CONSOLE INVADERS

(6.2.2) Tras leer la descripción de Space Invaders que vimos en el apartado anterior, crea una clase Juego, que sólo contenga un método Lanzar, void, sin parámetros, que escriba en pantalla "Bienvenido a Console Invaders. Pulse Intro para salir" y se parará hasta que el usuario pulse Intro. Prepara también un Main (en la misma clase), que cree un objeto de la clase Juego y lo lance.

```
using System;

namespace ConsoleInvaders
{
   class MainClass
   {
      public static void Main()
      {
         Juego juego = new Juego();
         juego.Lanzar();
      }
   }
}
using System;

namespace ConsoleInvaders
{
   public class Juego
   {
      public void Lanzar ()
      {
         Console.Write("Bienvenido a Console Invaders. Pulse Intro para salir.");
         Console.ReadLine();
      }
   }
}
```

(6.3.4) Amplía el esqueleto del ConsoleInvaders (ejercicio 6.2.2): crea un proyecto para Visual Studio o SharpDevelop. Además de la clase "Juego", crea una clase "Bienvenida" y una clase "Partida". El método "Lanzar" de la clase Juego, ya no escribirá nada en pantalla, sino que creará un objeto de la clase "Bienvenida" y lo lanzará y luego un objeto de la clase "Partida" y lo lanzará. El método Lanzar de la clase Bienvenida escribirá en pantalla "Bienvenido a Console Invaders. Pulse Intro para jugar". El método Lanzar de la clase Partida escribirá en pantalla "Ésta sería la pantalla de juego. Pulse Intro para salir" y se parará hasta que el usuario pulse Intro.

```
using System;

namespace ConsoleInvaders
{
   class MainClass
   {
      public static void Main()
      {
         Juego juego = new Juego();
         juego.Lanzar();
      }
```

```
    }
}
using System;

namespace ConsoleInvaders
{
    public class Juego
    {
        public void Lanzar ()
        {
            Bienvenida bienvenida = new Bienvenida();
            Partida partida = new Partida();
            bienvenida.Lanzar();
            partida.Lanzar();
        }
    }
}
using System;

namespace ConsoleInvaders
{
    public class Bienvenida
    {
        public void Lanzar()
        {
            Console.Write("Bienvenido a Console Invaders. Pulse Intro para jugar.");
            Console.ReadLine();
        }
    }
}
using System;

namespace ConsoleInvaders
{
    public class Partida
    {
        public void Lanzar ()
        {
            Console.Clear();
            Console.Write("Ésta sería la pantalla de juego. Pulsa Intro para salir.");
            Console.ReadLine();
        }
    }
}
```

(6.3.5) Amplía el esqueleto del ConsoleInvaders (ejercicio 6.3.4): El método Lanzar de la clase Bienvenida escribirá en pantalla "Bienvenido a Console Invaders. Pulse Intro para jugar o ESC para salir". Puedes comprobar si se pulsa ESC con "ConsoleKeyInfo tecla = Console.ReadKey(); if (tecla.Key == ConsoleKey.Escape) salir = true;". El código de la tecla Intro es " ConsoleKey.Enter". También puedes usar "Console.Clear();" si quieres borrar la pantalla. Añade un método "GetSalir" a la clase Bienvenida, que devuelva "true" si el usuario ha escogido Salir o "false" si ha elegido Jugar. El método Lanzar de la clase Juego repetirá la secuencia Bienvenida-Partida hasta que el usuario escoja Salir.

```
using System;

namespace Tercero
```

```
{
    class Program
    {
        static void Main()
        {
            Juego juego = new Juego();
            juego.Lanzar();
        }
    }
}
using System;

namespace Tercero
{
    class Juego
    {
        public void Lanzar()
        {
            Bienvenida bienvenida = new Bienvenida();
            Partida partida = new Partida();
            do
            {
                bienvenida.Lanzar();
                if (!bienvenida.GetSalir())
                    partida.Lanzar();
            } while (!bienvenida.GetSalir());
        }
    }
}
using System;

namespace Tercero
{
    class Bienvenida
    {
        ConsoleKeyInfo tecla;

        //public bool Salir { get; set; }

        private bool salir;

        public bool GetSalir()
        {
            return salir;
        }

        public void Lanzar()
        {
            Console.Clear();
            Console.Write("Bienvenido a Console Invaders. " +
                "Pulsa Intro para jugar o ESC para salir.");
            tecla = Console.ReadKey();
            if (tecla.Key == ConsoleKey.Escape)
            {
                salir = true;
            }
        }
    }
}
using System;

namespace Tercero
```

```
{
    class Partida
    {
        public void Lanzar()
        {
            Console.Clear();
            Console.Write("Ésta sería la pantalla de juego. " +
                "Pulsa Intro para salir.");
            Console.ReadLine();
        }
    }
}
```

(6.3.6) Amplía el esqueleto del ConsoleInvaders (ejercicio 6.3.5): Crea una clase Nave, con atributos "x" e "y" (números enteros, "x" de 0 a 1023 e "y" entre 0 y 767, pensando en una pantalla de 1024x768), e imagen (un string formado por dos caracteres, como "/\"). También tendrá un método MoverA(nuevaX, nuevaY) que lo mueva a una nueva posición, y un método Dibujar, que muestre esa imagen en pantalla (como esta versión es para consola, la X tendrá que rebajarse para que tenga un valor entre 0 y 79, y la Y entre 0 y 24). Puedes usar Console.SetCursorPosition(x,y) para situarte en unas coordenadas de pantalla. Crea también clase Enemigo, con los mismos atributos. Su imagen podría ser "][". El método Lanzar de la clase Partida creará una nave en las coordenadas (500, 600) y la dibujará, creará un enemigo en las coordenadas (100, 80) y lo dibujará, y finalmente esperará a que el usuario pulse Intro para terminar la falsa sesión de juego.

```
using System;

namespace Segundo
{
    class Program
    {
        static void Main()
        {
            Juego juego = new Juego();
            juego.Lanzar();
        }
    }
}
```

```
using System;

namespace Segundo
{
    class Juego
    {
        public void Lanzar()
        {
            Bienvenida bienvenida = new Bienvenida();
            Partida partida = new Partida();
            do
            {
                bienvenida.Lanzar();
                if (!bienvenida.Salir)
                    partida.Lanzar();
            } while (!bienvenida.Salir);
        }
    }
}
```

```
using System;
```

```
namespace Segundo
{
    class Bienvenida
    {
        ConsoleKeyInfo tecla;
        public bool Salir { get; set; }

        public void Lanzar()
        {
            Console.Clear();
            Console.Write("Bienvenido a Console Invaders. " +
                "Pulsa Intro para jugar o ESC para salir.");
            tecla = Console.ReadKey();
            if (tecla.Key == ConsoleKey.Escape)
            {
                Salir = true;
            }
        }
    }
}
using System;

namespace Segundo
{
    class Partida
    {
        public void Lanzar()
        {
            Nave nave = new Nave();
            Enemigo enemigo = new Enemigo();
            Console.Clear();
            nave.Dibujar();
            enemigo.Dibujar();
            Console.ReadLine();
        }
    }
}
using System;

namespace Segundo
{
    class Nave
    {
        private int x=40, y=20;
        private string imagen="/\\";

        public void MoverA(int nuevaX, int nuevaY)
        {

        }

        public void Dibujar()
        {
            Console.SetCursorPosition(x, y);
            Console.Write(imagen);
            Console.CursorVisible = false;
        }
    }
}
using System;

namespace Segundo
```

```
{
    class Enemigo
    {
        private int x = 40, y = 10;
        private string imagen = "][";

        public void MoverA(int nuevaX, int nuevaY)
        {

        }

        public void Dibujar()
        {
            Console.SetCursorPosition(x, y);
            Console.Write(imagen);
            Console.CursorVisible = false;
        }
    }
}
```

(6.4.4) Amplía el esqueleto del ConsoleInvaders (ejercicio 6.3.6): Crea una clase "Sprite", de la que heredarán "Nave" y "Enemigo". La nueva clase contendrá todos los atributos y métodos que son comunes a las antiguas (todos los existentes, por ahora).

(6.5.4) Amplía el esqueleto del ConsoleInvaders (ejercicio 6.4.4): Amplía la clase Nave con un método "MoverDerecha", que aumente su X en 10 unidades, y un "MoverIzquierda", que disminuya su X en 10 unidades. Necesitarás hacer que esos atributos sean "protected". El método Lanzar de la clase Partida no esperará hasta el usuario pulse Intro sin hacer nada, sino que ahora usará un do-while que compruebe si pulsa ESC (para salir) o flecha izquierda o flecha derecha (para mover la nave: sus códigos son ConsoleKey.LeftArrow y ConsoleKey. RightArrow). Si se pulsan las flechas, la nave se moverá a un lado o a otro (con los métodos que acabas de crear). Al principio de cada pasada del do-while se borrará la pantalla ("Console.Clear();").

```
using System;

namespace ConsoleInvaders
{
    class Program
    {
        static void Main()
        {
            Juego juego = new Juego();
            juego.Lanzar();
        }
    }
}
```

```
using System;

namespace ConsoleInvaders
{
    class Juego
    {
        public void Lanzar()
        {
```

```
            Bienvenida bienvenida = new Bienvenida();
            Partida partida = new Partida();

            do
            {
                bienvenida.Lanzar();
                if (!bienvenida.Salir)
                    partida.Lanzar();
            } while (!bienvenida.Salir);
        }
    }
}
```

```
using System;

namespace ConsoleInvaders
{
    class Bienvenida
    {
        ConsoleKeyInfo tecla;

        public bool Salir { get; private set; }

        public void Lanzar()
        {
            Console.Clear();
            Console.Write("Bienvenido a Console Invaders. " +
                "Pulse Intro para jugar o ESC para salir.");
            tecla = Console.ReadKey();
            if (tecla.Key == ConsoleKey.Escape)
                Salir = true;
        }
    }
}
```

```
using System;

namespace ConsoleInvaders
{
    class Partida
    {
        ConsoleKeyInfo tecla;
        Nave nave = new Nave();
        Enemigo enemigo = new Enemigo();

        public void Lanzar()
        {
            Console.Clear();
            nave.Dibujar();
            enemigo.Dibujar();
            do
            {
                tecla = Console.ReadKey();
                if (tecla.Key == ConsoleKey.RightArrow)
                    nave.MoverDerecha();
                if (tecla.Key == ConsoleKey.LeftArrow)
                    nave.MoverIzquierda();
                Console.Clear();
                nave.Dibujar();
                enemigo.Dibujar();
            } while (tecla.Key != ConsoleKey.Escape);
```

```
        }
    }
}

using System;

namespace ConsoleInvaders
{
    class Sprite
    {
        protected int x, y;
        protected string imagen;

        public void MoverA(int nuevaX, int nuevaY)
        {

        }

        public void Dibujar()
        {
            Console.SetCursorPosition(x, y);
            Console.Write(imagen);
            Console.CursorVisible = false;
        }
    }
}

using System;

namespace ConsoleInvaders
{
    class Nave : Sprite
    {
        public Nave()
        {
            x = 40;
            y = 20;
            imagen = "/\\";
        }

        public void MoverDerecha()
        {
            x += 10;
            if (x >= 77) x = 77;
        }

        public void MoverIzquierda()
        {
            x -= 10;
            if (x <= 0) x = 0;
        }
    }
}

using System;

namespace ConsoleInvaders
{
    class Enemigo : Sprite
    {
        public Enemigo()
        {
```

```
            x = 40;
            y = 10;
            imagen = "][";
        }
    }
}
```

