Head First C#

Fourth Edition

Andrew Stellman
Jennifer Greene

Beijing • Boston • Farnham • Sebastopol • Tokyo

O'REILLY®

Head First C#

Fourth Edition

by Andrew Stellman and Jennifer Greene

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly Media books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://oreilly.com*). For more information, contact our corporate/institutional sales department: (800) 998-9938 or *corporate@oreilly.com*.

Series Creators:	Kathy Sierra, Bert Bates
Cover Designer:	Ellie Volckhausen
Brain Image on Spine:	Eric Freeman
Editors:	Nicole Taché, Amanda Quinn
Proofreader:	Rachel Head
Indexer:	Potomac Indexing, LLC
Illustrator:	Jose Marzan
Page Viewers:	Greta the miniature bull terrier and Samosa the Pomeranian

Printing History:

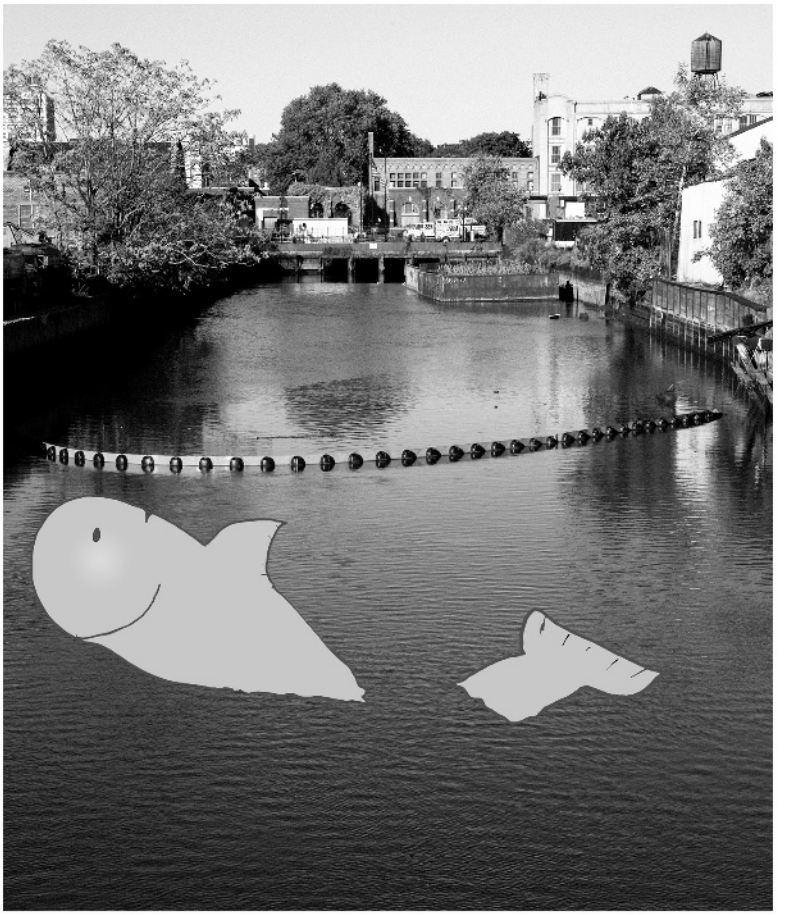
November 2007: First Edition
May 2010: Second Edition
August 2013: Third Edition
December 2020: Fourth Edition

No bees, space aliens, or comic book heroes were harmed in the making of this book.

