Learning C# by Programming Games

Wouter van Toll • Arjan Egges • Jeroen D. Fokker

Learning C#
by Programming Games

Second Edition

Springer

Wouter van Toll
Rennes, France

Jeroen D. Fokker
Department of Information and
Computing Sciences
Utrecht University
Utrecht, The Netherlands

Arjan Egges
Utrecht, The Netherlands

ISBN 978-3-662-59251-9 ISBN 978-3-662-59252-6 (eBook)
https://doi.org/10.1007/978-3-662-59252-6

© Springer-Verlag GmbH Germany, part of Springer Nature 2013, 2019
This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Cover design: Cover design based on a game sprite designed by Heiny Reimes. Reused by permission.

This Springer imprint is published by the registered company Springer-Verlag GmbH, DE, part of Springer Nature.
The registered company address is: Heidelberger Platz 3, 14197 Berlin, Germany

Preface

Introduction

If you are reading this, then we assume that you are interested in learning how to develop your own computer games. If you do not yet know how to program, don't worry! Developing computer games is actually a perfect way to learn how to program in modern programming languages. This book will teach you how to program in C# without requiring any previous programming experience. It does so through the creation of computer games.

In our opinion, C# is the language of choice if you want to learn how to program. The language is very well structured, it contains many modern features, and it is used a lot for game development and software development in general. Compared to Java (another popular programming language), some things are slightly easier to understand in C#. Also, once you understand how C# works, it is relatively easy to move to C++, which is still one of the most important programming languages in the game industry.

This book gives you a thorough introduction to C# and object-oriented programming in general. The unique aspect of this book is that it uses the structure and elements of computer games as a basis. For instance, the book contains chapters on dealing with player input, game objects, game worlds, levels, animation, physics, and intelligence. Along the way, you will learn about many programming concepts, such as variables, objects, classes, loops, collections, and exception handling.

We'll also discuss various aspects of software design that are useful for game development. For example, we propose useful techniques for managing levels and game states, as well as a hierarchy of game objects that together form an interactive game world. Furthermore, we will show a number of commonly used techniques in games, such as drawing layers of sprites, rotating, scaling and animating sprites, dealing with physics, handling interaction between game objects, and creating nice visual effects.

Throughout the book, you will create four games that gradually become more complicated. We have chosen different types of games to show the various aspects of game development. We start with a simple shooting game, we move on to puzzle games consisting of multiple levels, and we conclude with a full-fledged platform game with animation, game physics, and intelligent enemies.

For each of these games, the book invites you to write the code yourself. After all, the best way to learn how to program is to simply *do it*. Along with this book, we've created many example projects that you can use for reference.

Target Audience

This book is aimed at people who:

- are interested in game programming,
- have little to no programming experience so far,
- want to know how programming *really* works in general, so that they can use this knowledge in other places as well.

Using games as a guideline, this book will teach you the most important elements of object-oriented programming in C#. You'll also learn about programming on a more abstract level, so that you can easily teach yourself other programming languages in the future.

Due to this setup, the book is also very useful as a textbook for an introductory programming course on university level (see the "For Teachers" section). However, our target audience is not limited to students: if you recognize yourself in the bullet-point list above, then you can benefit from this book, no matter what your background is.

Structure of This Book

This book is divided into five parts. Each part consists of multiple chapters, and each chapter has its own collection of example programs in which you learn new programming concepts. You can download these example programs from the book's website.

Overall, the five parts of this book have the following contents:

- Part I serves as an introduction to programming in general, and it shows how games are generally developed. It introduces basic programming concepts such as variables, methods, and parameters. It also discusses the game loop: the basis of every game program.
- In Part II of the book, you will develop your first game called Painter. You will first learn how to deal with game assets such as sprites and sounds. Then, you will learn about the **if**-instruction and loops such as **for** or **while**. Furthermore, you'll learn about the basics of object-oriented programming. You will also see how to create a game world that consists of multiple interacting game objects. In terms of general programming techniques, this is the most important part of the book. After this part, the games will become more complicated, but they will reuse many of the techniques you've already learned in Part II.
- In Part III, you will continue setting up this structure of communicating game objects by using *arrays* and *collections*. These programming concepts are very important for the second game, Jewel Jam, which is a pattern recognition game. You will also see how to scale a game to different screen sizes and how to model the game world as a *hierarchy* of objects that can contain each other.
- Part IV introduces the puzzle game Penguin Pairs. This game shows you how to deal with different game states such as menus or level selection screens. You'll learn how to read levels from files and how to store the player's progress in a file. You'll also learn how to deal with sprite sheets: larger image files that store multiple "sub-images" for your game. Finally, this part deals with organizing classes into different libraries that you can use across different game projects.
- Finally, Part V revolves around the platform game Tick Tick, which reuses the game engine you've been building so far. In this part, you will deal with loading and playing animations, and you will add basic physics to the game world, such as jumping, falling, and handling collisions with other game objects. You'll also learn how to deal with unexpected errors (*exceptions*).

