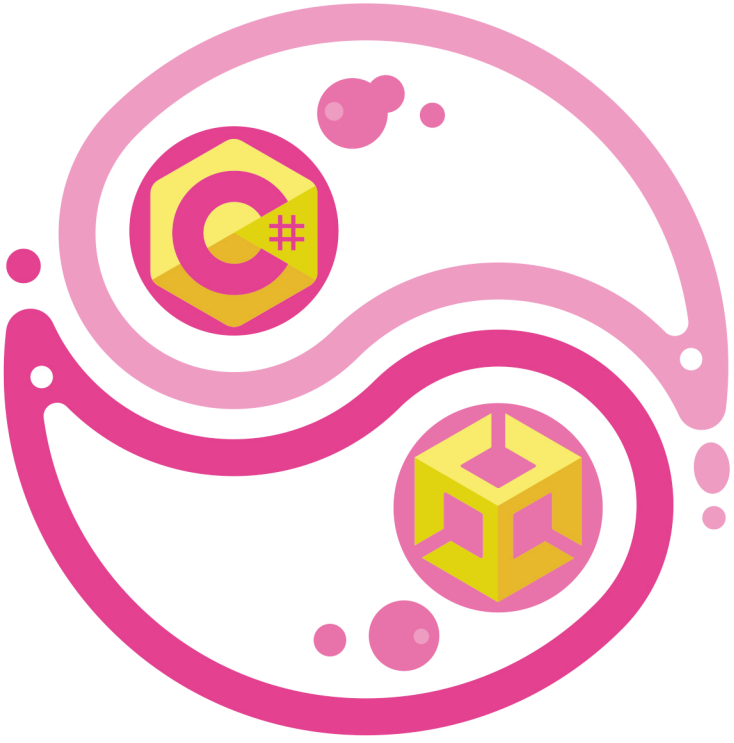
Informática

C# PARA UNITY

100 EJERCICIOS PRÁCTICOS

Cómo mejorar tu kungfú



Jorge García Colmenar

FC EDITORIAL

edU®

Informática

C# PARA UNITY

100 EJERCICIOS PRÁCTICOS

Cómo mejorar tu kungfú

Jorge García Colmenar

FC EDITORIAL

edU®
Conocimiento a su alcance

Bogotá - México, D.F.

García Colmenar, Jorge

C# para Unity. 100 ejercicios prácticos / Jorge García Colmenar -- Bogotá : Ediciones de la U, 2024
392 p. ; 24 cm.
ISBN 978-958-792-715-3 e-ISBN 978-958-792-716-0
1. Informática 2. Lenguaje de programación I. Tít.
621.39 ed. Thema: UFC

Edición original publicada por © Fundación Confemetal (España)
Edición autorizada a Ediciones de la U de Colombia

Área: Informática
Primera edición: Bogotá, Colombia, junio de 2024
ISBN. 978-958-792-715-3

Coordinación editorial: Adriana Gutiérrez M.
Carátula: Ediciones de la U
Impresión: DGP Editores SAS
Calle 63 #70D-34, Pbx. (+57) 601 7217756

Impreso y hecho en Colombia
Printed and made in Colombia

ÍNDICE

SEGUNDA PARTE

QUINTA PARTE

SOBRE EL AUTOR

Jorge García Colmenar ha trabajado como programador en importantes empresas (RD Sistemas, Grupo ZED, etc.), implementando bases de datos y algoritmos para juegos por SMS. Además, como autónomo, desarrolló aplicaciones y videojuegos de *smartphone* para clientes como José Luis Moreno o las Hermanas Hospitalarias.

Desde 2013 ha estado permanentemente formando en el desarrollo de videojuegos, habiendo sido profesor y coordinador de grado en escuelas como CES o Alfonso X el Sabio, e impartiendo cursos de formación para ADECCO. Principalmente enseña las asignaturas de programación y diseño de videojuegos, aunque también ha desarrollado conferencias y seminarios en diversos eventos como el Madrid Games Week, Valencia Indie Summit o ESADA.

Jorge es el desarrollador principal de cKolmos Narrative, un estudio *indie* de videojuegos, con el que ha publicado juegos comerciales en ocho idiomas y en plataformas como PlayStation, Nintendo Switch o XBOX, además de haberlo hecho en tiendas de juegos como Steam o Meta Quest, incluyendo juegos de realidad virtual.

Entre sus aficiones está la pasión por el go, un juego muy relacionado con la inteligencia artificial, tema de su interés y en el que tiene estudios de máster por la UNIR.