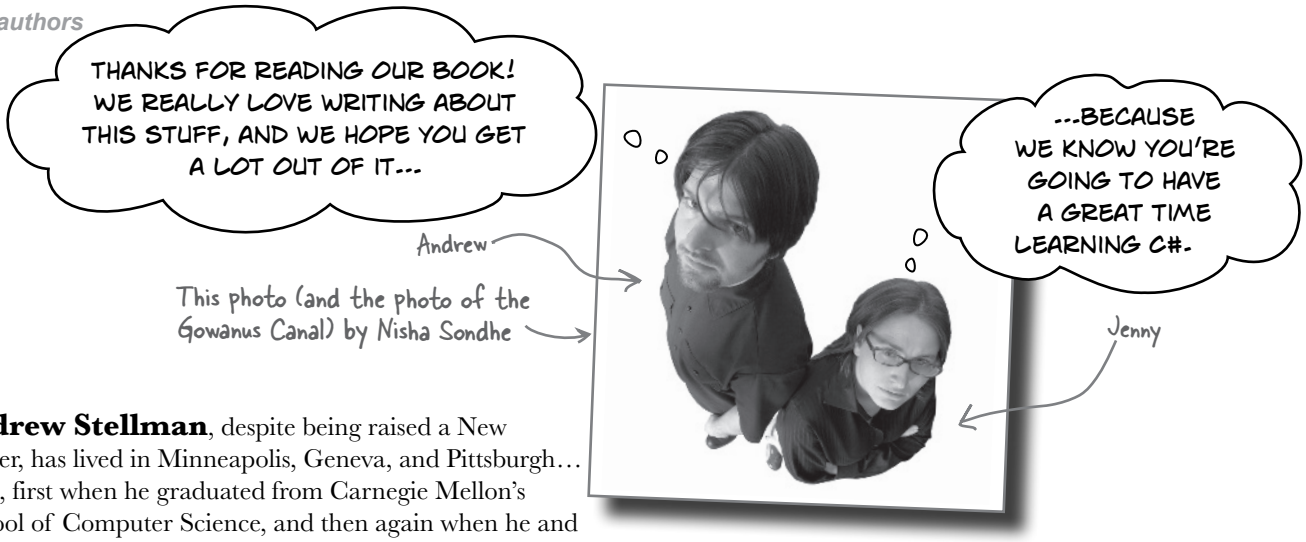
ISBN: 978-1-491-97670-8

[LSI] [2020-12-18]

This book is dedicated to the loving memory of Sludgie the Whale,
who swam to Brooklyn on April 17, 2007.



You were only in our canal for a day,
but you'll be in our hearts forever.



Andrew Stellman, despite being raised a New Yorker, has lived in Minneapolis, Geneva, and Pittsburgh… *twice*, first when he graduated from Carnegie Mellon's School of Computer Science, and then again when he and Jenny were starting their consulting business and writing their first book for O'Reilly.

Andrew's first job after college was building software at a record company, EMI-Capitol Records—which actually made sense, as he went to LaGuardia High School of Music & Art and the Performing Arts to study cello and jazz bass guitar. He and Jenny first worked together at a company on Wall Street that built financial software, where he was managing a team of programmers. Over the years he's been a vice president at a major investment bank, architected large-scale real-time backend systems, managed large international software teams, and consulted for companies, schools, and organizations, including Microsoft, the National Bureau of Economic Research, and MIT. He's had the privilege of working with some pretty amazing programmers during that time, and likes to think that he's learned a few things from them.

When he's not writing books, Andrew keeps himself busy writing useless (but fun) software, playing (and making) both music and video games, practicing krav maga, tai chi, and aikido, and owning a crazy Pomeranian.

Jennifer Greene studied philosophy in college but, like everyone else in the field, couldn't find a job doing it. Luckily, she's a great software engineer, so she started out working at an online service, and that's the first time she really got a good sense of what good software development looked like.

She moved to New York in 1998 to work on software quality at a financial software company. She's managed teams of developers, testers, and PMs on software projects in media and finance since then.

Jenny has traveled all over the world to work with different software teams and build all kinds of cool projects.

She loves traveling, watching Bollywood movies, reading the occasional comic book, playing video games, and hanging out with her huge Siberian cat, Sascha, and her miniature bull terrier, Greta.

Jenny and Andrew have been building software and writing about software engineering together since they first met in 1998. Their first book, **Applied Software Project Management**, was published by O'Reilly in 2005. Other Stellman and Greene books for O'Reilly include **Beautiful Teams** (2009), **Learning Agile** (2014), and their first book in the Head First series, **Head First PMP** (2007), now in its fourth edition.