Each chapter (except Chap. 1) ends with a collection of *exercises*. Sometimes, these are general questions that help you get more familiar with the concepts introduced in that chapter. Other exercises invite you to extend the example programs from that chapter, using the things you've learned so far. The more advanced exercises are marked with an asterisk (*). Furthermore, in the last chapter of each part (starting in Part II), there's an exercise that invites you to extend the main game from that part. These are open and creative challenges that you can take as far as you want.

Finally, throughout the book, you will find text in gray boxes. These boxes usually contain "side notes" that go a bit deeper than the main text. In this second edition of the book, we've also added *Quick Reference* boxes that each give a short summary of a programming concept.

Required Materials and Tools

To develop the games of this book, you will need to install a few tools on your computer:

- *Visual Studio Community 2017*, a free program by Microsoft for writing and compiling code in C#.
- *MonoGame 3.6*, an open-source library that adds game-specific functionality to the C# language.

On the website of this book, you can find detailed instructions on how to obtain and install these tools. Chapter 1 will also help you further with these preparations.

Note that Visual Studio and MonoGame are updated frequently. By the time you read this book, newer versions of these tools may already be available. Of course, you are free to use these newer versions as well. However, if you want to be sure that you can run our example projects, we advise you to install the versions mentioned above.

If these versions of Visual Studio or MonoGame become so outdated that the book becomes too difficult to use, we will try to update the book's website to help you out as much as possible.

Along with this book, we supply various materials. All the example programs used in this book are available as Visual Studio projects that you can open, edit, compile, and run yourself. Furthermore, we supply a set of game assets (sprites and sounds) that are used in our example programs. All these additional materials can be found on the accompanying website of this book.

Website — Along with the book itself, there is an accompanying website from which you can download all additional materials. The URL of this website is http://www.csharpprogrammingames.com/.

For Teachers: Using This Book for a Programming Course

This book is very suitable as a basis for a game-oriented programming course. Each part of this book is concluded by exercises and challenges. Solutions to the exercises are available on the book's website. The challenges are generally more complex programming exercises, and they can serve as practical assignments for students following the course. On the accompanying website, a number of additional challenges are available that can be used as a basis for practical assignments as well.

By following the structure of the book throughout the course, students will be introduced to the main aspects of programming in an object-oriented language. Supplementary materials for organiz-

ing such a course are available on the website of this book, www.csharpprogrammingames.com. A sample schedule of a course with 15 sessions and 3 practical assignments is given below:

Session	C# topics	Game topics	Chapters	Deadlines
1	Introduction		1, 2	
2	Types, variables	Game loop	3, 4	
3	**if**, **else**, booleans	Sprites, sound, player input	5, 6	
4	Classes, methods, objects	Object interaction	7, 8	
5	**for**, **while**, randomness		8, 9	Ass. 1
6	Inheritance	Drawing text	10, 11	
7	Arrays, strings, collections	Screen sizes, grids	12, 13	
8	Recursion	Game worlds	14, 15	
9	Abstract classes, interfaces		16, 17	
10		Game states, sprite sheets	17, 18	Ass. 2
11	File I/O, **switch**	Tiles, levels	19, 20	
12	Libraries, exceptions		21, 22	
13		Animation, game physics	23, 24	
14		Enemies	25, 26	
15	General questions		All	Ass. 3

New in the Second Edition

In this second edition of the book, we've made a number of key changes, based on our experience with using the book in a university-level game programming course. The text has been rewritten entirely, and chapters have been restructured and reordered. All in all, this second edition should provide an even smoother and more general introduction to (game) programming.

The most important changes in this edition are the following:

- We have updated all example programs to work with Visual Studio Community 2017 and MonoGame 3.6 (instead of the now obsolete XNA Game Studio). Readers can once again follow the material by using the most modern programming tools (at the time of writing this book).
- Throughout this book, the text now invites you to write your *own* code, instead of explaining how the book's example projects work. The example projects are still there as reference points, but you're encouraged to get to those reference points by writing the code yourself. We've added convenient checkpoint markers that indicate when your code should match a certain example project.
- We have made the most important terms easier to find in the text. All terms that appear in the index (at the end of the book) are printed in boldface when they're first explained.
- We have made a clearer distinction between general C# programming concepts and concepts that are specific to game development. For example, primitive types (such as **int** and **string**) and the concept of classes are part of the C# language, but the `Color` class and the game loop are specific to the MonoGame library. If you want to continue with software development in general, it's good to be aware of this distinction.
- We've made programming guidelines easier to recognize. The new "Quick Reference" boxes summarize a programming concept in a way that clearly stands out from the rest of the text. As a result,

compared to the first edition, it is now easier to see the difference between an example and the general lesson that you can learn from it.

- The exercises are now grouped per chapter (instead of per part), and they can now be found directly after their chapter (instead of in an appendix). This way, readers can check more easily whether they understand the material from a particular chapter. The exercises themselves have also been given an update: we've added and changed several exercises to give readers a more general education in C# programming.

If you're already using the first edition of the book for personal reference, then you can keep using it without too many difficulties: both versions of the book still cover the same material overall. In all other cases, we strongly recommend using the second edition: it's a more convenient introduction to (game) programming and a more complete basis for a course. If you are a student and your programming course has been updated to the second edition of the book, then upgrading is especially useful, also because the book's example projects and exercises have changed.

