@@@@@@@@@@@@@    @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
    .@@@@@@@@@@@@@@@@@@");
            Console.WriteLine("@@@@@@@@@@@@@@@@@@@    @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
    .@@@@@@@@@@@@@@@@@@");
            Console.WriteLine("@@@@@@@@@@@@@@@@@@@    @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
    .@@@@@@@@@@@@@@@@@@");
            Console.WriteLine("@@@@@@@@@@@@@@@@@@@    @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
    .@@@@@@@@@@@@@@@@@@");
            Console.WriteLine("@@@@@@@@@@@@@@@@@@@    @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
    .@@@@@@@@@@@@@@@@@@");
            Console.WriteLine("@@@@@@@@@@@@@@@@@@@    @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
    .@@@@@@@@@@@@@@@@@@");
            Console.WriteLine("@@@@@@@@@@@@@@@@@@@    @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
    .@@@@@@@@@@@@@@@@@@");
            Console.WriteLine("@@@@@@@@@@@@@@@@@@@    @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
    .@@@@@@@@@@@@@@@@@@");
```

```
            Console.WriteLine("@@@@@@@@@@@@@@@@@@@    &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
.@@@@@@@@@@@@@@@@@@");
            Console.WriteLine("@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@");
            Console.WriteLine("@@@@@@@@@@@@@@@(    (@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@");
            Console.WriteLine("@@@@@@@@@@@@@     @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@#
(@@@@@@@@@@@@");
            Console.WriteLine("@@@@@@@@@@@
.@@@@@@@@@@");
            Console.WriteLine("@@@@@@@@@@*
@@@@@@@@@@");
            Console.WriteLine("@@@@@@@@@@,
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@#   @@@@@@@@@@");
            Console.WriteLine("@@@@@@@@@@,
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@%   @@@@@@@@@@");
            Console.WriteLine("@@@@@/
@@@@@");
            Console.WriteLine("@@@@@,
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@   &@@@@");
            Console.WriteLine("@@@@@,
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@   &@@@@");
            Console.WriteLine("@@@@@(
@@@@@");

Console.WriteLine("@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@");

Console.WriteLine("@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@");

Console.WriteLine("@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@");

Console.WriteLine("@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@");

Console.WriteLine("@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@");

Console.WriteLine("@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@");
        }
    }
}
```

Puntuación

```
// JuegoTorreDeBatalla
// Miguel Rodríguez Gallego
// 15/05/2021
// Controlo la recogida de puntuación

using System;
using System.Collections.Generic;
using System.Text;

namespace TorreDeBatalla
{
    class Puntuacion
    {
        // Declaro la variable de la fecha y de los puntos
        private DateTime fecha;
```

```
        private int puntos;

        // Constructor
        public Puntuacion()
        {
            this.fecha = DateTime.Now;
        }

        // Métodos para cambiar los valores de las variables privadas (getters y setters)
        public DateTime Fecha { get => fecha; set => fecha = value; }
        public int Puntos { get => puntos; set => puntos = value; }
    }
}
```

TorreDeBatalla

```
// JuegoTorreDeBatalla
// Miguel Rodríguez Gallego
// 15/05/2021
// Donde se desarrollan las funcionalidades básicas de la partida


using System;
using System.IO;


namespace TorreDeBatalla
{
    class TorreDeBatalla
    {
        // Clases que voy a utilizar
        Jugador jugador;
        Cola enemigos;
        Enemigo enemigoActual;
        Puntuacion puntos;
        Pila puntuacionesMasAltas;


        enum MenuJugar
        {
            // atacar a un enemigo
            atacar,                 // 0
            // lanzar un proyectil mágico
            proyectil_magico,       // 1
            // analizar estadísticas del enemigo actual
            analizar,               // 2
            // no atacar a cambio de recuperar energía
            descansar,              // 3
            // guardar la partida actual
            guardar_partida,        // 4
            // salir del juego
```

```
    salir,                  // 5
}

// Constructor por defecto
public TorreDeBatalla()
{
    jugador = new Jugador();
    enemigos = new Cola();
    puntos = new Puntuacion();
    puntuacionesMasAltas = new Pila();
}

// Constructor para una partida nueva
public TorreDeBatalla(int numEnemigos)
{
    jugador = new Jugador(numEnemigos);
    enemigos = new Cola();
    this.GenerarEnemigos(numEnemigos);
    puntos = new Puntuacion();
    puntuacionesMasAltas = new Pila();
}

// Generador de enemigos segun el Nº de enemigos escogido
public void GenerarEnemigos(int numEnemigos)
{
    for (int i = 0; i < numEnemigos; i++)
    {
        Enemigo enemigo = new Enemigo();
        enemigos.Insertar(enemigo);
    }
}