They founded Stellman & Greene Consulting in 2003 to build a really neat software project for scientists studying herbicide exposure in Vietnam vets. In addition to building software and writing books, they've consulted for companies and spoken at conferences and meetings of software engineers, architects, and project managers.

Learn more about them on their website, **Building Better Software**: https://www.stellman-greene.com.

Follow **@AndrewStellman** and **@JennyGreene** on Twitter

☮❤👾 Jenny and Andrew

Table of Contents (the summary)

Let's add some excitement to the game! The time elapsed since the game started will appear at the bottom of the window, constantly going up, and only stopping after the last animal is matched.

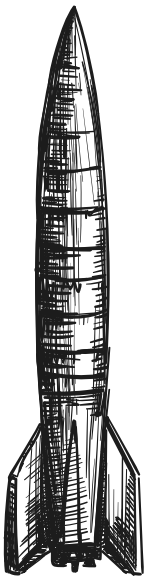


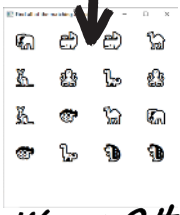
Table of Contents (the real thing)

Intro

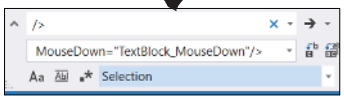
Your brain on C#. You're sitting around trying to *learn* something, but your *brain* keeps telling you all that learning *isn't important*. Your brain's saying, "Better leave room for more important things, like which wild animals to avoid and whether nude archery is a bad idea." So how *do* you trick your brain into thinking that your life really depends on learning C#?

CREATE THE PROJECT

DESIGN THE WINDOW



WRITE C# CODE



HANDLE MOUSE CLICKS

ADD A GAME TIMER

1 start building with C#

Build something great...fast!

Want to build great apps…right now?

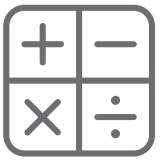
With C#, you've got a **modern programming language** and a **valuable tool** at your fingertips. And with **Visual Studio**, you've got an **amazing development environment** with highly intuitive features that make coding as easy as possible. Not only is Visual Studio a great tool for writing code, it's also a **really valuable learning tool** for exploring C#. Sound appealing? Turn the page, and let's get coding.

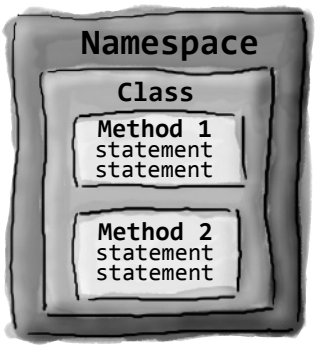
dive into C#

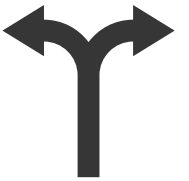
2 Statements, classes, and code

You're not just an IDE user. You're a developer.

You can get a lot of work done using the IDE, but there's only so far it can take you. Visual Studio is one of the most advanced software development tools ever made, but a **powerful IDE** is only the beginning. It's time to **dig in to C# code**: how it's structured, how it works, and how you can take control of it…because there's no limit to what you can get your apps to do.



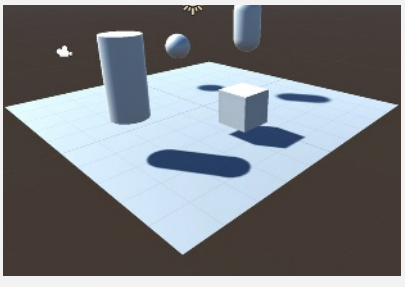


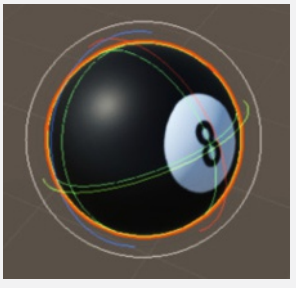


Unity Lab 1

Explore C# with Unity

Welcome to your first **Head First C# Unity Lab**. Writing code is a skill, and like any other skill, getting better at it takes **practice and experimentation**. Unity will be a really valuable tool for that. In this lab, you can begin practicing what you've learned about C# in Chapters 1 and 2.