We hope you'll enjoy using this book for gaining your first experience with object-oriented (game) programming. When you have reached the end of this book, you will have a solid basis for becoming a good programmer. What comes after that is up to you, but we hope you'll be inspired to create wonderful games and other programs in the future.

Good luck!

Rennes, France — Wouter van Toll
Utrecht, The Netherlands — Arjan Egges
Utrecht, The Netherlands — Jeroen D. Fokker

Acknowledgments

Just as games are generally not developed by a single person but by a team, this book was a team effort as well. First of all, I'd like to thank my co-authors Jeroen and Mark for their inspiring ideas and the interesting discussions we've had while writing this book.

This book is based on reading material that was handed out to the (many) students of the game programming course here at Utrecht University. Their feedback and critical analysis of the material have been of great help, as well as their motivation to work with the example games. I officially apologize for any frustration that occurred due to a few ridiculously difficult levels of the Tick Tick game!

I would like to thank my colleagues for their ideas and their interest in this work. In particular, I would like to thank Cathy Ennis and Sybren Stüvel for providing corrections and for reading the text in detail. Their feedback has resulted in many improvements of the text and the sample programs.

The sprites for all the example games in this book were designed by Heiny Reimes. It was a pleasure working with him on designing the example games and improving them.

I would also like to thank Ralf Gerstner from Springer for taking the time to read the manuscript and helping to make this book a reality.

Finally, I would like to thank my wife, Sterre, for her continuing support. I dedicate this work to her.

Utrecht, The Netherlands — Arjan Egges
January 2013

For this second edition, I've been given the chance to improve the book and add my own insights and twists. I should start by thanking everyone who made the *first* edition possible: you gave me a luxurious starting point with logically chosen topics and fun example games. Thanks to Arjan for entrusting me with this update, and thanks again to Ralf Steiner at Springer, who allowed me to write a much bigger overhaul than we'd initially planned.

The improvements in this new edition are based on my teaching experience at Utrecht University. I'd like to thank Paul Bergervoet for guiding me into the Game Programming course there in 2017 and his successor Remco Veltkamp for his interest in improving it with me. Also, thanks to all the students who've provided feedback on the first edition. Your comments have definitely helped me turn this book into a smoother resource for students.

Special shout-out to Jeroen van Knotsenburg, a C# guru with a keen eye for clean and reusable code but (most importantly) a great friend who's been nearby ever since we took our own first programming course together.

Last but not least, I sincerely thank Nicoline, who's not just one of the most intelligent people I've ever met but also a really cool partner.

Rennes, France Wouter van Toll
January 2019

Contents

Part I Getting Started

1 Building Your First Game Application . . . 3
1.1 Getting and Installing the Tools . . . 3
1.2 Creating Your First Game Application . . . 3
1.3 Running the Game Project . . . 5
1.4 Changing Your First Line of Code . . . 5
1.5 Projects and Solutions . . . 7
1.6 Running the Examples in This Book . . . 7
1.7 What You Have Learned . . . 8

2 What Is Programming? . . . 9
2.1 Computers and Programs . . . 9
2.2 Programming Languages . . . 10
2.2.1 Imperative Programming: Instructions . . . 11
2.2.2 Procedural Programming: Instructions in Methods . . . 11
2.2.3 Object-Oriented Programming: Methods in Classes . . . 12
2.2.4 Java . . . 13
2.2.5 C# . . . 13
2.2.6 Other Modern Languages . . . 14
2.3 Types of Applications . . . 15
2.4 Game Programming and Game Engines . . . 16
2.5 Translating a Program . . . 17
2.5.1 Assembler . . . 17
2.5.2 Compiler . . . 18
2.5.3 Interpreter . . . 18
2.5.4 Compiler and Interpreter . . . 19
2.5.5 Two Compilers . . . 19
2.5.6 A Final Note . . . 20
2.6 Syntax and Semantics . . . 20
2.7 Development Cycles . . . 21
2.7.1 Small Scale: Edit-Compile-Run . . . 21
2.7.2 Large Scale: Design-Specify-Implement . . . 21
2.8 What You Have Learned . . . 22
2.9 Exercises . . . 23

3 Game Programming Basics . . . 25
3.1 Structure of a C# Program . . . 25
3.1.1 Instructions . . . 25
3.1.2 Methods: Grouping Instructions Together . . . 26
3.1.3 Methods and Parameters . . . 26
3.1.4 Classes: Grouping Methods Together . . . 26
3.1.5 A Class Is a Blueprint for Objects . . . 27
3.2 Hello World: The Smallest C# Program You Will Ever See . . . 27
3.3 Building Blocks of a Game . . . 29
3.3.1 The Game World . . . 29
3.3.2 The Game Loop . . . 30
3.3.3 How the Game Loop Gives the Illusion of Movement . . . 31
3.4 Structure of a MonoGame Application . . . 31
3.4.1 The Game Loop in MonoGame . . . 32
3.4.2 Other Methods of the Game Class . . . 33
3.4.3 The Graphics Device . . . 34
3.5 Structure of a Larger Program . . . 35
3.5.1 Namespaces: Grouping Classes Together . . . 35
3.5.2 Libraries: Relying on Other Classes . . . 36
3.5.3 Compilation Units . . . 37
3.6 Program Layout . . . 37
3.6.1 Comments: Explaining What You're Doing . . . 37
3.6.2 Instructions vs. Lines . . . 39
3.6.3 Whitespace and Indentation . . . 39
3.7 What You Have Learned . . . 40
3.8 Exercises . . . 40