(6.6.3) Amplía el esqueleto del ConsoleInvaders (ejercicio 6.5.4): La clase Enemigo tendrá un constructor, sin parámetros, que prefijará su posición inicial. El constructor de la clase Nave recibirá como parámetros las coordenadas X e Y iniciales, para que se puedan cambiar desde el cuerpo del programa. Elimina las variables xNave e yNave de la clase Partida, que ya no serán necesarias.

```
using System;

namespace ConsoleInvaders
{
   public class Partida
   {
      Nave nave = new Nave(40, 20);
      Enemigo enemigo = new Enemigo();
      ConsoleKeyInfo tecla;

      public void Lanzar ()
      {
         Console.Clear();
         nave.Dibujar();
         enemigo.Dibujar();

         do
         {
            tecla = Console.ReadKey();
            if (tecla.Key == ConsoleKey.RightArrow){
               nave.MoverDerecha();
            }
            if (tecla.Key == ConsoleKey.LeftArrow)
            {
               nave.MoverIzquierda();
            }
            Console.Clear();
            nave.Dibujar();
            enemigo.Dibujar();

         } while (tecla.Key != ConsoleKey.Escape);

      }
   }
}

using System;

namespace ConsoleInvaders
{
   public class Nave : Sprite
   {

      public Nave(int x, int y)
      {
```

```
            this.x = x;
            this.y = y;
            imagen = "/\\";
        }


        public void MoverDerecha()
        {
            x += 10;
            if (x >= 77) x = 77;
        }

        public void MoverIzquierda()
        {
            x -= 10;
            if (x <= 0) x = 0;

        }
    }
}

using System;

namespace ConsoleInvaders
{
    public class Enemigo : Sprite
    {
        public Enemigo ()
        {
            x = 40;
            y = 10;
            imagen = "][";
        }
    }
}

using System;

namespace ConsoleInvaders
{
    public class Sprite
    {
        protected int x, y;
        protected string imagen;

        public void MoverA(int nuevaX, int nuevaY)
        {

        }

        public void Dibujar()
        {
            Console.SetCursorPosition(x, y);
            Console.Write(imagen);
            Console.CursorVisible = false;
        }
    }
}
```

Las demás clases permanecen igual.

(6.7.3) Amplía el esqueleto del ConsoleInvaders (6.6.3): La clase Nave tendrá un segundo constructor, sin parámetros, que prefijará su posición inicial a (500,600). La clase Enemigo tendrá un segundo constructor, con parámetros X e Y, para poder colocar un enemigo en cualquier punto desde Main.

```
using System;

namespace ConsoleInvaders
{
   public class Nave : Sprite
   {

      public Nave(int x, int y)
      {
         this.x = x;
         this.y = y;
         imagen = "/\\";
      }

      public Nave ()
      {
         x = 40;
         y = 20;
         imagen = "/\\";
      }


      public void MoverDerecha()
      {
         x += 10;
         if (x >= 78) x = 78;
      }

      public void MoverIzquierda()
      {
         x -= 10;
         if (x <= 0) x = 0;

      }
   }
}
```

```
using System;

namespace ConsoleInvaders
{
   public class Enemigo : Sprite
   {
      public Enemigo ()
      {
         x = 40;
         y = 10;
         imagen = "][";
      }

      public Enemigo(int x, int y)
      {
         this.x = x;
```

```
            this.y = y;
            imagen = "][";
        }
    }
}
```

Las demás clases permanecen igual.

(6.8.3) Crea una versión alternativa del esqueleto del ConsoleInvaders (6.7.3) en la que el constructor de Sprite escriba en pantalla "Creando sprite" y los constructores de Nave escriba en pantalla "Creando nave en posición prefijada" o "Creando nave en posición indicada por el usuario", según el caso. Comprueba su funcionamiento.

(7.2.3) Amplía el esqueleto del ConsoleInvaders (6.7.3), para que haya 10 enemigos en una misma fila (todos compartirán una misma coordenada Y, pero tendrán distinta coordenada X). Necesitarás un nuevo constructor en la clase Enemigo, que reciba los parámetros X e Y.

```
using System;

namespace ConsoleInvaders
{
    public class Partida
    {
        Nave nave = new Nave();
        Enemigo []enemigos = new Enemigo[10];
        ConsoleKeyInfo tecla;

        public void Lanzar ()
        {
            Console.Clear();
            nave.Dibujar();

            for (int i = 0; i < enemigos.Length; i++){
                enemigos[i] = new Enemigo(20 + (i * 4), 10);
                enemigos[i].Dibujar();
            }


            do
            {
                tecla = Console.ReadKey();
                if (tecla.Key == ConsoleKey.RightArrow){
                    nave.MoverDerecha();
                }
                if (tecla.Key == ConsoleKey.LeftArrow)
                {
                    nave.MoverIzquierda();
                }
                Console.Clear();
                nave.Dibujar();
                for (int i = 0; i < enemigos.Length; i++)
                {
                    //enemigos[i] = new Enemigo(i + 4, 10);
                    enemigos[i].Dibujar();
                }

            } while (tecla.Key != ConsoleKey.Escape);
```

```
    }
  }
}
```

Las demás clases permanecen igual.

(7.2.6) Amplía el esqueleto del ConsoleInvaders (7.2.3), para que haya tres tipos de enemigos, y un array que contenga 3x10 enemigos (3 filas, cada una con 10 enemigos de un mismo tipo, pero distinto del tipo de los elementos de las otras filas). Cada tipo de enemigos será una subclase de Enemigo, que se distinguirá por usar una "imagen" diferente. Puedes usar la "imagen" que quieras (siempre que sea un string de letras, como "}{" o "XX"). Si estas imágenes no se muestran correctamente en pantalla al lanzar una partida, no te preocupes, lo solucionaremos en el siguiente apartado.

```
using System;

namespace ConsoleInvaders
{
  class MainClass
  {
    public static void Main()
    {
      Juego juego = new Juego();
      juego.Lanzar();
    }
  }
}
```

```
using System;

namespace ConsoleInvaders
{
  public class Juego
  {
    public void Lanzar ()
    {
      Bienvenida bienvenida = new Bienvenida();
      Partida partida = new Partida();
      do
      {
        bienvenida.Lanzar();
        if (!bienvenida.Salir)
          partida.Lanzar();
      } while (!bienvenida.Salir);

    }
  }
}
```

```
using System;

namespace ConsoleInvaders
{
  public class Bienvenida
  {
    ConsoleKeyInfo tecla;
    public bool Salir{get;set;}
```

```
        public void Lanzar()
        {
            Console.Clear();
            Console.Write("Bienvenido a Console Invaders. Pulse Intro para jugar o ESC para salir."
);
            tecla = Console.ReadKey();
            if (tecla.Key == ConsoleKey.Escape)
            {
                Salir = true;
            }
        }
    }
}

using System;

namespace ConsoleInvaders
{
    public class Partida
    {
        Nave nave = new Nave();
        Enemigo []enemigos = new Enemigo[30];
        ConsoleKeyInfo tecla;

        public void Lanzar ()
        {
            Console.Clear();
            nave.Dibujar();

            for (int i = 0; i < 10; i++)
            {
                enemigos[i] = new Enemigo(20 + (i * 4), 8);
                enemigos[i].Dibujar();
            }

            for (int i = 0; i < 10; i++)
            {
                enemigos[i + 10] = new Enemigos1 (20 + (i * 4), 10);
                enemigos[i + 10].Dibujar();
            }

            for (int i = 0; i < 10; i++)
            {
                enemigos[i + 20] = new Enemigos2 (20 + (i * 4), 12);
                enemigos[i + 20].Dibujar();
            }

            do
            {
                tecla = Console.ReadKey();
                if (tecla.Key == ConsoleKey.RightArrow){
                    nave.MoverDerecha();
                }
                if (tecla.Key == ConsoleKey.LeftArrow)
                {
                    nave.MoverIzquierda();
                }
                Console.Clear();
                nave.Dibujar();
```

```
            for (int i = 0; i < 10; i++)
            {
                enemigos[i] = new Enemigo(20 + (i * 4), 8);
                enemigos[i].Dibujar();
            }

            for (int i = 0; i < 10; i++)
            {
                enemigos[i+10] = new Enemigos1 (20 + (i * 4), 10);
                enemigos[i+10].Dibujar();
            }

            for (int i = 0; i < 10; i++)
            {
                enemigos[i+20] = new Enemigos2 (20 + (i * 4), 12);
                enemigos[i+20].Dibujar();
            }

        } while (tecla.Key != ConsoleKey.Escape);

    }
  }
}
```

```
using System;

namespace ConsoleInvaders
{
  public class Sprite
  {
    protected int x, y;
    protected string imagen;

    public Sprite ()
    {
    }

    public void MoverA(int nuevaX, int nuevaY)
    {

    }

    public void Dibujar()
    {
      Console.SetCursorPosition(x, y);
      Console.Write(imagen);
      Console.CursorVisible = false;
    }
  }
}
```

```
using System;

namespace ConsoleInvaders
{
  public class Nave : Sprite
  {

    public Nave(int x, int y)
    {
```

```
            this.x = x;
            this.y = y;
            imagen = "/\\";
        }

        public Nave ()
        {
            x = 40;
            y = 20;
            imagen = "/\\";
        }


        public void MoverDerecha()
        {
            x += 10;
            if (x >= 78) x = 78;
        }

        public void MoverIzquierda()
        {
            x -= 10;
            if (x <= 0) x = 0;

        }
    }
}

using System;

namespace ConsoleInvaders
{
    public class Enemigo : Sprite
    {
        public Enemigo ()
        {
            x = 40;
            y = 10;
            imagen = "][";
        }

        public Enemigo(int x, int y)
        {
            this.x = x;
            this.y = y;
            imagen = "][";
        }
    }
}

using System;

namespace ConsoleInvaders
{
    public class Enemigos1 : Enemigo
    {

        public Enemigos1 (int x, int y)
        {
            this.x = x;
```

```
            this.y = y;
            imagen = "}{";
        }
    }
}

using System;

namespace ConsoleInvaders
{
    public class Enemigos2 : Enemigo
    {
        public Enemigos2(int x, int y)
        {
            this.x = x;
            this.y = y;
            imagen = ")(";
        }
    }
}
```

(7.3.5) Amplía el esqueleto de ConsoleInvaders (7.2.6) para que muestre las imágenes correctas de los enemigos, usando "virtual" y "override". Además, cada tipo de enemigos debe ser de un color distinto. (Nota: para cambiar colores puedes usar Console.ForegroundColor = ConsoleColor.Green;). La nave que maneja el usuario debe ser blanca.

(7.4.4) Amplía el esqueleto de ConsoleInvaders (7.3.5) para que en Enemigo haya un único constructor que reciba las coordenadas X e Y iniciales. Los constructores de los tres tipos de enemigos deben basarse en éste.

(7.5.4) Amplía el esqueleto de ConsoleInvaders (7.4.4), de modo que el constructor sin parámetros de la clase Nave se apoye en el constructor con parámetros de la misma clase, prefijando unas coordenadas que te parezcan las más adecuadas.