INTRODUCCIÓN

Estimado estudiante:

Me complace presentarte ***100 ejercicios en C# para Unity***. Este libro es una recopilación esencial de ejercicios para aquellos que desean mejorar sus habilidades en la programación de videojuegos utilizando el lenguaje de programación C# y el *software* Unity.

Como profesor de programación y desarrollo de videojuegos, he tenido la oportunidad de enseñar a muchos jóvenes creadores a lo largo de los años. Muchos de mis estudiantes han expresado interés en mejorar sus habilidades en la programación de videojuegos, pero no saben por dónde empezar o se pierden en el bosque de manuales de C# no específicos.

Durante años he ido creando y probando ejercicios para mis clases. En este libro he recopilado los más eficaces y divertidos y he descartado los aburridos o los que son excesivamente complejos. Están diseñados para ayudar de manera sistemática y eficaz a los nuevos desarrolladores a mejorar sus habilidades de programación de videojuegos en C# y Unity.

El libro presenta una serie de ejercicios que abarcan una amplia gama de conceptos, desde la creación de objetos en la escena hasta la implementación de mecánicas de juego complejas.

Gracias a los ejercicios, tendrás la oportunidad de aplicar los conceptos teóricos que hayas aprendido a la creación de juegos en Unity, mejorando así tus habilidades en la programación de videojuegos y tu capacidad para crear juegos emocionantes y dinámicos.

Mi objetivo es proporcionar una guía **práctica** y **efectiva** para mejorar tu kungfú en la programación de videojuegos porque, para mí, la programación de videojuegos es como un arte marcial. Descubrir, aprender y practicar cada técnica sirve para comprendernos mejor a nosotros mismos y para realizar proezas asombrosas.

Pero antes son precisos dos matices...

Cómo no usar este libro

100 ejercicios en C# para Unity no es un manual de programación ni un curso para empezar a programar. Necesitarás aprender los conceptos teóricos por tu cuenta antes de comenzar con este libro. Sin embargo, la información está estructurada para que puedas usarlo a la vez que estudias programación. No he dudado en incluir un montón de explicaciones ni tampoco en repasar conceptos que seguramente hayas estudiado, o que quizá no conozcas aún. Pero la información está estructurada con base en los ejercicios que, aunque siguen un orden parecido a aquel en el que se estudia el código, pueden variar mucho. Consulta las secciones y el índice para saber cuáles son los ejercicios más apropiados para ti y altera el orden como creas conveniente. Si tienes dudas, sigue el orden del índice, ya que está pensado para un aprendizaje con dificultad progresiva.

¿Cómo usar este libro?

Este libro se estructura en un listado de ejercicios, cada uno de ellos con la manera de llegar a su solución paso a paso y con comentarios sobre el método utilizado. La mejor forma de abordarlos es leer el enunciado y tratar de resolver el ejercicio en Unity. Los ejercicios están diseñados para que sea indiferente la versión de Unity que uses. Cada uno tiene un tiempo estimado de resolución.

Se da por hecho que no sepas resolver el ejercicio sin buscar documentación en la que apoyarte. Una recomendación que puedo hacerte es que, si no tienes ni idea de por dónde comenzar a resolverlo, lo dividas en partes pequeñas y comiences a solucionar lo que puedas. No tengas miedo de buscar respuestas en internet, mirar la documentación de C# y Unity, etc. De lo que se trata es de aprender y mejorar.

Es muy importante que no te frustres si no consigues resolver un ejercicio. Lo principal es que trates de solucionarlo por tu cuenta, sin mirar la respuesta antes de que se consuma todo el tiempo asignado. Gasta todo el tiempo del ejercicio en pensar formas de abordar el problema e investigar

alternativas. Cuando el tiempo acabe, tanto si lo has conseguido como si no, aplica la solución ofrecida y compárala con la tuya. Esta es la mejor forma de aprender a programar videojuegos. Si es doloroso o agobiante, piensa en el esfuerzo que le llevó a Son Goku, Aang o a la novia de Kill Bill mejorar sus habilidades, y nunca te des por vencido.

Apunte para profesores

Si eres profesor de programación y usas Unity, espero que encuentres en este libro una ayuda para tus clases. Diseñar y probar buenos ejercicios, como sabes, es un trabajo arduo y poco reconocido. Con este compendio podrás diseñar rutas de aprendizaje en tus clases o sorprender a tus alumnos con problemas que no se esperan. Muchos ejercicios plantean una forma básica en sus soluciones, pensada para desarrolladores que aprenden y no para *seniors* con muchas horas de vuelo, por lo que, en ocasiones, si hay una forma más eficiente pero más compleja de resolver el ejercicio, se menciona, pero no se profundiza en ella.