4 Creating a Game World . . . 41
4.1 Variables and Types . . . 41
4.1.1 Types . . . 41
4.1.2 Declaring a Variable . . . 42
4.1.3 Assigning a Value to a Variable . . . 43
4.1.4 Declarations and Assignments: More Complex Examples . . . 44
4.1.5 Constants . . . 45
4.2 Expressions: Pieces of Code with a Value . . . 45
4.2.1 Expressions Versus Instructions . . . 46
4.2.2 Operators: Combining Expressions into New Expressions . . . 46
4.2.3 Priority of Operators . . . 47
4.2.4 Variants of Operators . . . 48
4.3 Other Data Types . . . 49
4.3.1 The Double Type . . . 49
4.3.2 Data Types for Integers . . . 50
4.3.3 Data Types for Real Numbers . . . 50
4.3.4 Mixing Different Data Types . . . 51
4.3.5 Data Types in General . . . 52
4.3.6 Classes, Instances, and Data Types . . . 52

4.4 Putting It to Practice: A Changing Background Color . . . 53
4.4.1 Using a Variable to Store a Color . . . 54
4.4.2 Creating Your Own Colors . . . 54
4.4.3 Using the GameTime Class to Change the Color . . . 55
4.4.4 The Scope of a Variable . . . 57
4.4.5 Creating a Member Variable . . . 58
4.4.6 Summary of the DiscoWorld Example . . . 59
4.5 What You Have Learned . . . 61
4.6 Exercises . . . 61

Part II Game Objects and Interaction

Introduction . . . 67

5 Showing What the Player Is Doing . . . 69
5.1 Using Game Assets in MonoGame . . . 69
5.1.1 Managing Assets with the Pipeline Tool . . . 70
5.1.2 Telling MonoGame Where to Load Your Assets . . . 72
5.1.3 Loading Sprites . . . 73
5.1.4 Drawing Sprites . . . 74
5.1.5 Music and Sounds . . . 75
5.2 A Sprite Following the Mouse Pointer . . . 77
5.2.1 Adding a Background . . . 77
5.2.2 Using a Vector2 Variable for the Position of a Sprite . . . 78
5.2.3 Retrieving the Mouse Position . . . 80
5.3 Changing the Sprite Origin . . . 82
5.3.1 Using the Width and Height of a Sprite . . . 82
5.3.2 Vector Math . . . 82
5.3.3 Multiple Versions of a Method . . . 84
5.4 A Rotating Cannon Barrel . . . 85
5.4.1 More Sprites and Origins . . . 85
5.4.2 An Angle Variable . . . 86
5.4.3 Computing Angles with the Math Class . . . 86
5.5 What You Have Learned . . . 88
5.6 Exercises . . . 88

6 Reacting to Player Input . . . 91
6.1 Reacting to a Mouse Click . . . 91
6.1.1 Enumerated Types . . . 91
6.1.2 Executing Instructions Depending on a Condition . . . 92
6.2 Boolean Expressions . . . 95
6.2.1 Comparison Operators . . . 95
6.2.2 Logical Operators . . . 96
6.2.3 Boolean Variables . . . 97
6.3 More on **if** Instructions . . . 98
6.3.1 The **else** Keyword . . . 98
6.3.2 A Number of Different Alternatives . . . 99
6.3.3 Handling Mouse Clicks Instead of Presses . . . 101
6.4 A Multicolored Cannon . . . 102
6.4.1 Choosing Between Multiple Colors of Sprites . . . 103
6.4.2 Handling Keyboard Input . . . 104

6.5 What You Have Learned . . . 105
6.6 Exercises . . . 106

7 Basic Game Objects . . . 107
7.1 Grouping Instructions into Methods . . . 107
7.1.1 A Method for Input Handling . . . 108
7.1.2 Methods with Parameters . . . 109
7.1.3 The Scope of Method Parameters . . . 110
7.1.4 Methods with a Result . . . 111
7.1.5 More on the **return** Keyword . . . 112
7.1.6 More on Return Values . . . 113
7.1.7 Expressions, Instructions, and Method Calls . . . 115
7.2 Organizing a Game into Classes . . . 116
7.2.1 Classes, Instances, and Objects . . . 117
7.2.2 Instances and Member Variables . . . 117
7.2.3 Instances and Method Calls . . . 118
7.2.4 Summary: How All Concepts Are Related . . . 119
7.2.5 Access Modifiers . . . 120
7.2.6 **this**: The Object That Is Being Manipulated . . . 121
7.3 Putting It to Practice: Adding a Cannon Class to Painter . . . 123
7.3.1 Adding a Class File to the Project . . . 123
7.3.2 Creating an Instance of Cannon in the Game . . . 124
7.3.3 Adding Member Variables . . . 125
7.3.4 Adding Your Own Constructor Method . . . 125
7.3.5 Adding a Parameter to the Constructor . . . 126
7.3.6 Adding Other Methods . . . 128
7.4 Accessing the Data in the Cannon Class . . . 129
7.4.1 Option 1: Marking Member Variables as **public** . . . 130
7.4.2 Option 2: Adding Getter and Setter Methods . . . 130
7.4.3 Option 3: Adding Properties . . . 131
7.4.4 Using the Properties of Cannon in the Painter Class . . . 134
7.4.5 More on Properties . . . 134
7.5 Reorganizing the Class Structure . . . 135
7.5.1 A Class for Handling Input . . . 135
7.5.2 Letting the Cannon Class Maintain Itself . . . 138
7.5.3 A Class for Representing the Game World . . . 139
7.6 What You Have Learned . . . 142
7.7 Exercises . . . 142