(7.7.9) Amplía el esqueleto de ConsoleInvaders (7.5.4), con una clase "BloqueDeEnemigos", que será la que contenga el array de enemigos, de modo que se simplifique la lógica de la clase Partida. Esta clase tendrá un método Dibujar, que mostrará todo el array en pantalla, y un método Mover, que moverá todo el bloque hacia la derecha y la izquierda de forma alternativa (deberás comprobar la posición inicial del primer enemigo y la posición final del último enemigo). En cada pasada por el bucle de juego deberás llamar a Mover.

```
using System;

namespace Segundo
{
    class Program
    {
        static void Main()
        {
            Juego juego = new Juego();
            juego.Lanzar();
        }
    }
```

```
}
```

```
using System;

namespace Segundo
{
    class Juego
    {
        Bienvenida bienvenida;
        Partida partida;

        public Juego()
        {
            bienvenida = new Bienvenida();
        }

        public void Lanzar()
        {
            do
            {
                bienvenida.Lanzar();
                if (!bienvenida.Salir)
                {
                    partida = new Partida();
                    partida.Lanzar();
                }
            } while (!bienvenida.Salir);
        }
    }
}
```

```
using System;

namespace Segundo

{
    class Bienvenida
    {
        ConsoleKeyInfo tecla;
        public bool Salir { get; private set; }

        public void Lanzar()
        {
            Console.Clear();
            Console.ForegroundColor = ConsoleColor.White;
            Console.Write("Bienvenido a Console Invaders. " +
             " Pulsa Intro para jugar o ESC para salir.");
            tecla = Console.ReadKey();
            if (tecla.Key == ConsoleKey.Escape)
            {
                Salir = true;
            }
        }
    }
}
```

```
using System;
using System.Threading;

namespace Segundo
{
    class Partida
```

```
    {
        ConsoleKeyInfo tecla;
        Nave nave;
        BloqueDeEnemigos bloque;

        public Partida()
        {
            nave = new Nave(40, 20);
            bloque = new BloqueDeEnemigos();
        }

        public void Lanzar()
        {
            Console.Clear();
            nave.Dibujar();
            bloque.Dibujar();
            bloque.Mover();

            do
            {
                Console.Clear();
                nave.Dibujar();
                bloque.Dibujar();
                bloque.Mover();

                if (Console.KeyAvailable)
                {
                    tecla = Console.ReadKey();
                    if (tecla.Key == ConsoleKey.RightArrow)
                        nave.MoverDerecha();
                    if (tecla.Key == ConsoleKey.LeftArrow)
                        nave.MoverIzquierda();
                }
                Thread.Sleep(80);
            } while (tecla.Key != ConsoleKey.Escape);
        }
    }
}

using System;

namespace Segundo
{
    class Nave : Sprite
    {
        public Nave(int x, int y)
        {
            this.x = x;
            this.y = y;
            imagen = "/\\";
        }

        public Nave() : this (40, 20)
        {

        }

        public void MoverDerecha()
        {
            x += 10;
            if (x >= 76) x = 76;
        }
```

```
        public void MoverIzquierda()
        {
            x -= 10;
            if (x <= 0) x = 0;
        }

        protected override string DevolverImagen()
        {
            return "<->";
        }

        protected override ConsoleColor DevolverColor()
        {
            return ConsoleColor.White;
        }
    }
}

using System;

namespace Segundo
{
    abstract class Sprite
    {
        protected int x, y;
        protected string imagen;

        public void MoverA(int nuevaX, int nuevaY)
        {
        }

        public void Dibujar()
        {
            Console.SetCursorPosition(x, y);
            Console.ForegroundColor = DevolverColor();
            Console.Write(DevolverImagen());
            Console.CursorVisible = false;
        }

        protected abstract string DevolverImagen();

        protected abstract ConsoleColor DevolverColor();

    }
}

using System;

namespace Segundo
{
    class Enemigo : Sprite
    {
        public Enemigo()
        {
            x = 40;
            y = 10;
            imagen = "][";
        }

        public Enemigo(int x, int y)
        {
```

```
            this.x = x;
            this.y = y;
            imagen = "][";
        }

        protected override string DevolverImagen()
        {
            return "][";
        }

        protected override ConsoleColor DevolverColor()
        {
            return ConsoleColor.Yellow;
        }
    }
}

using System;

namespace Segundo
{
    class Enemigo2 : Enemigo
    {
        public Enemigo2(int x, int y): base (x, y)
        {

        }

        protected override string DevolverImagen()
        {
            return "}{";
        }

        protected override ConsoleColor DevolverColor()
        {
            return ConsoleColor.Blue;
        }
    }
}

using System;

namespace Segundo
{
    class Enemigo3 : Enemigo
    {
        public Enemigo3(int x, int y): base (x, y)
        {

        }

        protected override string DevolverImagen()
        {
            return ")(";
        }

        protected override ConsoleColor DevolverColor()
        {
            return ConsoleColor.Green;
        }
    }
}
```

```
using System;

namespace Segundo
{
    class BloqueDeEnemigos
    {
        Enemigo[] enemigos;
        int x, y;
        int incremento;

        public BloqueDeEnemigos()
        {
            enemigos = new Enemigo[30];
            x = 20;
            y = 12;
            incremento = 1;
        }

        public void Dibujar()
        {
            for (int i = 0; i < 10; i++)
            {
                enemigos[i] = new Enemigo(x + (i * 4), y-4);
                enemigos[i].Dibujar();
            }
            for (int i = 0; i < 10; i++)
            {
                enemigos[i + 10] = new Enemigo2(x + (i * 4), y-2);
                enemigos[i + 10].Dibujar();
            }
            for (int i = 0; i < 10; i++)
            {
                enemigos[i + 20] = new Enemigo3(x + (i * 4), y);
                enemigos[i + 20].Dibujar();
            }
        }

        public void Mover()
        {
            x += incremento;
            if (x<=0 || x >= 40)
            {
                y++;
                incremento = -incremento;
            }
        }

    }
}
```

(7.7.10) Crea una clase Disparo en ConsoleInvaders. Cuando el usuario pulse cierta tecla (Espacio, por ejemplo), aparecerá un disparo encima de la nave, y se moverá hacia arriba hasta que desaparezca por la parte superior de la pantalla. Existirá un único disparo, y no se podrá volver a disparar si está activo (en pantalla). Inicialmente estará desactivado, y lo volverá a estar cuando llegue al margen de la pantalla.

```
using System;
```

```
namespace Segundo
{
    class Disparo: Sprite
    {
        public Disparo(int x, int y)
        {
            this.x = x+1;
            this.y = y;
            Activo = true;
        }

        public bool Activo { get; set; }

        protected override string DevolverImagen()
        {
            return "|";
        }

        protected override ConsoleColor DevolverColor()
        {
            return ConsoleColor.Yellow;
        }

        public void Mover()
        {
            y--;
            if (y == 0) Activo = false;
        }
    }
}
```

```
using System;
using System.Threading;

namespace Segundo
{
    class Partida
    {
        ConsoleKeyInfo tecla;
        Nave nave;
        BloqueDeEnemigos bloque;
        Disparo disparo;

        public Partida()
        {
            nave = new Nave(40, 20);
            bloque = new BloqueDeEnemigos();
            disparo = null;
        }

        public void Lanzar()
        {
            Console.Clear();
            nave.Dibujar();
            bloque.Dibujar();
            bloque.Mover();

            do
            {
                Console.Clear();
                nave.Dibujar();
                bloque.Dibujar();
```

```
                bloque.Mover();
                if (disparo!=null && disparo.Activo)
                {
                    disparo.Mover();
                    disparo.Dibujar();
                }

                if (Console.KeyAvailable)
                {
                    tecla = Console.ReadKey();
                    if (tecla.Key == ConsoleKey.RightArrow)
                        nave.MoverDerecha();
                    if (tecla.Key == ConsoleKey.LeftArrow)
                        nave.MoverIzquierda();
                    if (tecla.Key == ConsoleKey.Spacebar)
                        disparo = nave.Disparar();
                }
                Thread.Sleep(80);
            } while (tecla.Key != ConsoleKey.Escape);
        }
    }
}

using System;

namespace Segundo
{
    class Nave : Sprite
    {
        Disparo disparo;

        public Nave(int x, int y)
        {
            this.x = x;
            this.y = y;
            imagen = "/\\";
        }

        public Nave() : this (40, 20)
        {

        }

        public void MoverDerecha()
        {
            x += 10;
            if (x >= 76) x = 76;
        }

        public void MoverIzquierda()
        {
            x -= 10;
            if (x <= 0) x = 0;
        }

        protected override string DevolverImagen()
        {
            return "<->";
        }

        protected override ConsoleColor DevolverColor()
        {
```

```
            return ConsoleColor.White;
        }

        public Disparo Disparar()
        {
            if (disparo == null || !disparo.Activo)
            {
                disparo = new Disparo(x, y);
            }
            return disparo;
        }
    }
}
```

(7.7.11) En ConsoleInvaders, crea un método "ColisionaCon" en la clase Sprite, que reciba otro Sprite como parámetro y devuelva el valor booleano "true" si ambos sprites están "chocando" o "false" en caso contrario. Tendrás que pensar qué relación habrá entre las coordenadas X e Y de ambos sprites para que "choquen".

```
using System;

namespace Segundo
{
    abstract class Sprite
    {
        protected int x, y;
        protected string imagen;

        public void MoverA(int nuevaX, int nuevaY)
        {
        }

        public void Dibujar()
        {
            Console.SetCursorPosition(x, y);
            Console.ForegroundColor = DevolverColor();
            Console.Write(DevolverImagen());
            Console.CursorVisible = false;
        }

        protected abstract string DevolverImagen();

        protected abstract ConsoleColor DevolverColor();

        public bool ColisionaCon (Sprite sprite)
        {
            if (this.x == sprite.x && this.y == sprite.y)
                return true;
            else return false;
        }
    }
}
```

(7.7.12) En ConsoleInvaders, crea una clase Ovni, con un nuevo tipo de Enemigo que no estará siempre activo. Su método Mover hará que se mueva hacia la derecha si ya

está activo o, en caso contrario, que genere un número al azar para decidir si debe activarse.

```
using System;

namespace Segundo
{
    class Ovni : Sprite
    {
        Random generador;
        int aleatorio;

        public Ovni()
        {
            x = 0;
            y = 6;
            Activo = false;
            generador = new Random();
        }

        public bool Activo { get; set; }

        protected override string DevolverImagen()
        {
            if (Activo) return "XXXX";
            else return "";
        }

        protected override ConsoleColor DevolverColor()
        {
            return ConsoleColor.Yellow;
        }

        public void Mover()
        {
            if (!Activo)
            {
                aleatorio = generador.Next(1, 61);
                if (aleatorio == 2)
                {
                    Activo = true;
                    x = 0;
                }
            }
            else
            {
                x++;
                if (x >= 76) Activo = false;
            }
        }
    }
}

using System;
using System.Threading;

namespace Segundo
{
    class Partida
    {
        ConsoleKeyInfo tecla;
```

```
        Nave nave;
        BloqueDeEnemigos bloque;
        Disparo disparo;
        Ovni ovni;

        public Partida()
        {
            nave = new Nave(40, 20);
            bloque = new BloqueDeEnemigos();
            disparo = null;
            ovni = new Ovni();
        }

        public void Lanzar()
        {
            Console.Clear();
            nave.Dibujar();
            bloque.Dibujar();
            bloque.Mover();

            do
            {
                Console.Clear();
                nave.Dibujar();
                bloque.Dibujar();
                bloque.Mover();
                ovni.Mover();
                ovni.Dibujar();
                if (disparo!=null && disparo.Activo)
                {
                    disparo.Mover();
                    disparo.Dibujar();
                }

                if (Console.KeyAvailable)
                {
                    tecla = Console.ReadKey();
                    if (tecla.Key == ConsoleKey.RightArrow)
                        nave.MoverDerecha();
                    if (tecla.Key == ConsoleKey.LeftArrow)
                        nave.MoverIzquierda();
                    if (tecla.Key == ConsoleKey.Spacebar)
                        disparo = nave.Disparar();
                }
                Thread.Sleep(120);
            } while (tecla.Key != ConsoleKey.Escape);
        }
    }
}
```