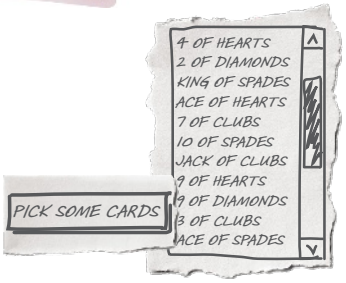


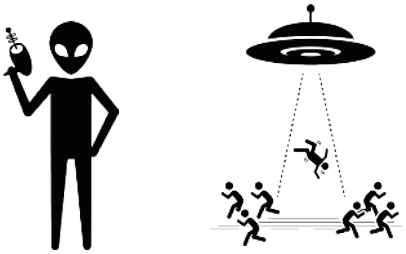
objects...get oriented!

3 Making code make sense

Every program you write solves a problem.

When you're building a program, it's always a good idea to start by thinking about what *problem* your program's supposed to solve. That's why **objects** are really useful. They let you structure your code based on the problem it's solving so that you can spend your time *thinking about the problem* you need to work on rather than getting bogged down in the mechanics of writing code. When you use objects right—and really put some thought into how you design them—you end up with code that's *intuitive* to write, and easy to read and change.

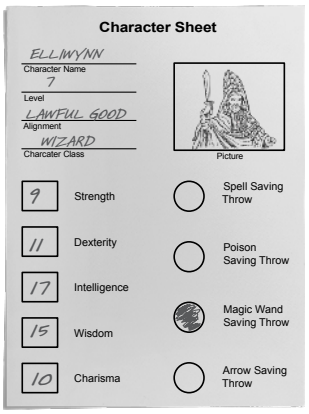




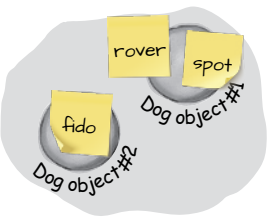
types and references

4 Getting the reference

What would your apps be without data? Think about it for a minute. Without data, your programs are…well, it's actually hard to imagine writing code without data. You need **information** from your users, and you use that to look up or produce new information to give back to them. In fact, almost everything you do in programming involves **working with data** in one way or another. In this chapter, you'll learn the ins and outs of C#'s **data types** and **references**, see how to work with data in your program, and even learn a few more things about **objects** (*guess what…objects are data, too!*).



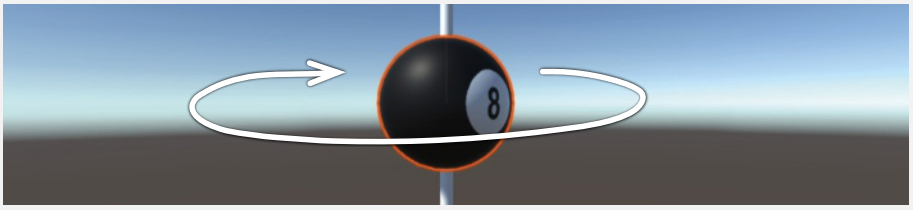
Creating a reference is like writing a name on a sticky note and sticking it to the object. You're using it to label an object so you can refer to it later.



Unity Lab 2

Write C# Code for Unity

Unity isn't *just* a powerful, cross-platform engine and editor for building 2D and 3D games and simulations. It's also a **great way to get practice writing C# code**. In this lab, you'll get more practice writing C# code for a project in Unity.

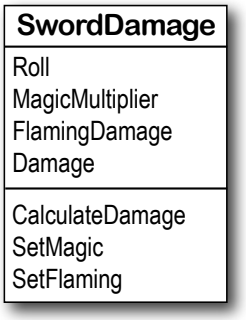


encapsulation

5 Keep your privates...private

Ever wished for a little more privacy?