8 Communication and Interaction Between Objects . . . 145
8.1 How Objects Are Stored in Memory . . . 145
8.1.1 Primitive Types: Values . . . 145
8.1.2 Class Types: References . . . 146
8.1.3 The Consequence for Method Parameters . . . 147
8.1.4 The **null** Keyword . . . 149
8.1.5 Reference Counting and Garbage Collection . . . 150
8.1.6 Classes Versus Structs . . . 150
8.1.7 Complex Objects . . . 151

8.2 The Ball Class and Its Interaction with the Game World . . . 151
8.2.1 Structure of the Ball Class . . . 151
8.2.2 Overview of the Ball Behavior . . . 154
8.2.3 **static**: Making the Game World Accessible Everywhere . . . 154
8.2.4 Drawing the Ball at the Cannon Tip . . . 157
8.2.5 Letting the Ball Move . . . 157
8.2.6 Giving the Ball the Correct Launch Velocity . . . 160
8.2.7 Applying Gravity . . . 160
8.2.8 Resetting the Ball When It Leaves the Screen . . . 161
8.3 The PaintCan Class . . . 162
8.3.1 Structure of the Class . . . 162
8.3.2 Creating Three Different Paint Cans . . . 163
8.3.3 Making the Paint Cans Loop Around . . . 165
8.3.4 Taking the Origin into Account . . . 165
8.3.5 Giving the Cans Random Speeds . . . 166
8.3.6 More Randomness . . . 168
8.4 Handling Collisions Between the Ball and Cans . . . 169
8.4.1 Using Rectangular Bounding Boxes . . . 170
8.4.2 Responding to a Collision . . . 171
8.5 What You Have Learned . . . 171
8.6 Exercises . . . 172

9 A Limited Number of Lives . . . 173
9.1 Maintaining the Number of Lives . . . 173
9.2 Loops: Executing Instructions Multiple Times . . . 174
9.2.1 The **while** Instruction . . . 174
9.2.2 The **for** Instruction: A Compact Version of **while** . . . 176
9.3 Special Cases of Loops . . . 178
9.3.1 Zero Iterations . . . 178
9.3.2 Infinite Loops . . . 178
9.3.3 Nested Loops . . . 179
9.3.4 Loops Inside the Game Loop . . . 180
9.3.5 The Keywords **continue** and **break** . . . 181
9.4 Adding a “Game Over” State . . . 182
9.5 More on the Scope of Variables . . . 184
9.5.1 Variable Scope Inside a Loop . . . 184
9.5.2 Conclusions . . . 185
9.6 What You Have Learned . . . 186
9.7 Exercises . . . 186

10 Organizing Game Objects . . . 189
10.1 Introduction to Inheritance . . . 189
10.1.1 Motivation: Similarities Between Game Objects . . . 189
10.1.2 Example and Terminology . . . 190
10.2 Creating Your First Base Class: ThreeColorGameObject . . . 191
10.2.1 Member Variables . . . 191
10.2.2 Methods and Properties . . . 191
10.2.3 Constructor . . . 193

10.3 Turning Cannon into a Subclass . . . 193
10.3.1 Class Outline, Member Variables, and Properties . . . 194
10.3.2 Changing the Constructor . . . 196
10.3.3 **protected**: Making Things Available to Base Classes . . . 197
10.3.4 Overriding Methods from the Base Class . . . 198
10.3.5 Reusing Base Methods: The **base** Keyword . . . 200
10.4 Turning Ball and PaintCan into Subclasses . . . 202
10.5 More on Inheritance . . . 202
10.5.1 Polymorphism . . . 203
10.5.2 Hierarchies of Classes . . . 204
10.5.3 Sealed Methods and Classes . . . 206
10.5.4 How Not to Use Inheritance . . . 206
10.6 What You Have Learned . . . 206
10.7 Exercises . . . 207

11 Finishing the Game . . . 211
11.1 Adding the Finishing Touches . . . 211
11.1.1 Motion Effects . . . 211
11.1.2 Sounds and Music . . . 212
11.1.3 Maintaining a Score . . . 212
11.2 Dealing with Text in C# . . . 213
11.2.1 The **char** Data Type . . . 213
11.2.2 The **string** Data Type . . . 214
11.2.3 Special Characters and Escaping . . . 215
11.2.4 Operations with Strings . . . 216
11.2.5 Operations with Characters . . . 217
11.3 Putting It to Practice: Drawing the Score . . . 218
11.3.1 Adding a Font in MonoGame . . . 218
11.3.2 Drawing the Score on the Screen . . . 219
11.4 Writing Documentation . . . 219
11.5 What You Have Learned . . . 221
11.6 Exercises . . . 221