(7.7.13) En ConsoleInvaders, comprueba colisiones entre el disparo y el Ovni. Si hay colisión, desaparecerán ambos y el jugador obtendrá 50 puntos.

```
using System;
using System.Threading;

namespace Segundo
{
    class Partida
    {
```

```
        ConsoleKeyInfo tecla;
        Nave nave;
        BloqueDeEnemigos bloque;
        Disparo disparo;
        Ovni ovni;

        public Partida()
        {
            nave = new Nave(40, 20);
            bloque = new BloqueDeEnemigos();
            disparo = null;
            ovni = new Ovni();
        }

        public void Lanzar()
        {
            Console.Clear();
            nave.Dibujar();
            bloque.Dibujar();
            bloque.Mover();

            do
            {
                Console.Clear();
                nave.Dibujar();
                bloque.Dibujar();
                bloque.Mover();
                ovni.Mover();
                ovni.Dibujar();
                if (disparo!=null && disparo.Activo)
                {
                    disparo.Mover();
                    disparo.Dibujar();
                }
                if ((disparo!=null) && (disparo.ColisionaCon(ovni)))
                {
                    disparo.Activo = false;
                    ovni.Activo = false;
                    //marcador += 50;
                }

                if (Console.KeyAvailable)
                {
                    tecla = Console.ReadKey();
                    if (tecla.Key == ConsoleKey.RightArrow)
                        nave.MoverDerecha();
                    if (tecla.Key == ConsoleKey.LeftArrow)
                        nave.MoverIzquierda();
                    if (tecla.Key == ConsoleKey.Spacebar)
                        disparo = nave.Disparar();
                }
                Thread.Sleep(120);
            } while (tecla.Key != ConsoleKey.Escape);
        }
    }
}

using System;

namespace Segundo
{
    abstract class Sprite
```

```
    {
        protected int x, y;
        protected string imagen;

        public void MoverA(int nuevaX, int nuevaY)
        {
        }

        public void Dibujar()
        {
            Console.SetCursorPosition(x, y);
            Console.ForegroundColor = DevolverColor();
            Console.Write(DevolverImagen());
            Console.CursorVisible = false;
        }

        protected abstract string DevolverImagen();

        protected abstract ConsoleColor DevolverColor();

        public bool ColisionaCon (Sprite sprite)
        {
            if ((this.x == sprite.x|| this.x == sprite.x+1 || this.x ==
sprite.x+2 ||
                this.x == sprite.x+3) && (this.y == sprite.y))
                return true;
            else return false;
        }
    }
}
```

(7.7.14) En ConsoleInvaders, comprueba colisiones entre el disparo y el bloque de enemigos. Si el disparo toca algún enemigo, ambos desaparecerán y el jugador obtendrá 10 puntos.

```
using System;
using System.Threading;

namespace Segundo
{
    class Partida
    {
        ConsoleKeyInfo tecla;
        Nave nave;
        BloqueDeEnemigos bloque;
        Disparo disparo;
        Ovni ovni;

        public Partida()
        {
            nave = new Nave(40, 20);
            bloque = new BloqueDeEnemigos();
            disparo = null;
            ovni = new Ovni();
        }

        public void Lanzar()
        {
            Console.Clear();
            nave.Dibujar();
            bloque.Dibujar();
```

```
            do
            {
                Console.Clear();
                nave.Dibujar();
                bloque.Dibujar();
                bloque.Mover();
                ovni.Mover();
                ovni.Dibujar();
                if (disparo!=null && disparo.Activo)
                {
                    disparo.Mover();
                    disparo.Dibujar();
                    if (disparo.ColisionaCon(ovni))
                    {
                        disparo.Activo = false;
                        ovni.Activo = false;
                        //marcador += 50;
                    }
                    for (int i=0; i<30; i++)
                    {
                        if (disparo.ColisionaCon(bloque.Enemigos[i]) &&
bloque.Enemigos[i].Activo)
                        {
                            disparo.Activo = false;
                            bloque.Enemigos[i].Activo = false;
                            //marcador += 10;
                        }
                    }
                }

                if (Console.KeyAvailable)
                {
                    tecla = Console.ReadKey();
                    if (tecla.Key == ConsoleKey.RightArrow)
                        nave.MoverDerecha();
                    if (tecla.Key == ConsoleKey.LeftArrow)
                        nave.MoverIzquierda();
                    if (tecla.Key == ConsoleKey.Spacebar)
                        disparo = nave.Disparar();
                }
                Thread.Sleep(120);
            } while (tecla.Key != ConsoleKey.Escape);
        }
    }
}

using System;

namespace Segundo
{
    class Enemigo : Sprite
    {
        public bool Activo { get; set; }

        public Enemigo()
        {
            X = 40;
            Y = 10;
            imagen = "][";
            Activo = true;
        }
```

```
        public Enemigo(int x, int y)
        {
            this.X = x;
            this.Y = y;
            imagen = "][";
            Activo = true;
        }

        protected override string DevolverImagen()
        {
            return "][";
        }

        protected override ConsoleColor DevolverColor()
        {
            return ConsoleColor.Yellow;
        }

        protected override int DevolverLongitud()
        {
            return 2;
        }
    }
}

using System;

namespace Segundo
{
    abstract class Sprite
    {
        protected string imagen;

        public int X { get; set; }

        public int Y { get; set; }

        public void Dibujar()
        {
            Console.SetCursorPosition(X, Y);
            Console.ForegroundColor = DevolverColor();
            Console.Write(DevolverImagen());
            Console.CursorVisible = false;
        }

        protected abstract string DevolverImagen();

        protected abstract ConsoleColor DevolverColor();

        protected abstract int DevolverLongitud();

        public bool ColisionaCon (Sprite sprite)
        {
            if (this.Y != sprite.Y)
                return false;
            else
            {
                for (int i = 0; i < this.DevolverLongitud(); i++)
                {
                    for (int j = 0; j < sprite.DevolverLongitud(); j++)
                    {
```

```
                        if ((this.X + i) == (sprite.X + j))
                            return true;
                    }
                }
            }
            return false;
        }
    }
}

using System;

namespace Segundo
{
    class BloqueDeEnemigos
    {
        int x, y;
        int incremento;

        public BloqueDeEnemigos()
        {
            Enemigos = new Enemigo[30];
            x = 20;
            y = 12;
            incremento = 1;
            for (int i = 0; i < 10; i++)
                Enemigos[i] = new Enemigo(x + (i * 4), y - 4);
            for (int i = 0; i < 10; i++)
                Enemigos[i + 10] = new Enemigo2(x + (i * 4), y - 2);
            for (int i = 0; i < 10; i++)
                Enemigos[i + 20] = new Enemigo3(x + (i * 4), y);
        }

        public Enemigo[] Enemigos { get; set; }

        public void Dibujar()
        {
            for (int i = 0; i < 30; i++)
            {
                if (Enemigos[i].Activo)
                    Enemigos[i].Dibujar();
            }
        }

        public void Mover()
        {
            x += incremento;
            if (x<=0 || x >= 40)
            {
                y++;
                incremento = -incremento;
            }
            for (int i = 0; i < 10; i++)
            {
                Enemigos[i].X = x + (i * 4);
                Enemigos[i].Y = y-4;
            }

            for (int i = 0; i < 10; i++)
            {
                Enemigos[i + 10].X = x + (i * 4);
                Enemigos[i + 10].Y = y-2;
```

```
            }

            for (int i = 0; i < 10; i++)
            {
                Enemigos[i + 20].X = x + (i * 4);
                Enemigos[i + 20].Y = y;
            }
        }

    }
}
```

El resto de las clases permanecen igual.