Sometimes your objects feel the same way. Just like you don't want anybody you don't trust reading your journal or paging through your bank statements, good objects don't let ***other*** objects go poking around their fields. In this chapter, you're going to learn about the power of **encapsulation**, a way of programming that helps you make code that's flexible, easy to use, and difficult to misuse. You'll **make your objects' data private**, and add **properties** to protect how that data is accessed.



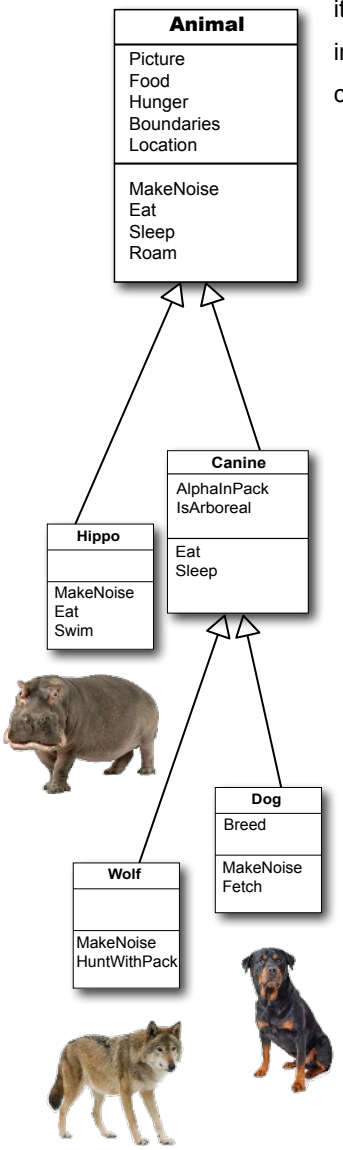

```
RealName: "Herb Jones"

Alias: "Dash Martin"

Password: "the crow flies at midnight"
```

inheritance

6 Your object's family tree

Sometimes you *DO* want to be just like your parents.

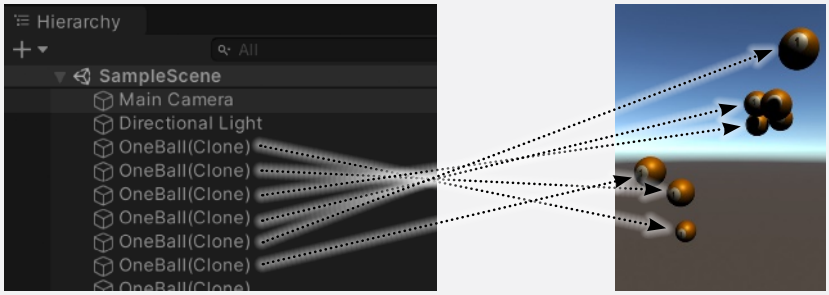
Ever run across a class that ***almost*** does exactly what you want ***your*** class to do? Found yourself thinking that if you could just ***change a few things***, that class would be perfect? With **inheritance**, you can **extend** an existing class so your new class gets all of its behavior—with the **flexibility** to make changes to that behavior so you can tailor it however you want. Inheritance is one of the most powerful concepts and techniques in the C# language: with it you'll **avoid duplicate code**, **model the real world** more closely, and end up with apps that are **easier to maintain** and **less prone to bugs**.



Unity Lab 3

GameObject Instances

C# is an object-oriented language, and since these Head First C# Unity Labs are all **about getting practice writing C# code**, it makes sense that these labs will focus on creating objects.

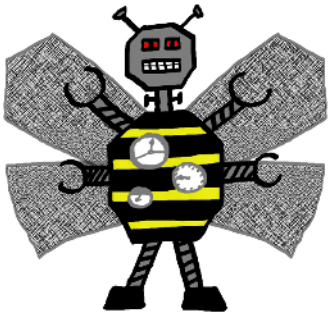


interfaces, casting, and "is"

7 Making classes keep their promises

Actions speak louder than words.