Part III Structures and Patterns

Introduction . . . 225

12 Dealing with Different Screen Sizes . . . 227
12.1 Scaling the Game World to Fit in the Window . . . 227
12.1.1 Changing the Window Size . . . 228
12.1.2 Storing the World Size and the Window Size Separately . . . 228
12.1.3 Calculating How to Scale the Game World . . . 229
12.1.4 Scaling the Game World . . . 230
12.2 Making the Game Full-Screen . . . 230
12.2.1 Allowing Full-Screen Mode and Windowed Mode . . . 231
12.2.2 Toggling Between the Two Modes . . . 232
12.2.3 Allowing the Player to Quit the Game . . . 233
12.3 Maintaining the Aspect Ratio . . . 233
12.3.1 What Is a Viewport? . . . 233
12.3.2 Calculating the Viewport . . . 234
12.3.3 Using the Viewport for Drawing Sprites . . . 235

12.4 Screen Coordinates and World Coordinates . . . 236
12.4.1 Converting Screen Coordinates to World Coordinates . . . 237
12.4.2 Showing the Mouse Position . . . 237
12.5 What You Have Learned . . . 238
12.6 Exercises . . . 238

13 Arrays and Collections . . . 241
13.1 Arrays: Storing a Sequence of Objects . . . 241
13.1.1 Basic Usage of Arrays . . . 242
13.1.2 Arrays in Memory . . . 243
13.1.3 Combining Arrays with Loops . . . 245
13.1.4 Multidimensional Arrays . . . 245
13.1.5 Arrays of Arrays . . . 247
13.1.6 Shorthand for Initializing an Array . . . 247
13.2 Putting It to Practice: Jewels in a Grid . . . 248
13.2.1 Creating the Grid . . . 248
13.2.2 Filling the Grid with Random Jewels . . . 249
13.2.3 Drawing the Jewels on the Screen . . . 249
13.2.4 Letting the Player Change the Array . . . 251
13.3 Strings and Arrays . . . 252
13.3.1 Immutability . . . 253
13.3.2 Useful String Methods . . . 254
13.3.3 String Operators . . . 254
13.4 Collections . . . 254
13.4.1 The List Class . . . 255
13.4.2 Lists, Loops, and the **foreach** Keyword . . . 256
13.4.3 Other Collections . . . 257
13.5 What You Have Learned . . . 258
13.6 Exercises . . . 259

14 Game Objects in a Structure . . . 263
14.1 Creating Your Own Game Class . . . 263
14.1.1 Outline, Member Variables, and Properties . . . 263
14.1.2 Methods . . . 264
14.1.3 Turning JewelJam into a Subclass . . . 266
14.2 A List of Game Objects . . . 266
14.2.1 The GameObject Class . . . 267
14.2.2 The SpriteGameObject Class . . . 268
14.2.3 Describing the Game World as a List of Game Objects . . . 269
14.2.4 Collections and Polymorphism . . . 271
14.3 Changing the Grid of Jewels . . . 271
14.3.1 Turning a Jewel into a Game Object . . . 271
14.3.2 Turning the Grid of Jewels into a Game Object . . . 272
14.3.3 Moving Behavior to the JewelGrid Class . . . 274
14.4 A Hierarchy of Game Objects . . . 275
14.4.1 Relations Between Game Objects . . . 276
14.4.2 Global and Local Positions . . . 277
14.4.3 Recursion: A Method or Property Calling Itself . . . 279
14.4.4 The GameObjectList Class . . . 279

14.5 More on Recursion . . . 280
14.5.1 The Two Ingredients of Recursion . . . 281
14.5.2 Example: The Sum from 1 to n . . . 281
14.5.3 Watch Out for Infinite Recursion . . . 283
14.6 What You Have Learned . . . 283
14.7 Exercises . . . 284

15 Gameplay Programming . . . 285
15.1 Selecting Rows and Moving the Jewels . . . 286
15.1.1 Class Overview and Constructor . . . 286
15.1.2 Adding a RowSelector to the Game World . . . 286
15.1.3 Changing the Selected Row . . . 287
15.1.4 Shifting the Jewels in a Row . . . 288
15.2 More Types of Jewels . . . 290
15.2.1 One Sprite Sheet for All Jewels . . . 290
15.2.2 Three Properties of a Jewel . . . 290
15.2.3 Drawing the Correct Part of the Sprite Sheet . . . 291
15.3 Finding Combinations of Jewels . . . 292
15.3.1 Checking if a Combination Is Valid . . . 292
15.3.2 Handling Keyboard Input . . . 293
15.3.3 Removing Jewels from the Grid . . . 294
15.4 Maintaining and Showing the Score . . . 295
15.4.1 A Separate Class for the Game World . . . 295
15.4.2 Updating the Score . . . 297
15.4.3 The TextGameObject Class . . . 297
15.4.4 Showing the Score on the Screen . . . 299
15.4.5 Extra Points for Multiple Combinations . . . 299
15.5 A Moving Jewel Cart . . . 300
15.5.1 The JewelCart Class . . . 300
15.5.2 Adding the Cart to the Game . . . 301
15.6 Multiple Game States . . . 302
15.6.1 Adding Overlay Images . . . 303
15.6.2 Using an Enum to Represent the Game State . . . 304
15.6.3 Different Behaviors per Game State . . . 304
15.6.4 Adding a Help Button . . . 306
15.7 What You Have Learned . . . 308
15.8 Exercises . . . 308