(7.7.15) En ConsoleInvaders, añade una clase "Marcador", que muestre la puntuación y la cantidad de vidas restantes (que por ahora, siempre será 3).

```
using System;

namespace Segundo
{
    class Bienvenida
    {
        ConsoleKeyInfo tecla;
        public bool Salir { get; private set; }

        public void Lanzar()
        {
            Console.Clear();
            Console.ForegroundColor = ConsoleColor.White;
            Console.Write("Bienvenido a Console Invaders. " +
             " Pulsa Intro para jugar o ESC para salir.");
            tecla = Console.ReadKey();
            if (tecla.Key == ConsoleKey.Escape)
            {
                Salir = true;
            }
        }
    }
}

using System;

namespace Segundo
{
    class BloqueDeEnemigos
    {
        int x, y;
        int incremento;

        public BloqueDeEnemigos()
        {
            Enemigos = new Enemigo[30];
            x = 20;
            y = 12;
            incremento = 1;
            for (int i = 0; i < 10; i++)
                Enemigos[i] = new Enemigo(x + (i * 4), y - 4);
            for (int i = 0; i < 10; i++)
                Enemigos[i + 10] = new Enemigo2(x + (i * 4), y - 2);
            for (int i = 0; i < 10; i++)
```

```
                Enemigos[i + 20] = new Enemigo3(x + (i * 4), y);
        }

        public Enemigo[] Enemigos { get; set; }

        public void Dibujar()
        {
            for (int i = 0; i < 30; i++)
            {
                if (Enemigos[i].Activo)
                    Enemigos[i].Dibujar();
            }
        }

        public void Mover()
        {
            x += incremento;
            if (x<=0 || x >= 40)
            {
                y++;
                incremento = -incremento;
            }
            for (int i = 0; i < 10; i++)
            {
                Enemigos[i].X = x + (i * 4);
                Enemigos[i].Y = y-4;
            }

            for (int i = 0; i < 10; i++)
            {
                Enemigos[i + 10].X = x + (i * 4);
                Enemigos[i + 10].Y = y-2;
            }

            for (int i = 0; i < 10; i++)
            {
                Enemigos[i + 20].X = x + (i * 4);
                Enemigos[i + 20].Y = y;
            }
        }

    }
}

using System;

namespace Segundo
{
    class Disparo: Sprite
    {
        public Disparo(int x, int y)
        {
            this.X = x+1;
            this.Y = y;
            Activo = true;
        }

        public bool Activo { get; set; }

        protected override string DevolverImagen()
        {
            return "|";
```

```
        }

        protected override ConsoleColor DevolverColor()
        {
            return ConsoleColor.Yellow;
        }

        public void Mover()
        {
            Y--;
            if (Y == 0) Activo = false;
        }

        protected override int DevolverLongitud()
        {
            return 1;
        }
    }
}

using System;

namespace Segundo
{
    class Enemigo : Sprite
    {
        public bool Activo { get; set; }

        public Enemigo()
        {
            X = 40;
            Y = 10;
            imagen = "][";
            Activo = true;
        }

        public Enemigo(int x, int y)
        {
            this.X = x;
            this.Y = y;
            imagen = "][";
            Activo = true;
        }

        protected override string DevolverImagen()
        {
            return "][";
        }

        protected override ConsoleColor DevolverColor()
        {
            return ConsoleColor.Yellow;
        }

        protected override int DevolverLongitud()
        {
            return 2;
        }
    }
}

using System;
```

```
namespace Segundo
{
    class Enemigo2 : Enemigo
    {
        public Enemigo2(int x, int y): base (x, y)
        {

        }

        protected override string DevolverImagen()
        {
            return "}{";
        }

        protected override ConsoleColor DevolverColor()
        {
            return ConsoleColor.Blue;
        }
    }
}
```

```
using System;

namespace Segundo
{
    class Enemigo3 : Enemigo
    {
        public Enemigo3(int x, int y): base (x, y)
        {

        }

        protected override string DevolverImagen()
        {
            return ")(";
        }

        protected override ConsoleColor DevolverColor()
        {
            return ConsoleColor.Green;
        }
    }
}
```

```
using System;

namespace Segundo
{
    class Juego
    {
        Bienvenida bienvenida;
        Partida partida;

        public Juego()
        {
            bienvenida = new Bienvenida();
        }

        public void Lanzar()
        {
            do
```

```
            {
                bienvenida.Lanzar();
                if (!bienvenida.Salir)
                {
                    partida = new Partida();
                    partida.Lanzar();
                }
            } while (!bienvenida.Salir);
        }
    }
}
```

```
using System;

namespace Segundo
{
    class Marcador
    {
        public static int vidas = 3;
        public static int score = 0;

        public Marcador()
        {

        }

        public void ActualizarMarcador()
        {
            Console.SetCursorPosition(3, 1);
            Console.ForegroundColor = ConsoleColor.White;
            Console.Write("VIDAS: {0}\tSCORE: {1}", vidas, score);
            Console.CursorVisible = false;
        }
    }
}
```

```
using System;

namespace Segundo
{
    class Nave : Sprite
    {
        Disparo disparo;

        public Nave(int x, int y)
        {
            this.X = x;
            this.Y = y;
            imagen = "/\\";
        }

        public Nave() : this (40, 20)
        {

        }

        public void MoverDerecha()
        {
            X += 10;
            if (X >= 76) X = 76;
        }
```

```
        public void MoverIzquierda()
        {
            X -= 10;
            if (X <= 0) X = 0;
        }

        protected override string DevolverImagen()
        {
            return "<->";
        }

        protected override ConsoleColor DevolverColor()
        {
            return ConsoleColor.White;
        }

        public Disparo Disparar()
        {
            if (disparo == null || !disparo.Activo)
            {
                disparo = new Disparo(X, Y);
            }
            return disparo;
        }

        protected override int DevolverLongitud()
        {
            return 3;
        }
    }
}

using System;

namespace Segundo
{
    class Ovni : Sprite
    {
        Random generador;
        int aleatorio;

        public Ovni()
        {
            X = 0;
            Y = 6;
            Activo = false;
            generador = new Random();
        }

        public bool Activo { get; set; }

        protected override string DevolverImagen()
        {
            if (Activo) return "(||)";
            else return "";
        }

        protected override ConsoleColor DevolverColor()
        {
            return ConsoleColor.Yellow;
        }
```

```
        public void Mover()
        {
            if (!Activo)
            {
                aleatorio = generador.Next(1, 61);
                if (aleatorio == 2)
                {
                    Activo = true;
                    X = 0;
                }
            }
            else
            {
                X++;
                if (X >= 76) Activo = false;
            }
        }

        protected override int DevolverLongitud()
        {
            return 4;
        }
    }
}

using System;
using System.Threading;

namespace Segundo
{
    class Partida
    {
        ConsoleKeyInfo tecla;
        Nave nave;
        BloqueDeEnemigos bloque;
        Disparo disparo;
        Ovni ovni;
        Marcador marcador;

        public Partida()
        {
            nave = new Nave(40, 20);
            bloque = new BloqueDeEnemigos();
            disparo = null;
            ovni = new Ovni();
            marcador = new Marcador();
        }

        public void Lanzar()
        {
            Console.Clear();
            nave.Dibujar();
            bloque.Dibujar();

            do
            {
                Console.Clear();
                nave.Dibujar();
                bloque.Dibujar();
                bloque.Mover();
                ovni.Mover();
```

```
                ovni.Dibujar();
                marcador.ActualizarMarcador();
                if (disparo!=null && disparo.Activo)
                {
                    disparo.Mover();
                    disparo.Dibujar();
                    if (disparo.ColisionaCon(ovni))
                    {
                        disparo.Activo = false;
                        ovni.Activo = false;
                        Marcador.score += 50;
                    }
                    for (int i=0; i<30; i++)
                    {
                        if (disparo.ColisionaCon(bloque.Enemigos[i]) &&
bloque.Enemigos[i].Activo)
                        {
                            disparo.Activo = false;
                            bloque.Enemigos[i].Activo = false;
                            Marcador.score += 10;
                        }
                    }
                }

                if (Console.KeyAvailable)
                {
                    tecla = Console.ReadKey();
                    if (tecla.Key == ConsoleKey.RightArrow)
                        nave.MoverDerecha();
                    if (tecla.Key == ConsoleKey.LeftArrow)
                        nave.MoverIzquierda();
                    if (tecla.Key == ConsoleKey.Spacebar)
                        disparo = nave.Disparar();
                }
                Thread.Sleep(120);
            } while (tecla.Key != ConsoleKey.Escape);
        }
    }
}

using System;

namespace Segundo
{
    class Program
    {
        static void Main()
        {
            Juego juego = new Juego();
            juego.Lanzar();
        }
    }
}

using System;

namespace Segundo
{
    abstract class Sprite
    {
        protected string imagen;
```

```
        public int X { get; set; }

        public int Y { get; set; }

        public void Dibujar()
        {
            Console.SetCursorPosition(X, Y);
            Console.ForegroundColor = DevolverColor();
            Console.Write(DevolverImagen());
            Console.CursorVisible = false;
        }

        protected abstract string DevolverImagen();

        protected abstract ConsoleColor DevolverColor();

        protected abstract int DevolverLongitud();

        public bool ColisionaCon (Sprite sprite)
        {
            if (this.Y != sprite.Y)
                return false;
            else
            {
                for (int i = 0; i < this.DevolverLongitud(); i++)
                {
                    for (int j = 0; j < sprite.DevolverLongitud(); j++)
                    {
                        if ((this.X + i) == (sprite.X + j))
                            return true;
                    }
                }
            }
            return false;
        }
    }
}
```