// Menu de opciones durante la partida frente a un enemigo
public void Jugar()
{
    int valor;
    MenuJugar opcion;
    enemigoActual = enemigos.Leer().enemigo;

    Console.WriteLine("Te vas a enfrentar a " + enemigoActual.Nombre);
```

```
while (true)
{
    // Muestro el menú al usuario
    Console.WriteLine(" ");
    Console.WriteLine("¿Qué quieres hacer?");
    Console.WriteLine(" ");
    Console.WriteLine("0 - Atacar.");
    Console.WriteLine("1 - Lanzar un proyectil mágico.");
    Console.WriteLine("2 - Analizar enemigo.");
    Console.WriteLine("3 - Descansar y recuperar energía.");
    Console.WriteLine("4 - Guardar la partida actual.");
    Console.WriteLine("5 - Salir del juego.");
    Console.WriteLine(" ");
    // Pido al usuario un valor para escoger una opción del menú
    try
    {
        valor = Convert.ToInt32(Console.ReadLine());
    }
    catch (Exception)
    {
        valor = 9;
    }

    opcion = (MenuJugar)valor;

    // Introduzco la opción del usuario en un switch con diferentes opciones
    switch (opcion)
    {
        // Ataque físico
        case MenuJugar.atacar:
            Ataque();
            ContraAtaque();
            RegeneraEnergia();
            break;

        // Ataque mágico
        case MenuJugar.proyectil_magico:
            ProyectilMagico();
            ContraAtaque();
```

```
                        RegeneraEnergia();
                        break;


                    // Analizar al rival
                                case MenuJugar.analizar:
                        Console.WriteLine(" ");
                        Console.WriteLine("Jugador: " + " fuerza = " + this.jugador.Fuerza + " |
poder magico = " +
                            this.jugador.Poder_magico + " | defensa = " + this.jugador.Defensa +
" | energia = " + this.jugador.Energia +
                            " | barrera magica = " + this.jugador.Barrera_magica + " | vida = " +
this.jugador.Vida);


                        Console.WriteLine(" ");


                        Console.WriteLine(enemigoActual.Nombre + ": fuerza = " +
enemigoActual.Fuerza + " | poder magico = " +
                            enemigoActual.Poder_magico + " | defensa = " + enemigoActual.Defensa
+ " | energia = " + enemigoActual.Energia +
                            " | barrera magica = " + enemigoActual.Barrera_magica + " | vida = "
+ enemigoActual.Vida);


                        //enemigos.Mostrar();
                        break;


                    // Descansar para recuperar energia
                    case MenuJugar.descansar:
                        RegeneraEnergia();
                        ContraAtaque();
                        break;


                    // Guardar la partida
                    case MenuJugar.guardar_partida:
                        GuardarPartida();
                        break;


                    // Salir de la partida
                    case MenuJugar.salir:
                        Environment.Exit(0);
                        break;


                    // Por si el usuario introduce un valor que no sea uno de los anteriores del
menú
```

```
                    default:
                        Console.WriteLine("Has pulsado una tecla errónea.");
                        break;
                }

                if (jugador.Vida <= 0)
                {
                    SplashScreen2();
                    Console.WriteLine(" ");
                    Console.WriteLine("Has perdido la partida. Pulse una tecla para salir.");
                    Console.ReadLine();
                    Environment.Exit(0);
                }
                if (enemigoActual.Vida <= 0)
                {
                    if (enemigos.Vacia())
                    {
                        puntos.Puntos += 500;
                        CargarPuntuaciones();
                        puntuacionesMasAltas.Insertar(puntos);
                        puntuacionesMasAltas.Mostrar();
                        GuardarPuntuaciones();
                        Console.WriteLine(" ");
                        Console.WriteLine(" ");
                        Console.WriteLine("Has ganado la partida. Conseguiste derrotar a todos los
enemigos. Pulse una tecla para salir.");
                        Console.ReadLine();
                        Environment.Exit(0);
                    } else
                    {
                        puntos.Puntos += 50;
                        Console.WriteLine(" ");
                        Console.WriteLine(" ");
                        Console.WriteLine("Has derrotado a " + enemigoActual.Nombre + ". Ahora a
por el siguiente...");
                        enemigoActual = enemigos.Leer().enemigo;
                    }
                }
            }
        }
```

```
// Ataque físico del jugador
public void Ataque()
{
    if (jugador.Energia > 0)
    {
        if (jugador.Fuerza > enemigoActual.Defensa)
        {
            enemigoActual.Vida -= jugador.Fuerza;
        }
        jugador.Energia -= jugador.Fuerza;
        puntos.Puntos += jugador.Fuerza;
        if (jugador.Energia < 0)
        {
            jugador.Energia = 0;
        }
    } else
    {
        Console.WriteLine("Ataque fallido. No tienes energía.");
    }
}