16 Finishing the Game . . . 309
16.1 Making the Jewels Move Smoothly . . . 309
16.1.1 Adding a Target Position . . . 309
16.1.2 Setting the Target Positions of All Jewels . . . 310
16.1.3 Letting the New Jewels Fall from the Sky . . . 311
16.2 Showing Combo Images for a Certain Amount of Time . . . 312
16.2.1 Managing the Visibility of Another Object . . . 312
16.2.2 Adding the Objects to the Game World . . . 313
16.2.3 Applying It to the Combo Images . . . 314

16.3 Adding Glitter Effects . . . 315
16.3.1 Outline of the GlitterField Class . . . 315
16.3.2 Calculating a Random Glitter Position . . . 317
16.3.3 Adding Glitter Fields to Game Objects . . . 318
16.4 Adding Music and Sound Effects . . . 320
16.4.1 An Improved Asset Manager . . . 320
16.4.2 Playing the New Sounds and Music . . . 321
16.5 What You Have Learned . . . 321
16.6 Exercises . . . 322

Part IV Menus and Levels

Introduction . . . 327

17 Better Game State Management . . . 329
17.1 New Classes for Game State Management . . . 329
17.1.1 The GameState Class . . . 330
17.1.2 A Collection of Game States . . . 331
17.1.3 Letting the Active Game State Do Its Job . . . 331
17.1.4 Adding a GameStateManager to the Game . . . 333
17.2 Creating the Game States of Penguin Pairs . . . 333
17.2.1 Adding and Organizing the Assets . . . 333
17.2.2 Adding a Class for Each Game State . . . 333
17.2.3 Preparing the PenguinPairs Class . . . 335
17.2.4 Adding Constants for Object Names . . . 336
17.2.5 Switching Between Game States . . . 337
17.3 Abstract Classes . . . 338
17.3.1 What Is an Abstract Class? . . . 338
17.3.2 Abstract Classes in the Game Engine . . . 339
17.3.3 Abstract Methods . . . 339
17.3.4 Abstract Properties . . . 340
17.4 Interfaces . . . 341
17.4.1 What Is an Interface? . . . 341
17.4.2 Implementing an Interface . . . 342
17.4.3 Implementing Multiple Interfaces . . . 343
17.4.4 The IGameLoopObject Interface . . . 343
17.4.5 Interfaces and Polymorphism . . . 344
17.5 What You Have Learned . . . 346
17.6 Exercises . . . 346

18 User Interfaces and Menus . . . 349
18.1 Sprite Sheets . . . 349
18.1.1 Overview of the SpriteSheet Class . . . 349
18.1.2 Adding a Constructor . . . 351
18.1.3 Reading the Number of Sprites from the Filename . . . 352
18.1.4 A Property for the Sheet Index . . . 353
18.1.5 Updating the SpriteGameObject Class . . . 354

18.2 Menu Elements . . . 354
18.2.1 Obtaining the Mouse Position in World Coordinates . . . 354
18.2.2 The Button Class . . . 355
18.2.3 The Switch Class . . . 356
18.2.4 The Slider Class . . . 357
18.3 Adding the First Menus of Penguin Pairs . . . 360
18.3.1 Filling the Title Screen . . . 360
18.3.2 Adding the Three Back Buttons . . . 360
18.3.3 Adding Buttons to the Playing State . . . 361
18.4 Filling the Options Menu . . . 361
18.4.1 Adding the UI Elements . . . 361
18.4.2 Changing the Music Volume . . . 363
18.4.3 Enabling and Disabling Hints . . . 363
18.5 Filling the Level Selection Screen . . . 364
18.5.1 Defining Level Statuses . . . 364
18.5.2 The LevelButton Class . . . 364
18.5.3 Adding a Grid of Level Buttons . . . 366
18.5.4 Starting a Level . . . 367
18.6 What You Have Learned . . . 368
18.7 Exercises . . . 368

19 Loading Levels from Files . . . 371
19.1 Structure of a Level . . . 371
19.1.1 All Possible Level Elements . . . 372
19.1.2 Defining a File Format . . . 373
19.2 Creating the Classes of Level Elements . . . 374
19.2.1 The Tile Class . . . 374
19.2.2 The Animal Class . . . 376
19.2.3 Subclasses of the Animal Class . . . 376
19.3 Reading and Writing Files . . . 377
19.3.1 Introduction to File I/O . . . 378
19.3.2 I/O in C#: Streams, Readers, and Writers . . . 378
19.3.3 Reading a Text File in C# . . . 380
19.3.4 Reading One Line at a Time . . . 381
19.3.5 Writing a Text File in C# . . . 382
19.4 Putting It to Practice: Reading a Level File . . . 383
19.4.1 Outline of the Level Class . . . 383
19.4.2 Reading the General Data . . . 385
19.4.3 Reading the Lines of the Grid . . . 386
19.4.4 Preparing the Grid . . . 386
19.4.5 Filling the Grid with Tiles and Animals . . . 387
19.4.6 The AddTile Method . . . 388
19.4.7 The AddAnimal Method . . . 389
19.5 Adding Levels to the Game . . . 391
19.5.1 Adding the Level Files to the Project . . . 391
19.5.2 Storing a Level in the Playing State . . . 391
19.5.3 Letting the Playing State Load a Level . . . 392
19.5.4 Loading a Level When the Player Asks for It . . . 393