(7.7.16) En ConsoleInvaders, añade la posibilidad de que algunos enemigos al azar puedan disparar (los disparos de los enemigos "va hacia abajo"; piensa si crear una nueva clase o ampliar las funcionalidades de la clase Disparo que ya tienes). Ajusta la frecuencia de disparos, de modo que el juego continúe siendo jugable. Si uno de esos disparos impacta con la nave, se perderá una vida y el disparo desaparecerá. Si se pierden las 3 vidas, acaba la partida y se volverá a la pantalla de presentación.

```
using System;

namespace ConsoleInvaders
{
    class Bienvenida
    {
        ConsoleKeyInfo tecla;
        public bool Salir { get; private set; }

        public void Lanzar()
        {
            Console.Clear();
            Console.SetCursorPosition(33, 6);
            Console.ForegroundColor = ConsoleColor.Yellow;
```

```
            Console.Write("CONSOLE INVADERS");
            Console.SetCursorPosition(20, 7);
            Console.ForegroundColor = ConsoleColor.White;
            Console.Write("(Pulsa Intro para jugar o ESC para salir)");
            Console.CursorVisible = false;
            tecla = Console.ReadKey();
            if (tecla.Key == ConsoleKey.Escape)
            {
                Salir = true;
            }
        }
    }
}

using System;

namespace ConsoleInvaders
{
    class BloqueDeEnemigos
    {
        int x, y;
        int incremento;
        Disparo disparo;
        Random generador;
        int numAleatorio;

        public BloqueDeEnemigos()
        {
            Enemigos = new Enemigo[30];
            x = 20;
            y = 12;
            generador = new Random();
            incremento = 1;
            for (int i = 0; i < 10; i++)
                Enemigos[i] = new Enemigo(x + (i * 4), y - 4);
            for (int i = 0; i < 10; i++)
                Enemigos[i + 10] = new Enemigo2(x + (i * 4), y - 2);
            for (int i = 0; i < 10; i++)
                Enemigos[i + 20] = new Enemigo3(x + (i * 4), y);
        }

        public Enemigo[] Enemigos { get; set; }

        public void Dibujar()
        {
            for (int i = 0; i < 30; i++)
            {
                if (Enemigos[i].Activo)
                    Enemigos[i].Dibujar();
            }
        }

        public void Mover()
        {
            x += incremento;
            if (x<=0 || x >= 40)
            {
                y++;
                incremento = -incremento;
            }
            for (int i = 0; i < 10; i++)
            {
```

```
                Enemigos[i].X = x + (i * 4);
                Enemigos[i].Y = y-4;
            }

            for (int i = 0; i < 10; i++)
            {
                Enemigos[i + 10].X = x + (i * 4);
                Enemigos[i + 10].Y = y-2;
            }

            for (int i = 0; i < 10; i++)
            {
                Enemigos[i + 20].X = x + (i * 4);
                Enemigos[i + 20].Y = y;
            }
        }

        public Disparo Disparar()
        {
            numAleatorio = generador.Next(1, 30);

            if (disparo == null || !disparo.Activo)
            {
                disparo = new Disparo(Enemigos[numAleatorio].X,
Enemigos[numAleatorio].Y);
            }
            return disparo;
        }
    }
}

using System;

namespace ConsoleInvaders
{
    class Disparo: Sprite
    {
        public Disparo(int x, int y)
        {
            this.X = x+1;
            this.Y = y;
            Activo = true;
        }

        public bool Activo { get; set; }

        protected override string DevolverImagen()
        {
            return "|";
        }

        protected override ConsoleColor DevolverColor()
        {
            return ConsoleColor.Yellow;
        }

        public void MoverArriba()
        {
            Y--;
            if (Y == 0) Activo = false;
        }
```

```
        public void MoverAbajo()
        {
            Y++;
            if (Y >= 23) Activo = false;
        }

        protected override int DevolverLongitud()
        {
            return 1;
        }
    }
}

using System;

namespace ConsoleInvaders
{
    class Enemigo : Sprite
    {
        public bool Activo { get; set; }

        public Enemigo()
        {
            X = 40;
            Y = 10;
            imagen = "][";
            Activo = true;
        }

        public Enemigo(int x, int y)
        {
            this.X = x;
            this.Y = y;
            imagen = "][";
            Activo = true;
        }

        protected override string DevolverImagen()
        {
            return "][";
        }

        protected override ConsoleColor DevolverColor()
        {
            return ConsoleColor.Yellow;
        }

        protected override int DevolverLongitud()
        {
            return 2;
        }
    }
}

using System;

namespace ConsoleInvaders
{
    class Enemigo2 : Enemigo
    {
        public Enemigo2(int x, int y): base (x, y)
        {
```

```
        }

        protected override string DevolverImagen()
        {
            return "}{";
        }

        protected override ConsoleColor DevolverColor()
        {
            return ConsoleColor.Blue;
        }
    }
}
```

```
using System;

namespace ConsoleInvaders
{
    class Enemigo3 : Enemigo
    {
        public Enemigo3(int x, int y): base (x, y)
        {

        }

        protected override string DevolverImagen()
        {
            return ")(";
        }

        protected override ConsoleColor DevolverColor()
        {
            return ConsoleColor.Green;
        }
    }
}
```

```
using System;

namespace ConsoleInvaders
{
    class Juego
    {
        Bienvenida bienvenida;
        Partida partida;

        public Juego()
        {
            bienvenida = new Bienvenida();
        }

        public void Lanzar()
        {
            do
            {
                bienvenida.Lanzar();
                if (!bienvenida.Salir)
                {
                    partida = new Partida();
                    partida.Lanzar();
                }
```

```
            } while (!bienvenida.Salir);
        }
    }
}

using System;

namespace ConsoleInvaders
{
    class Marcador
    {
        private int vidas;
        private int score;

        public Marcador()
        {
            vidas = 3;
            score = 0;
        }

        public void ActualizarMarcador()
        {
            Console.SetCursorPosition(3, 1);
            Console.ForegroundColor = ConsoleColor.White;
            Console.Write("VIDAS: {0}\tSCORE: {1}", vidas, score);
            Console.CursorVisible = false;
        }

        public int SumarPuntos (int puntos)
        {
            return score += puntos;
        }

        public void RestarVidas()
        {
            vidas--;
        }

        public int CuantasVidasQuedan()
        {
            return vidas;
        }
    }
}

using System;

namespace ConsoleInvaders
{
    class Nave : Sprite
    {
        Disparo disparo;

        public Nave(int x, int y)
        {
            this.X = x;
            this.Y = y;
            imagen = "/\\";
        }

        public Nave() : this (40, 20)
        {
```

```
        }

        public void MoverDerecha()
        {
            X += 10;
            if (X >= 76) X = 76;
        }

        public void MoverIzquierda()
        {
            X -= 10;
            if (X <= 0) X = 0;
        }

        protected override string DevolverImagen()
        {
            return "<->";
        }

        protected override ConsoleColor DevolverColor()
        {
            return ConsoleColor.White;
        }

        public Disparo Disparar()
        {
            if (disparo == null || !disparo.Activo)
            {
                disparo = new Disparo(X, Y);
            }
            return disparo;
        }

        protected override int DevolverLongitud()
        {
            return 3;
        }

        public void Reset()
        {
            X = 40;
            Y = 20;
        }
    }
}

using System;

namespace ConsoleInvaders
{
    class Ovni : Sprite
    {
        Random generador;
        int aleatorio;

        public Ovni()
        {
            X = 0;
            Y = 6;
            Activo = false;
            generador = new Random();
```

```
        }

        public bool Activo { get; set; }

        protected override string DevolverImagen()
        {
            if (Activo) return "(||)";
            else return "";
        }

        protected override ConsoleColor DevolverColor()
        {
            return ConsoleColor.Yellow;
        }

        public void Mover()
        {
            if (!Activo)
            {
                aleatorio = generador.Next(1, 61);
                if (aleatorio == 2)
                {
                    Activo = true;
                    X = 0;
                }
            }
            else
            {
                X++;
                if (X >= 76) Activo = false;
            }
        }

        protected override int DevolverLongitud()
        {
            return 4;
        }
    }
}

using System;
using System.Threading;

namespace ConsoleInvaders
{
    class Partida
    {
        ConsoleKeyInfo tecla;
        Nave nave;
        BloqueDeEnemigos bloque;
        Disparo disparoNave;
        Disparo disparoEnemigo;
        Ovni ovni;
        Marcador marcador;
        private bool finPartida;
        Random generador;
        private int numAleatorio;

        public Partida()
        {
            nave = new Nave(40, 20);
            bloque = new BloqueDeEnemigos();
```

```
            disparoNave = null;
            disparoEnemigo = null;
            ovni = new Ovni();
            marcador = new Marcador();
            finPartida = false;
            generador = new Random();
        }

        public void GameOver()
        {
            Console.Clear();
            Console.SetCursorPosition(37, 12);
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.Write("GAME OVER");
            Console.CursorVisible = false;
            Console.ReadLine();
        }

        public void Lanzar()
        {
            Console.Clear();
            nave.Dibujar();
            bloque.Dibujar();

            do
            {
                Console.Clear();
                nave.Dibujar();
                bloque.Dibujar();
                bloque.Mover();
                ovni.Mover();
                ovni.Dibujar();
                marcador.ActualizarMarcador();
                numAleatorio = generador.Next(1, 7);
                if (numAleatorio == 3)
                    disparoEnemigo = bloque.Disparar();
                if (disparoNave!=null && disparoNave.Activo)
                {
                    disparoNave.MoverArriba();
                    disparoNave.Dibujar();
                    if (disparoNave.ColisionaCon(ovni))
                    {
                        disparoNave.Activo = false;
                        ovni.Activo = false;
                        marcador.SumarPuntos(50);
                    }
                    for (int i=0; i<30; i++)
                    {
                        if (disparoNave.ColisionaCon(bloque.Enemigos[i]) &&
bloque.Enemigos[i].Activo)
                        {
                            disparoNave.Activo = false;
                            bloque.Enemigos[i].Activo = false;
                            marcador.SumarPuntos(10);
                        }
                    }
                }

                if (disparoEnemigo != null && disparoEnemigo.Activo)
                {
                    disparoEnemigo.MoverAbajo();
                    disparoEnemigo.Dibujar();
```

```
                    if (disparoEnemigo.ColisionaCon(nave))
                    {
                        disparoEnemigo.Activo = false;
                        marcador.RestarVidas();
                        nave.Reset();
                        if (marcador.CuantasVidasQuedan() == 0)
                        {
                            GameOver();
                            finPartida = true;
                        }
                    }
                }

                for (int i=0; i<30; i++)
                {
                    if ((bloque.Enemigos[i].ColisionaCon(nave)) &&
(bloque.Enemigos[i].Activo))
                    {
                        GameOver();
                        finPartida = true;
                    }
                }

                if (Console.KeyAvailable)
                {
                    tecla = Console.ReadKey();
                    if (tecla.Key == ConsoleKey.RightArrow)
                        nave.MoverDerecha();
                    if (tecla.Key == ConsoleKey.LeftArrow)
                        nave.MoverIzquierda();
                    if (tecla.Key == ConsoleKey.Spacebar)
                        disparoNave = nave.Disparar();
                    if (tecla.Key == ConsoleKey.Escape)
                        finPartida = true;
                }
                Thread.Sleep(120);
            } while (!finPartida);
        }
    }
}

using System;

namespace ConsoleInvaders
{
    class Program
    {
        static void Main()
        {
            Juego juego = new Juego();
            juego.Lanzar();
        }
    }
}

using System;

namespace ConsoleInvaders
{
    abstract class Sprite
    {
        protected string imagen;
```

```
        public int X { get; set; }

        public int Y { get; set; }

        public void Dibujar()
        {
            Console.SetCursorPosition(X, Y);
            Console.ForegroundColor = DevolverColor();
            Console.Write(DevolverImagen());
            Console.CursorVisible = false;
        }

        protected abstract string DevolverImagen();

        protected abstract ConsoleColor DevolverColor();

        protected abstract int DevolverLongitud();

        public bool ColisionaCon (Sprite sprite)
        {
            if (this.Y != sprite.Y)
                return false;
            else
            {
                for (int i = 0; i < this.DevolverLongitud(); i++)
                {
                    for (int j = 0; j < sprite.DevolverLongitud(); j++)
                    {
                        if ((this.X + i) == (sprite.X + j))
                            return true;
                    }
                }
            }
            return false;
        }
    }
}
```

(7.7.17) En ConsoleInvaders, crea la estructura que sea necesaria para almacenar las "mejores puntuaciones", que se actualizarán al terminar cada partida y se mostrarán en la pantalla de bienvenida.

```
using System;
using System.Collections.Generic;

namespace ConsoleInvaders
{
    class Juego
    {
        Bienvenida bienvenida;
        Partida partida;
        List<int> puntuaciones = new List<int>();

        public Juego()
        {
            bienvenida = new Bienvenida();
        }

        public void Lanzar()
```

```
        {
            do
            {
                bienvenida.Lanzar(puntuaciones);
                if (!bienvenida.Salir)
                {
                    partida = new Partida();
                    puntuaciones.Add(partida.Lanzar());
                    puntuaciones.Sort(CompararNumerosDescendiente);
                }
            } while (!bienvenida.Salir);
        }

        private int CompararNumerosDescendiente(int num1, int num2)
        {
            if (num2 > num1)
                return 1;
            else if (num2 < num1)
                return -1;
            else
                return 0;
        }
    }
}

using System;
using System.Collections.Generic;

namespace ConsoleInvaders
{
    class Bienvenida
    {
        ConsoleKeyInfo tecla;
        public bool Salir { get; private set; }

        public void Lanzar(List<int> puntuaciones)
        {
            Console.Clear();
            Console.SetCursorPosition(33, 6);
            Console.ForegroundColor = ConsoleColor.Yellow;
            Console.Write("CONSOLE INVADERS");
            Console.SetCursorPosition(20, 7);
            Console.ForegroundColor = ConsoleColor.White;
            Console.Write("(Pulsa Intro para jugar o ESC para salir)");
            Console.SetCursorPosition(30, 10);
            Console.ForegroundColor = ConsoleColor.Gray;
            Console.Write("MEJORES PUNTUACIONES: ");
            for (int i=0; i<puntuaciones.Count && i<3; i++)
            {
                Console.SetCursorPosition(30, 11+i);
                Console.ForegroundColor = ConsoleColor.Gray;
                Console.Write("{0}º) {1}", i+1, puntuaciones[i]);
            }
            Console.CursorVisible = false;
            tecla = Console.ReadKey();
            if (tecla.Key == ConsoleKey.Escape)
            {
                Salir = true;
            }
        }
```

```
    }
}

using System;

namespace ConsoleInvaders
{
    class Marcador
    {
        private int vidas;
        private int score;

        public Marcador()
        {
            vidas = 3;
            score = 0;
        }

        public void ActualizarMarcador()
        {
            Console.SetCursorPosition(3, 1);
            Console.ForegroundColor = ConsoleColor.White;
            Console.Write("VIDAS: {0}\tSCORE: {1}", vidas, score);
            Console.CursorVisible = false;
        }

        public int SumarPuntos (int puntos)
        {
            return score += puntos;
        }

        public void RestarVidas()
        {
            vidas--;
        }

        public int CuantasVidasQuedan()
        {
            return vidas;
        }

        public int DevolverPuntuacionFinal()
        {
            return score;
        }
    }
}

using System;
using System.Threading;

namespace ConsoleInvaders
{
    class Partida
    {
        ConsoleKeyInfo tecla;
        Nave nave;
        BloqueDeEnemigos bloque;
        Disparo disparoNave;
        Disparo disparoEnemigo;
        Ovni ovni;
        Marcador marcador;
```

```
        private bool finPartida;
        Random generador;
        private int numAleatorio;

        public Partida()
        {
            nave = new Nave(40, 20);
            bloque = new BloqueDeEnemigos();
            disparoNave = null;
            disparoEnemigo = null;
            ovni = new Ovni();
            marcador = new Marcador();
            finPartida = false;
            generador = new Random();
        }

        public void GameOver()
        {
            Console.Clear();
            Console.SetCursorPosition(37, 12);
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.Write("GAME OVER");
            Console.CursorVisible = false;
            Console.ReadLine();
        }

        public int Lanzar()
        {
            Console.Clear();
            nave.Dibujar();
            bloque.Dibujar();

            do
            {
                Console.Clear();
                nave.Dibujar();
                bloque.Dibujar();
                bloque.Mover();
                ovni.Mover();
                ovni.Dibujar();
                marcador.ActualizarMarcador();
                numAleatorio = generador.Next(1, 7);
                if (numAleatorio == 3)
                    disparoEnemigo = bloque.Disparar();
                if (disparoNave!=null && disparoNave.Activo)
                {
                    disparoNave.MoverArriba();
                    disparoNave.Dibujar();
                    if (disparoNave.ColisionaCon(ovni))
                    {
                        disparoNave.Activo = false;
                        ovni.Activo = false;
                        marcador.SumarPuntos(50);
                    }
                    for (int i=0; i<30; i++)
                    {
                        if (disparoNave.ColisionaCon(bloque.Enemigos[i]) &&
bloque.Enemigos[i].Activo)
                        {
                            disparoNave.Activo = false;
                            bloque.Enemigos[i].Activo = false;
                            marcador.SumarPuntos(10);
```

```
                        }
                    }
                }

                if (disparoEnemigo != null && disparoEnemigo.Activo)
                {
                    disparoEnemigo.MoverAbajo();
                    disparoEnemigo.Dibujar();
                    if (disparoEnemigo.ColisionaCon(nave))
                    {
                        disparoEnemigo.Activo = false;
                        marcador.RestarVidas();
                        nave.Reset();
                        if (marcador.CuantasVidasQuedan() == 0)
                        {
                            GameOver();
                            finPartida = true;
                        }
                    }
                }

                for (int i=0; i<30; i++)
                {
                    if ((bloque.Enemigos[i].ColisionaCon(nave)) &&
(bloque.Enemigos[i].Activo))
                    {
                        GameOver();
                        finPartida = true;
                    }
                }

                if (Console.KeyAvailable)
                {
                    tecla = Console.ReadKey();
                    if (tecla.Key == ConsoleKey.RightArrow)
                        nave.MoverDerecha();
                    if (tecla.Key == ConsoleKey.LeftArrow)
                        nave.MoverIzquierda();
                    if (tecla.Key == ConsoleKey.Spacebar)
                        disparoNave = nave.Disparar();
                    if (tecla.Key == ConsoleKey.Escape)
                        finPartida = true;
                }
                Thread.Sleep(120);
            } while (!finPartida);
            return marcador.DevolverPuntuacionFinal();
        }
    }
}
```