// Ataque Proyectil Mágico del jugador
public void ProyectilMagico()
{
    if (jugador.Energia > 0)
    {
        if (jugador.Poder_magico > enemigoActual.Barrera_magica)
        {
            enemigoActual.Vida -= jugador.Poder_magico;
        }
        jugador.Energia -= jugador.Poder_magico;
        puntos.Puntos += jugador.Poder_magico;
        if (jugador.Energia < 0)
        {
            jugador.Energia = 0;
        }
    }
    else
    {
        Console.WriteLine("Ataque fallido. No tienes energía.");
```

```
    }
}
// Regenerar Energia del jugador
public void RegeneraEnergia()
{
    jugador.Energia += Utilidades.NumAleatorio(1,2);
}
// Contraataque del enemigo tras el ataque del jugador
public void ContraAtaque()
{
    // El contraataque necesita un nº aleatorio entre 1 y 4 para decidir la opción a ejecutar.
    int decision = Utilidades.NumAleatorio(1, 3);

    // Se realiza la acción en función de la decision obtenida.
    switch (decision)
    {
        case 1:
            AtaqueEnemigo();
            RegeneraEnergiaEnemigo();
            Console.WriteLine(" ");
            Console.WriteLine("Decision ataque enemigo : ataque");
            break;

        case 2:
            ProyectilMagicoEnemigo();
            RegeneraEnergiaEnemigo();
            Console.WriteLine(" ");
            Console.WriteLine("Decision ataque enemigo : proyectil mágico");
            break;

        case 3:
            RegeneraEnergiaEnemigo();
            Console.WriteLine(" ");
            Console.WriteLine("Decision ataque enemigo : descansar");
            break;
    }
}

// Ataque del enemigo
public void AtaqueEnemigo()
```

```
{
    if (enemigoActual.Energia > 0)
    {
        if (enemigoActual.Fuerza > jugador.Defensa)
        {
            jugador.Vida -= enemigoActual.Fuerza;
        }
        enemigoActual.Energia -= enemigoActual.Fuerza;
        if (enemigoActual.Energia < 0)
        {
            enemigoActual.Energia = 0;
        }
    }
}

// Ataque Mágico del Enemigo
public void ProyectilMagicoEnemigo()
{
    if (enemigoActual.Energia > 0)
    {
        if (enemigoActual.Poder_magico > jugador.Barrera_magica)
        {
            jugador.Vida -= enemigoActual.Poder_magico;
        }
        enemigoActual.Energia -= enemigoActual.Poder_magico;
        if (enemigoActual.Energia < 0)
        {
            enemigoActual.Energia = 0;
        }
    }
}

// Regenerar Energia del Enemigo
public void RegeneraEnergiaEnemigo()
{
    enemigoActual.Energia += Utilidades.NumAleatorio(1, 2);
}

// Guardar progreso de la Partida
public void GuardarPartida()
```

```
{
    // Paso la ruta del fichero al lector de texto "StreamReader"
    StreamWriter sw = File.CreateText("partida.txt");

    sw.WriteLine(jugador.Vida);
    sw.WriteLine(jugador.Fuerza);
    sw.WriteLine(jugador.Defensa);
    sw.WriteLine(jugador.Poder_magico);
    sw.WriteLine(jugador.Barrera_magica);
    sw.WriteLine(jugador.Energia);

    sw.WriteLine(enemigoActual.Nombre);
    sw.WriteLine(enemigoActual.Vida);
    sw.WriteLine(enemigoActual.Fuerza);
    sw.WriteLine(enemigoActual.Defensa);
    sw.WriteLine(enemigoActual.Poder_magico);
    sw.WriteLine(enemigoActual.Barrera_magica);
    sw.WriteLine(enemigoActual.Energia);

    // Paso los enemigos de la cola a un array.
    Enemigo[] enemigosArray = enemigos.ColaToArray();

    // Se recorre el array de enemigos asignando cada enemigo a la variable "c"
    foreach (Enemigo en in enemigosArray)
    {
        sw.WriteLine(en.Nombre);
        sw.WriteLine(en.Vida);
        sw.WriteLine(en.Fuerza);
        sw.WriteLine(en.Defensa);
        sw.WriteLine(en.Poder_magico);
        sw.WriteLine(en.Barrera_magica);
        sw.WriteLine(en.Energia);
    }

    //Cierro el fichero
    sw.Close();
}

// Cargar progreso de la partida Guardada
public void CargarPartida()
```

```
{
    // Declaro variables
    string linea;