¡Vamos a empezar!

PRIMERA PARTE

En esta primera parte nos centraremos en los ejercicios más básicos para acostumbrarnos al uso de dos programas: el editor de código y el motor de videojuegos. El **editor de código** es el *software* donde escribimos nuestros *scripts*. El **motor de videojuegos** es donde se renderizan los gráficos y se ejecutan estos *scripts* para conseguir funcionalidad. Nuestro motor de videojuegos será **Unity** y nuestro editor de código podría ser cualquiera, hasta un bloc de notas, ya que el código no es más que texto.

NOTAS
(puedes tomar anotaciones aquí)

Es importante destacar que el entorno de desarrollo integrado (IDE, por sus siglas en inglés) **Visual Studio** tiene integración con Unity y se instala a la vez, por lo que es el más usado como editor de código. Su resaltado de sintaxis y ayudas al programador son muy apreciadas y mejoran el tiempo de desarrollo, facilitando mucho el proceso de trabajo.

CIEN EJERCICIOS BÁSICOS DE PROGRAMACIÓN

Vamos a comenzar con un calentamiento. Los ejercicios listados a continuación solo requieren el uso de un editor de código asociado a Unity, que en tus clases de programación te habrán enseñado a utilizar.

Cada uno de los siguientes diez ejercicios puede hacerse en un *script* independiente, asociado a un objeto cualquiera de la escena de Unity. No vamos a usar aún ningún componente ni nada relacionado con videojuegos, simplemente vamos a practicar un poco con el código. Ten localizada la consola de Unity para ver los resultados de los ejercicios y comprobar si todo ha ido bien.

NOTAS

La dificultad de este calentamiento es **incremental**, siendo los primeros ejercicios muy fáciles y los últimos, algo más complejos. El ejercicio 10 es el jefe de área. Si crees que ya dominas bien la programación en C# puedes tratar de resolverlo y saltarte los nueve primeros, pero si fallas deberás comenzar desde el principio. ¡Nada de trampas!

1. Crea una nueva variable llamada `velocidad` y asígnale el valor de `5.0F`. Imprime su valor por consola.

Tiempo: 5 minutos

```
using UnityEngine;
public class EjemploVelocidad : MonoBehaviour
{
  //  Declaración de la variable velocidad con valor
  //  inicial de 5.0F
  public float velocidad = 5.0F;

  void Start()
  {
    //  Imprime por consola el valor de la variable
    //  velocidad
    Debug.Log("La velocidad es: " + velocidad);
  }
}
```

Este *script* en C# para Unity comienza importando el espacio de nombres `UnityEngine`, esencial para acceder a las funciones específicas del motor de juego. La clase `EjemploVelocidad` se declara heredando de `MonoBehaviour`, lo que permite que nuestro código pueda utilizar la funcionalidad de la librería de Unity.

NOTAS

Dentro de la clase, declaramos una variable pública llamada `velocidad` con un valor inicial de `5.0F`. Las variables marcadas como públicas se pueden ver en el inspector de Unity y permiten modificar sus valores iniciales.

El método `Start()` se emplea para ejecutar código al inicio del juego. En este caso, utilizamos `Debug.Log` para imprimir en la consola un mensaje informativo que incluye el valor de la variable. Esto sirve para depurar el código durante el desarrollo del juego, encontrar *bugs* o revisar el comportamiento del *script*, aunque un método más avanzado implicaría el uso de depuradores integrados en IDE como **Visual Studio**.

El código está estructurado de forma que la declaración de variables se encuentra al inicio de la clase, antes de la definición del método `Start()`.

2. Utiliza dos variables públicas de tipo `float` para mostrar por consola sus valores sumados, restados, multiplicados y divididos.