(7.7.18) En ConsoleInvaders, añade las "torres defensivas", que protegen al jugador y que se van rompiendo poco a poco cada vez que un disparo impacta con ellas.

```
using System;

namespace ConsoleInvaders
{
    class TorresDefensivas : Sprite
    {
        public TorresDefensivas(int x, int y)
```

```
        {
            this.X = x;
            this.Y = y;
            Activo = true;
        }

        protected override string DevolverImagen()
        {
            return "--";
        }

        protected override ConsoleColor DevolverColor()
        {
            return ConsoleColor.Green;
        }

        protected override int DevolverLongitud()
        {
            return 2;
        }

        public bool Activo { get; set; }
    }
}

using System;
using System.Threading;

namespace ConsoleInvaders
{
    class Partida
    {
        ConsoleKeyInfo tecla;
        Nave nave;
        BloqueDeEnemigos bloque;
        Disparo disparoNave;
        Disparo disparoEnemigo;
        Ovni ovni;
        Marcador marcador;
        private bool finPartida;
        Random generador;
        private int numAleatorio;
        TorresDefensivas[] escudos;
        int pausa;


        public Partida()
        {
            nave = new Nave(40, 20);
            bloque = new BloqueDeEnemigos();
            disparoNave = null;
            disparoEnemigo = null;
            ovni = new Ovni();
            marcador = new Marcador();
            finPartida = false;
            pausa = 120;
            generador = new Random();
            escudos = new TorresDefensivas[15];
            for (int i = 0; i < 15; i++)
                escudos[i] = new TorresDefensivas(10 + (i * 4), 19);
        }
```

```
        public void GameOver()
        {
            Console.Clear();
            Console.SetCursorPosition(35, 10);
            Console.ForegroundColor = ConsoleColor.Red;
            Console.Write("GAME OVER");
            Console.CursorVisible = false;
            Console.ReadLine();
        }

        public int Lanzar()
        {
            Console.Clear();
            nave.Dibujar();
            bloque.Dibujar();

            do
            {
                Console.Clear();
                nave.Dibujar();
                bloque.Dibujar();
                bloque.Mover();
                ovni.Mover();
                ovni.Dibujar();
                for (int i = 0; i < 15; i++)
                    if (escudos[i].Activo)
                        escudos[i].Dibujar();
                marcador.ActualizarMarcador();
                numAleatorio = generador.Next(1, 7);
                if (numAleatorio == 3)
                    disparoEnemigo = bloque.Disparar();
                if (disparoNave!=null && disparoNave.Activo)
                {
                    disparoNave.MoverArriba();
                    disparoNave.Dibujar();
                    if (disparoNave.ColisionaCon(ovni))
                    {
                        disparoNave.Activo = false;
                        ovni.Activo = false;
                        marcador.SumarPuntos(50);
                    }
                    for (int i=0; i<30; i++)
                    {
                        if (disparoNave.ColisionaCon(bloque.Enemigos[i]) &&
bloque.Enemigos[i].Activo)
                        {
                            disparoNave.Activo = false;
                            bloque.Enemigos[i].Activo = false;
                            marcador.SumarPuntos(10);
                        }
                    }
                    for (int i = 0; i < 15; i++)
                    {
                        if ((disparoNave.ColisionaCon(escudos[i])) &&
(escudos[i].Activo))
                        {
                            escudos[i].Activo = false;
                            disparoNave.Activo = false;
                        }
                    }
                }
```

```
                if (disparoEnemigo != null && disparoEnemigo.Activo)
                {
                    disparoEnemigo.MoverAbajo();
                    disparoEnemigo.Dibujar();
                    if (disparoEnemigo.ColisionaCon(nave))
                    {
                        disparoEnemigo.Activo = false;
                        marcador.RestarVidas();
                        nave.Reset();
                        if (marcador.CuantasVidasQuedan() == 0)
                        {
                            GameOver();
                            finPartida = true;
                        }
                    }
                    for (int i=0; i<15; i++)
                    {
                        if ((disparoEnemigo.ColisionaCon(escudos[i])) &&
(escudos[i].Activo))
                        {
                            escudos[i].Activo = false;
                            disparoEnemigo.Activo = false;
                        }
                    }
                }

                for (int i=0; i<30; i++)
                {
                    if ((bloque.Enemigos[i].ColisionaCon(nave)) &&
(bloque.Enemigos[i].Activo))
                    {
                        GameOver();
                        finPartida = true;
                    }
                    for (int j=0; j<15; j++)
                    {
                        if ((bloque.Enemigos[i].ColisionaCon(escudos[j])) &&
(bloque.Enemigos[i].Activo) && (escudos[j].Activo))
                        {
                            bloque.Enemigos[i].Activo = false;
                            escudos[j].Activo = false;
                        }
                    }

                }

                if (Console.KeyAvailable)
                {
                    tecla = Console.ReadKey();
                    if (tecla.Key == ConsoleKey.RightArrow)
                        nave.MoverDerecha();
                    if (tecla.Key == ConsoleKey.LeftArrow)
                        nave.MoverIzquierda();
                    if (tecla.Key == ConsoleKey.Spacebar)
                        disparoNave = nave.Disparar();
                    if (tecla.Key == ConsoleKey.Escape)
                        finPartida = true;
                }
                Thread.Sleep(pausa);
            } while (!finPartida);
            return marcador.DevolverPuntuacionFinal();
        }
```

```
    }
}
```