Sometimes you need to group your objects together based on the **things they can do** rather than the classes they inherit from. That's where **interfaces** come in—they let you work with any class that can do the job. But with **great power comes great responsibility**, and any class that implements an interface must promise to **fulfill all of its obligations**…or the compiler will break its kneecaps, see?



enums and collections

8 Organizing your data

When it rains, it pours.

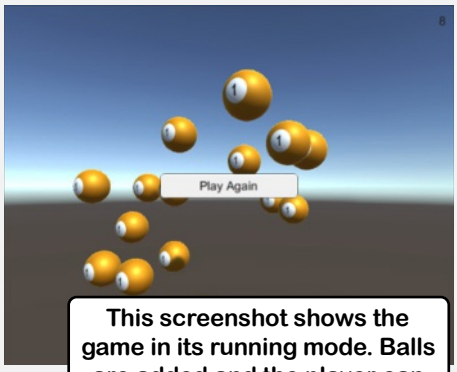
In the real world, you don't receive your data in tiny little bits and pieces. No, your data's going to come at you in **loads, piles, and bunches**. You'll need some pretty powerful tools to organize all of it, and that's where **enums** and collections come in. Enums are types that let you define valid values to categorize your data. Collections are special objects that store many values, letting you **store, sort, and manage** all the data that your programs need to pore through. That way, you can spend your time thinking about writing programs to work with your data, and let the collections worry about keeping track of it for you.

The rarely-played duke of oxen card

Unity Lab 4

User Interfaces

In the last Unity Lab you started to build a game, using a prefab to create GameObject instances that appear at random points in 3D space and fly in circles. This Unity Lab picks up where the last one left off, allowing you to apply what you've learned about interfaces in C# and more.



This screenshot shows the game in its running mode. Balls are added and the player can click on them to score.

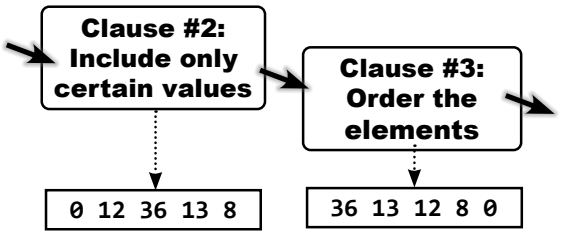
When the last ball is added, the game switches to its Game Over mode. The Play Again button pops up and no more balls get added.

LINQ and lambdas

9 Get control of your data

You're ready for a whole new world of app development.

Using WinForms to build Windows Desktop apps is a great way to learn important C# concepts, but there's *so much more* you can do with your programs. In this chapter, you'll use **XAML** to design your Windows Store apps, you'll learn how to **build pages to fit any device**, **integrate** your data into your pages with **data binding**, and use Visual Studio to cut through the mystery of XAML pages by exploring the objects created by your XAML code.

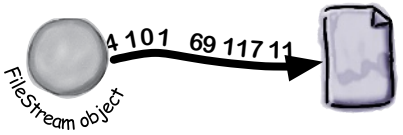


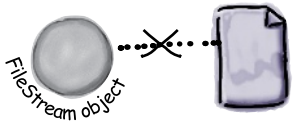
reading and writing files

10 Save the last byte for me!

Sometimes it pays to be a little persistent.

So far, all of your programs have been pretty short-lived. They fire up, run for a while, and shut down. But that's not always enough, especially when you're dealing with important information. You need to be able to **save your work**. In this chapter, we'll look at how to **write data to a file**, and then how to **read that information back in** from a file. You'll learn about the .NET **stream classes**, and also take a look at the mysteries of **hexadecimal** and **binary**.