Tiempo: 5 minutos

```
using UnityEngine;
public class OperacionesMatematicas : MonoBehaviour
{
  //  Dos variables públicas de tipo float
  public float numero1 = 10.0F;
  public float numero2 = 5.0F;

  void Start()
  {
    //  Operación de suma
    float suma = numero1 + numero2;
    Debug.Log("Suma: " + suma);
```

NOTAS

```
        //  Operación de resta
        float resta = numero1 - numero2;
        Debug.Log("Resta: " + resta);
        //  Operación de multiplicación
        float multiplicacion = numero1 * numero2;
        Debug.Log("Multiplicación: " + multiplicacion);
        //  Operación de división (evitando división
        //  entre cero)
        if (numero2 != 0)
        {
          float division = numero1 / numero2;
          Debug.Log("División: " + division);
        }
        else
        {
          Debug.Log("No se puede dividir entre cero.");
        }
    }
}
```

Estamos usando dos variables públicas de tipo `float`: `numero1` y `numero2`. Las variables de este tipo están pensadas para hacer operaciones matemáticas en coma flotante (con decimales).

Luego, para cada una de las operaciones que queremos hacer, declaramos una nueva variable float que almacenará el resultado.

La primera operación es la suma, donde las dos variables se suman y su resultado se guarda en la variable `suma`.

A continuación, se ejecuta la operación de resta, donde el valor de `numero2` es restado de `numero1`. El resultado se almacena en la variable `resta`, y se imprime en la consola el mensaje «Resta: [resultado de la resta]».

NOTAS

Posteriormente, se lleva a cabo la operación de multiplicación, donde los valores de `numero1` y `numero2` se multiplican entre sí. El resultado se almacena en la variable `multiplicacion`, y se imprime en la consola el mensaje «Multiplicación: [resultado de la multiplicación]».

Finalmente, se aborda la operación de división. Antes de realizar la división, se verifica si `numero2` es diferente de cero para evitar divisiones entre cero. En caso afirmativo, se procede con la operación y el resultado se almacena en la variable `division`. Se imprime en la consola el mensaje «División: [resultado de la división]». Si `numero2` es cero, se imprime el mensaje «No se puede dividir entre cero».

3. Crea dos variables privadas de tipo `string`, concaténalas y muestra el resultado por consola.

Tiempo: 5 minutos

```
using UnityEngine;
public class ConcatenacionStrings : MonoBehaviour
{
  //  Dos variables privadas de tipo string
  private string cadena1 = “Hola, “;
  private string cadena2 = “mundo!”;

  void Start()
  {
    //  Operación de concatenación
    string resultadoConcatenacion = cadena1 +
    cadena2;
    //  Muestra el resultado por consola
    Debug.Log(resultadoConcatenacion);
  }
}
```

NOTAS

En este caso declaramos dos variables de tipo `string`: `cadena1` y `cadena2`. Al ser privadas, estas variables solo son accesibles dentro del ámbito de la clase en la que se definen. Asignamos algo de texto a cada variable.

A continuación realizamos la operación de concatenación, donde se combinan los valores de `cadena1` y `cadena2` en una nueva variable llamada `resultadoConcatenacion`. Esta variable ahora contiene la unión de ambas cadenas de texto.

Finalmente, utilizando `Debug.Log`, imprimimos en la consola de Unity el resultado de la concatenación para ver la cadena concatenada.

Este ejercicio sirve para demostrar **cómo podemos combinar cadenas de caracteres**.

4. Crea una variable privada booleana y dos variables públicas `int`. Si al ejecutar el juego la suma de las variables `int` es par, la booleana debe ser `true`. Muestra su valor por consola.

 **Tiempo:** 5 minutos

```
using UnityEngine;
public class VerificacionSuma : MonoBehaviour
{
    //  Variables privadas
    private bool esSumaPar;
    //  Variables públicas
    public int numero1 = 5;
    public int numero2 = 7;
```

```
    void Start()
    {
      //  Calcula la suma de las variables públicas
      int suma = numero1 + numero2;
      //  Verifica si la suma es par
      esSumaPar = (suma % 2) == 0;
      //  Muestra el resultado por consola
      Debug.Log("¿La suma es par? " + esSumaPar);
    }
}
```

NOTAS

En el método `Start()`, se realiza la suma de las dos variables enteras. Luego, mediante una operación de módulo (`%`), se verifica si la suma es divisible entre `2`, lo que indica si es, o no, un número par. El resultado de esta comprobación se almacena en la variable booleana `esSumaPar`. Para realizar esta comprobación hemos utilizado el operador de comparación. Normalmente solemos verlo dentro de una condición (un `if`), pero se puede usar también de esta manera. Devolverá `true` o `false`, como siempre, y ese valor se asigna a la variable booleana.

Finalmente, se utiliza `Debug.Log` para imprimir el resultado de la verificación en la consola de Unity. El mensaje indica si la suma de las variables enteras es un número par o no lo es. Este proceso permite una visualización rápida y eficiente de la condición booleana resultante.