(Extra) Hacer que se incremente el nivel de dificultad, subiendo la velocidad del juego, al eliminar a todos los enemigos.

```
using System;
using System.Collections.Generic;

namespace ConsoleInvaders
{
    class Bienvenida
    {
        ConsoleKeyInfo tecla;
        public bool Salir { get; private set; }

        public void Lanzar(List<int> puntuaciones)
        {
            Console.Clear();
            Console.SetCursorPosition(33, 6);
            Console.ForegroundColor = ConsoleColor.Yellow;
            Console.Write("CONSOLE INVADERS");
            Console.SetCursorPosition(20, 7);
            Console.ForegroundColor = ConsoleColor.White;
            Console.Write("(Pulsa Intro para jugar o ESC para salir)");
            Console.SetCursorPosition(30, 10);
            Console.ForegroundColor = ConsoleColor.Gray;
            Console.Write("MEJORES PUNTUACIONES: ");
            for (int i=0; i<puntuaciones.Count && i<3; i++)
            {
                Console.SetCursorPosition(30, 11+i);
                Console.ForegroundColor = ConsoleColor.Gray;
                Console.Write("{0}º) {1}", i+1, puntuaciones[i]);
            }
            Console.CursorVisible = false;
            tecla = Console.ReadKey();
            if (tecla.Key == ConsoleKey.Escape)
            {
                Salir = true;
            }
        }


    }
}

using System;

namespace ConsoleInvaders
{
    class BloqueDeEnemigos
    {
        int x, y;
        int incremento;
        Disparo disparo;
        Random generador;
        int numAleatorio;

        public BloqueDeEnemigos()
```

```
        {
            Enemigos = new Enemigo[30];
            x = 20;
            y = 10;
            generador = new Random();
            incremento = 1;
            for (int i = 0; i < 10; i++)
                Enemigos[i] = new Enemigo(x + (i * 4), y - 4);
            for (int i = 0; i < 10; i++)
                Enemigos[i + 10] = new Enemigo2(x + (i * 4), y - 2);
            for (int i = 0; i < 10; i++)
                Enemigos[i + 20] = new Enemigo3(x + (i * 4), y);
        }

        public Enemigo[] Enemigos { get; set; }

        public void Dibujar()
        {
            for (int i = 0; i < 30; i++)
            {
                if (Enemigos[i].Activo)
                    Enemigos[i].Dibujar();
            }
        }

        public void Mover()
        {
            x += incremento;
            if (x<=0 || x >= 40)
            {
                y++;
                incremento = -incremento;
            }
            for (int i = 0; i < 10; i++)
            {
                Enemigos[i].X = x + (i * 4);
                Enemigos[i].Y = y-4;
            }

            for (int i = 0; i < 10; i++)
            {
                Enemigos[i + 10].X = x + (i * 4);
                Enemigos[i + 10].Y = y-2;
            }

            for (int i = 0; i < 10; i++)
            {
                Enemigos[i + 20].X = x + (i * 4);
                Enemigos[i + 20].Y = y;
            }
        }

        public Disparo Disparar()
        {
            numAleatorio = generador.Next(1, 30);

            if (disparo == null || !disparo.Activo)
            {
                disparo = new Disparo(Enemigos[numAleatorio].X,
Enemigos[numAleatorio].Y);
            }
            return disparo;
```

```
        }

        public void Reset()
        {
            Enemigos = new Enemigo[30];
            x = 20;
            y = 12;
            for (int i = 0; i < 10; i++)
                Enemigos[i] = new Enemigo(x + (i * 4), y - 4);
            for (int i = 0; i < 10; i++)
                Enemigos[i + 10] = new Enemigo2(x + (i * 4), y - 2);
            for (int i = 0; i < 10; i++)
                Enemigos[i + 20] = new Enemigo3(x + (i * 4), y);
        }
    }
}

using System;

namespace ConsoleInvaders
{
    class Disparo: Sprite
    {
        public Disparo(int x, int y)
        {
            this.X = x+1;
            this.Y = y;
            Activo = true;
        }

        public bool Activo { get; set; }

        protected override string DevolverImagen()
        {
            return "|";
        }

        protected override ConsoleColor DevolverColor()
        {
            return ConsoleColor.Yellow;
        }

        public void MoverArriba()
        {
            Y--;
            if (Y == 0) Activo = false;
        }

        public void MoverAbajo()
        {
            Y++;
            if (Y >= 23) Activo = false;
        }

        protected override int DevolverLongitud()
        {
            return 1;
        }
    }
}

using System;
```

```
namespace ConsoleInvaders
{
    class Enemigo3 : Enemigo
    {
        public Enemigo3(int x, int y): base (x, y)
        {

        }

        protected override string DevolverImagen()
        {
            return ")(";
        }

        protected override ConsoleColor DevolverColor()
        {
            return ConsoleColor.Green;
        }
    }
}
```

```
using System;

namespace ConsoleInvaders
{
    class Enemigo : Sprite
    {
        public bool Activo { get; set; }

        public Enemigo()
        {
            X = 40;
            Y = 10;
            imagen = "][";
            Activo = true;
        }

        public Enemigo(int x, int y)
        {
            this.X = x;
            this.Y = y;
            imagen = "][";
            Activo = true;
        }

        protected override string DevolverImagen()
        {
            return "][";
        }

        protected override ConsoleColor DevolverColor()
        {
            return ConsoleColor.Yellow;
        }

        protected override int DevolverLongitud()
        {
            return 2;
        }
    }
}
```

```
using System;

namespace ConsoleInvaders
{
    class Enemigo2 : Enemigo
    {
        public Enemigo2(int x, int y): base (x, y)
        {

        }

        protected override string DevolverImagen()
        {
            return "}{";
        }

        protected override ConsoleColor DevolverColor()
        {
            return ConsoleColor.Blue;
        }
    }
}

using System;
using System.Collections.Generic;

namespace ConsoleInvaders
{
    class Juego
    {
        Bienvenida bienvenida;
        Partida partida;
        List<int> puntuaciones = new List<int>();

        public Juego()
        {
            bienvenida = new Bienvenida();
        }

        public void Lanzar()
        {
            do
            {
                bienvenida.Lanzar(puntuaciones);
                if (!bienvenida.Salir)
                {
                    partida = new Partida();
                    puntuaciones.Add(partida.Lanzar());
                    puntuaciones.Sort(CompararNumerosDescendiente);
                }
            } while (!bienvenida.Salir);
        }

        private int CompararNumerosDescendiente(int num1, int num2)
        {
            if (num2 > num1)
                return 1;
            else if (num2 < num1)
                return -1;
            else
                return 0;
```

```
        }
    }
}

using System;

namespace ConsoleInvaders
{
    class Marcador
    {
        private int vidas;
        private int score;

        public Marcador()
        {
            vidas = 3;
            score = 0;
        }

        public void ActualizarMarcador()
        {
            Console.SetCursorPosition(3, 1);
            Console.ForegroundColor = ConsoleColor.White;
            Console.Write("VIDAS: {0}\tSCORE: {1}", vidas, score);
            Console.CursorVisible = false;
        }

        public int SumarPuntos (int puntos)
        {
            return score += puntos;
        }

        public void RestarVidas()
        {
            vidas--;
        }

        public int CuantasVidasQuedan()
        {
            return vidas;
        }

        public int DevolverPuntuacionFinal()
        {
            return score;
        }

        public void VidaExtra()
        {
            vidas++;
        }
    }
}

using System;

namespace ConsoleInvaders
{
    class Nave : Sprite
    {
        Disparo disparo;
```

```
        public Nave(int x, int y)
        {
            this.X = x;
            this.Y = y;
            imagen = "/\\";
        }

        public Nave() : this (40, 20)
        {

        }

        public void MoverDerecha()
        {
            X += 10;
            if (X >= 76) X = 76;
        }

        public void MoverIzquierda()
        {
            X -= 10;
            if (X <= 0) X = 0;
        }

        protected override string DevolverImagen()
        {
            return "<->";
        }

        protected override ConsoleColor DevolverColor()
        {
            return ConsoleColor.White;
        }

        public Disparo Disparar()
        {
            if (disparo == null || !disparo.Activo)
            {
                disparo = new Disparo(X, Y);
            }
            return disparo;
        }

        protected override int DevolverLongitud()
        {
            return 3;
        }

        public void Reset()
        {
            X = 40;
            Y = 20;
        }
    }
}

using System;

namespace ConsoleInvaders
{
    class Ovni : Sprite
    {
```

```
        Random generador;
        int aleatorio;

        public Ovni()
        {
            X = 0;
            Y = 4;
            Activo = false;
            generador = new Random();
        }

        public bool Activo { get; set; }

        protected override string DevolverImagen()
        {
            if (Activo) return "(||)";
            else return "";
        }

        protected override ConsoleColor DevolverColor()
        {
            return ConsoleColor.Yellow;
        }

        public void Mover()
        {
            if (!Activo)
            {
                aleatorio = generador.Next(1, 61);
                if (aleatorio == 2)
                {
                    Activo = true;
                    X = 0;
                }
            }
            else
            {
                X++;
                if (X >= 76) Activo = false;
            }
        }

        protected override int DevolverLongitud()
        {
            return 4;
        }
    }
}

using System;
using System.Threading;

namespace ConsoleInvaders
{
    class Partida
    {
        ConsoleKeyInfo tecla;
        Nave nave;
        BloqueDeEnemigos bloque;
        Disparo disparoNave;
        Disparo disparoEnemigo;
        Ovni ovni;
```

```
Marcador marcador;
private bool finPartida;
Random generador;
private int numAleatorio;
TorresDefensivas[] escudos;
int pausa;
int contadorEnemigosEliminados;

public Partida()
{
    nave = new Nave(40, 20);
    bloque = new BloqueDeEnemigos();
    disparoNave = null;
    disparoEnemigo = null;
    ovni = new Ovni();
    marcador = new Marcador();
    finPartida = false;
    pausa = 120;
    contadorEnemigosEliminados = 0;
    generador = new Random();
    escudos = new TorresDefensivas[15];
    for (int i = 0; i < 15; i++)
        escudos[i] = new TorresDefensivas(10 + (i * 4), 19);
}

public void GameOver()
{
    Console.Clear();
    Console.SetCursorPosition(35, 10);
    Console.ForegroundColor = ConsoleColor.Red;
    Console.Write("GAME OVER");
    Console.CursorVisible = false;
    Console.ReadLine();
}

public int Lanzar()
{
    Console.Clear();
    nave.Dibujar();
    bloque.Dibujar();

    do
    {
        Console.Clear();
        nave.Dibujar();
        bloque.Dibujar();
        bloque.Mover();
        ovni.Mover();
        ovni.Dibujar();
        for (int i = 0; i < 15; i++)
            if (escudos[i].Activo)
                escudos[i].Dibujar();
        marcador.ActualizarMarcador();
        numAleatorio = generador.Next(1, 7);
        if (numAleatorio == 3)
            disparoEnemigo = bloque.Disparar();
        if (disparoNave!=null && disparoNave.Activo)
        {
            disparoNave.MoverArriba();
            disparoNave.Dibujar();
            if (disparoNave.ColisionaCon(ovni))
            {
```

```
                        disparoNave.Activo = false;
                        ovni.Activo = false;
                        marcador.SumarPuntos(50);
                    }
                    for (int i=0; i<30; i++)
                    {
                        if (disparoNave.ColisionaCon(bloque.Enemigos[i]) &&
bloque.Enemigos[i].Activo)
                        {
                            disparoNave.Activo = false;
                            bloque.Enemigos[i].Activo = false;
                            marcador.SumarPuntos(10);
                            contadorEnemigosEliminados++;
                        }
                    }
                    for (int i = 0; i < 15; i++)
                    {
                        if ((disparoNave.ColisionaCon(escudos[i])) &&
(escudos[i].Activo))
                        {
                            escudos[i].Activo = false;
                            disparoNave.Activo = false;
                        }
                    }
                }

                if (disparoEnemigo != null && disparoEnemigo.Activo)
                {
                    disparoEnemigo.MoverAbajo();
                    disparoEnemigo.Dibujar();
                    if (disparoEnemigo.ColisionaCon(nave))
                    {
                        disparoEnemigo.Activo = false;
                        marcador.RestarVidas();
                        nave.Reset();
                        if (marcador.CuantasVidasQuedan() == 0)
                        {
                            GameOver();
                            finPartida = true;
                        }
                    }
                    for (int i=0; i<15; i++)
                    {
                        if ((disparoEnemigo.ColisionaCon(escudos[i])) &&
(escudos[i].Activo))
                        {
                            escudos[i].Activo = false;
                            disparoEnemigo.Activo = false;
                        }
                    }
                }

                for (int i=0; i<30; i++)
                {
                    if ((bloque.Enemigos[i].ColisionaCon(nave)) &&
(bloque.Enemigos[i].Activo))
                    {
                        GameOver();
                        finPartida = true;
                    }
                    for (int j=0; j<15; j++)
                    {
```

```
                        if ((bloque.Enemigos[i].ColisionaCon(escudos[j])) &&
(bloque.Enemigos[i].Activo) && (escudos[j].Activo))
                        {
                            bloque.Enemigos[i].Activo = false;
                            escudos[j].Activo = false;
                            contadorEnemigosEliminados++;
                        }
                    }
                    if ((bloque.Enemigos[i].Y ==
23)&&(bloque.Enemigos[i].Activo))
                    {
                        GameOver();
                        finPartida = true;
                    }
                }
                if (contadorEnemigosEliminados == 30)
                {
                    bloque.Reset();
                    marcador.VidaExtra();
                    pausa -= 20;
                    if (pausa <= 40) pausa = 40;
                    contadorEnemigosEliminados = 0;
                }

                if (Console.KeyAvailable)
                {
                    tecla = Console.ReadKey();
                    if (tecla.Key == ConsoleKey.RightArrow)
                        nave.MoverDerecha();
                    if (tecla.Key == ConsoleKey.LeftArrow)
                        nave.MoverIzquierda();
                    if (tecla.Key == ConsoleKey.Spacebar)
                        disparoNave = nave.Disparar();
                    if (tecla.Key == ConsoleKey.Escape)
                        finPartida = true;
                }
                Thread.Sleep(pausa);
            } while (!finPartida);
            return marcador.DevolverPuntuacionFinal();
        }
    }
}

using System;

namespace ConsoleInvaders
{
    class Program
    {
        static void Main()
        {
            Juego juego = new Juego();
            juego.Lanzar();
        }
    }
}

using System;

namespace ConsoleInvaders
{
    abstract class Sprite
```

```
    {
        protected string imagen;

        public int X { get; set; }

        public int Y { get; set; }

        public void Dibujar()
        {
            Console.SetCursorPosition(X, Y);
            Console.ForegroundColor = DevolverColor();
            Console.Write(DevolverImagen());
            Console.CursorVisible = false;
        }

        protected abstract string DevolverImagen();

        protected abstract ConsoleColor DevolverColor();

        protected abstract int DevolverLongitud();

        public bool ColisionaCon (Sprite sprite)
        {
            if (this.Y != sprite.Y)
                return false;
            else
            {
                for (int i = 0; i < this.DevolverLongitud(); i++)
                {
                    for (int j = 0; j < sprite.DevolverLongitud(); j++)
                    {
                        if ((this.X + i) == (sprite.X + j))
                            return true;
                    }
                }
            }
            return false;
        }
    }
}

using System;

namespace ConsoleInvaders
{
    class TorresDefensivas : Sprite
    {
        public TorresDefensivas(int x, int y)
        {
            this.X = x;
            this.Y = y;
            Activo = true;
        }

        protected override string DevolverImagen()
        {
            return "--";
        }

        protected override ConsoleColor DevolverColor()
        {
            return ConsoleColor.Green;
```

```
        }

        protected override int DevolverLongitud()
        {
            return 2;
        }

        public bool Activo { get; set; }
    }
}
```