    //Paso la ruta del fichero al lector de texto "StreamReader"
    StreamReader sr = File.OpenText("partida.txt");
    //Leo la primera linea de fichero para comprobar que hay texto
    linea = sr.ReadLine();
    jugador.Vida = Convert.ToInt32(linea);
    linea = sr.ReadLine();
    jugador.Fuerza = Convert.ToInt32(linea);
    linea = sr.ReadLine();
    jugador.Defensa = Convert.ToInt32(linea);
    linea = sr.ReadLine();
    jugador.Poder_magico = Convert.ToInt32(linea);
    linea = sr.ReadLine();
    jugador.Barrera_magica = Convert.ToInt32(linea);
    linea = sr.ReadLine();
    jugador.Energia = Convert.ToInt32(linea);
    linea = sr.ReadLine();

    //Continuo leyendo hasta que terminen los datos del fichero
    while (linea != null)
    {
        Enemigo enemigo = new Enemigo();

        // Traslado el valor que recojo con la variable "linea" al enemigo
        enemigo.Nombre = linea;
        linea = sr.ReadLine();
        enemigo.Vida = Convert.ToInt32(linea);
        linea = sr.ReadLine();
        enemigo.Fuerza = Convert.ToInt32(linea);
        linea = sr.ReadLine();
        enemigo.Defensa = Convert.ToInt32(linea);
        linea = sr.ReadLine();
        enemigo.Poder_magico = Convert.ToInt32(linea);
        linea = sr.ReadLine();
        enemigo.Barrera_magica = Convert.ToInt32(linea);
        linea = sr.ReadLine();
        enemigo.Energia = Convert.ToInt32(linea);
```

```
            linea = sr.ReadLine();

            // Una vez leído un enemigo lo añado a la cola
            enemigos.Insertar(enemigo);
        }
        //Cierro el fichero
        sr.Close();

    }

    // Guardar progreso de la Partida
    public void GuardarPuntuaciones()
    {
        // Paso la ruta del fichero al lector de texto "StreamReader"
        StreamWriter sw = File.AppendText("puntuaciones.txt");

        sw.WriteLine(puntos.Fecha);
        sw.WriteLine(puntos.Puntos);

        //Cierro el fichero
        sw.Close();
    }

    // Cargar progreso de la partida Guardada
    public void CargarPuntuaciones()
    {
        // Declaro variables
        string linea;

        try
        {
            //Paso la ruta del fichero al lector de texto "StreamReader"
            StreamReader sr = File.OpenText("puntuaciones.txt");
            //Leo la primera linea de fichero para comprobar que hay texto
            linea = sr.ReadLine();

            //Continuo leyendo hasta que terminen los datos del fichero
            while (linea != null)
            {
                Puntuacion p = new Puntuacion();
```

```
                    // Traslado el valor que recojo con la variable "linea" al enemigo
                    p.Fecha = Convert.ToDateTime(linea);
                    linea = sr.ReadLine();
                    p.Puntos = Convert.ToInt32(linea);
                    linea = sr.ReadLine();

                    // Una vez leído un enemigo lo añado a la cola
                    puntuacionesMasAltas.Insertar(p);
                }
                //Cierro el fichero
                sr.Close();
            }
            catch (FileNotFoundException)
            {
            }
        }

        // Mensaje ASCII para cuando el jugador pierda
        public static void SplashScreen2()
        {
            Console.WriteLine(" ");
            Console.WriteLine("#####     #    #     # #######    ####### #     # ####### ######
");
            Console.WriteLine("#    #   # #   ##   ## #          #     # #     # #       #     #
");
            Console.WriteLine("#     #   #  # # # # #          #     # #     # #       #     #
");
            Console.WriteLine("#  #### #     # #  #  # #####      #     # #     # #####   ######
");
            Console.WriteLine("#     # ####### #     # #          #     #  #   #  #       #   #
");
            Console.WriteLine("#     # #     # #     # #          #     #   # #   #       #    #
");
            Console.WriteLine("#####   #      # #      # #######    #######    #    ####### #
# ");
        }
    }
}
```

Utilidades

```
// JuegoTorreDeBatalla
```

```
// Miguel Rodríguez Gallego
// 15/05/2021
// Utilidad para hacer diferentes acciones dentro del programa

using System;
using System.Collections.Generic;
using System.Text;

namespace TorreDeBatalla
{
    class Utilidades
    {
        // Utilidad para generar números aleatorios en un rango de valores
        public static int NumAleatorio(int inicio, int fin)
        {
            Random rnd = new Random();
            int randNum = rnd.Next(inicio, fin);
            return randNum;
        }
    }
}
```