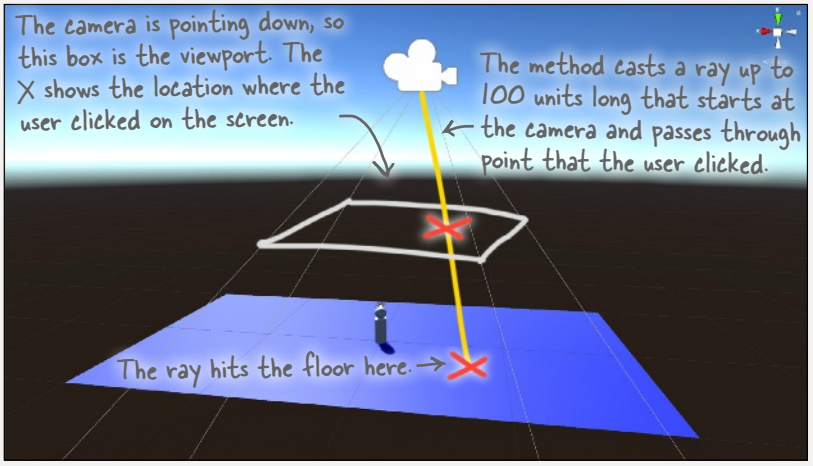


Eureka! → 

Unity Lab 5

Raycasting

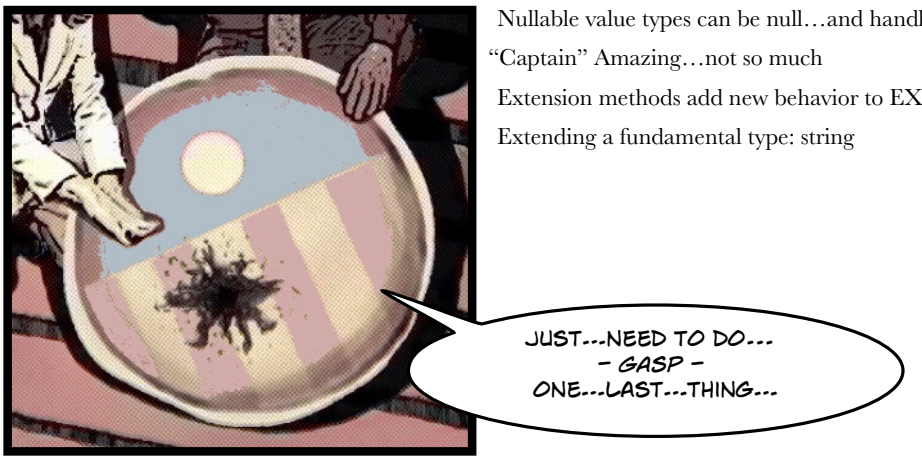
When you set up a scene in Unity, you're creating a virtual 3D world for the characters in your game to move around in. But in most games, things aren't directly controlled by the player. So how do these objects find their way around a scene? In this lab, we'll look at how C# can help.



CAPTAIN AMAZING

THE DEATH OF THE OBJECT

Head First C#	
Four bucks	Chapter 11

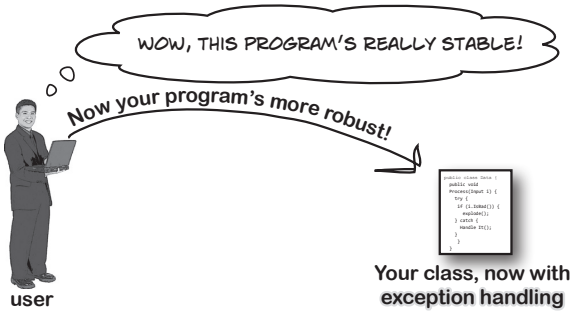


12 exception handling

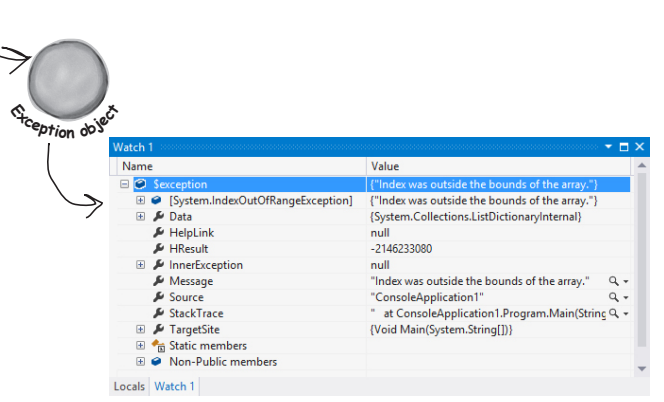
Putting out fires gets old

Programmers aren't meant to be firefighters.

You've worked your tail off, waded through technical manuals and a few engaging *Head First* books, and you've reached the pinnacle of your profession. But you're still getting panicked phone calls in the middle of the night from work because **your program crashes**, or **doesn't behave like it's supposed to**. Nothing pulls you out of the programming groove like having to fix a strange bug…but with **exception handling**, you can write code to **deal with problems** that come up. Better yet, you can even plan for those problems, and **keep things running** when they happen.



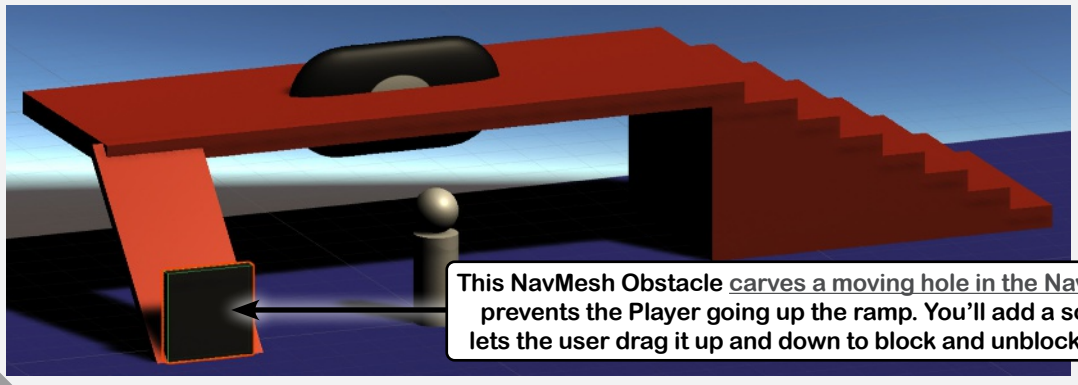

```
int[] anArray = {3, 4, 1, 11};
int aValue = anArray[15];
```



Unity Lab 6

Scene Navigation

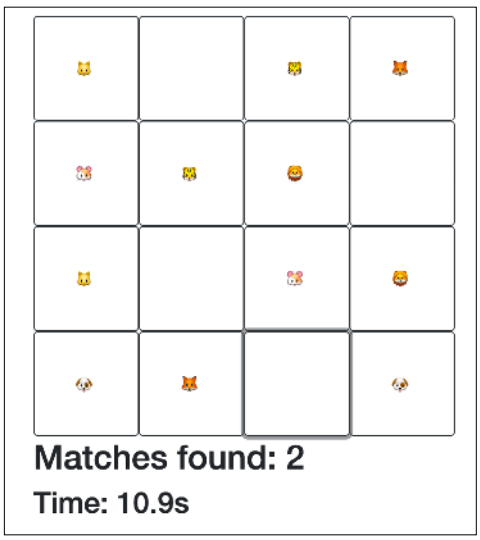
In the last Unity Lab, you created a scene with a floor (a plane) and a player (a sphere nested under a cylinder), and you used a NavMesh, a NavMesh Agent, and raycasting to get your player to follow your mouse clicks around the scene. In this lab, you'll add to the scene with the help of C#.



This NavMesh Obstacle carves a moving hole in the NavMesh that prevents the Player going up the ramp. You'll add a script that lets the user drag it up and down to block and unblock the ramp.

appendix i: ASP.NET Core Blazor projects

Visual Studio for Mac Learner's Guide



appendix ii: Code Kata

A learning guide for advanced and impatient readers