19.6 Reading and Writing the Player's Progress . . . 393
19.6.1 Representing the Player's Progress . . . 394
19.6.2 Loading the Progress File . . . 394
19.6.3 Saving a New Progress File . . . 395
19.6.4 Showing the Player's Progress in the Level Menu . . . 395
19.7 The **switch** Instruction: Handling Many Alternatives . . . 396
19.7.1 Basic Usage of **switch** . . . 396
19.7.2 Using **return** Instead of **break** . . . 398
19.7.3 Restrictions . . . 398
19.8 What You Have Learned . . . 399
19.9 Exercises . . . 399

20 Gameplay Programming . . . 401
20.1 Selecting Animals . . . 401
20.1.1 The Arrow Class . . . 401
20.1.2 The MovableAnimalSelector Class: Outline . . . 403
20.1.3 The MovableAnimalSelector Class: Game Loop . . . 404
20.1.4 Handling Clicks on an Animal . . . 405
20.1.5 The Order of Input Handling . . . 406
20.2 Making the Animals Move . . . 407
20.2.1 Linking the Animals to the Grid . . . 407
20.2.2 Overview of the MovableAnimal Design . . . 408
20.2.3 Moving and Stopping . . . 409
20.3 Adding the Game Logic . . . 410
20.3.1 Preparing the Level Class . . . 410
20.3.2 Checking if a Move Is Possible . . . 411
20.3.3 Checking if Two Movable Animals Can Form a Pair . . . 412
20.3.4 Checking if an Animal Is Movable: The Keyword **is** . . . 413
20.3.5 Applying the New Position of a Movable Animal . . . 414
20.3.6 Cleaning Up the Code . . . 415
20.4 Maintaining the Number of Pairs . . . 415
20.4.1 The PairList Class . . . 416
20.4.2 Storing and Showing Pairs in the Level . . . 417
20.4.3 Adding a Pair at the Right Time . . . 418
20.5 What You Have Learned . . . 418
20.6 Exercises . . . 419

21 Finishing the Game . . . 421
21.1 Hints and Resetting . . . 421
21.1.1 Showing a Hint . . . 421
21.1.2 Resetting a Level . . . 422
21.1.3 Resetting the Other Game Objects . . . 423
21.2 Completing a Level . . . 423
21.2.1 Updating the PlayingState and Level Classes . . . 424
21.2.2 Going to the Next Level . . . 424
21.2.3 Updating the Buttons in the Level Menu . . . 425
21.3 Adding Music and Sound Effects . . . 426

21.4 Using Libraries and Namespaces . . . 426
21.4.1 A Separate Library for the Engine . . . 427
21.4.2 Access Modifiers for Classes . . . 427
21.4.3 A Namespace for the Engine Classes . . . 428
21.5 What You Have Learned . . . 429
21.6 Exercises . . . 430

Part V Animation and Complexity

Introduction . . . 433

22 Creating the Main Game Structure . . . 435
22.1 Changes in the Engine Project . . . 435
22.1.1 Classes for Games with Levels . . . 436
22.1.2 Adding Depth to Game Objects . . . 436
22.1.3 Other Changes . . . 437
22.2 Overview of the Initial Game . . . 438
22.2.1 Main Classes . . . 438
22.2.2 The Tile Class . . . 439
22.2.3 The WaterDrop Class . . . 439
22.2.4 The Goal Object . . . 440
22.2.5 Level Files . . . 440
22.2.6 Loading a Level . . . 441
22.3 Dealing with Large Classes . . . 441
22.3.1 Partial Classes . . . 442
22.3.2 Regions . . . 442
22.4 Exceptions: Dealing with Unexpected Errors . . . 443
22.4.1 Dealing with Exceptions: **try** and **catch** . . . 444
22.4.2 Multiple Types of Exceptions . . . 444
22.4.3 Throwing Your Own Exceptions . . . 445
22.4.4 When to Use Exceptions? . . . 446
22.4.5 Exceptions and Casting: The Keyword **as** . . . 447
22.5 What You Have Learned . . . 448
22.6 Exercises . . . 448

23 Animated Game Objects . . . 451
23.1 A Class for Animations . . . 451
23.1.1 New Member Variables and Properties . . . 453
23.1.2 Playing an Animation . . . 453
23.2 Adding Animations to Game Objects . . . 454
23.2.1 Managing Multiple Animations . . . 455
23.2.2 Updating the Current Animation . . . 456
23.3 Adding the Player Character to Tick Tick . . . 456
23.3.1 Overview of the Player Class . . . 456
23.3.2 Adding an Instance to the Game . . . 458
23.4 What You Have Learned . . . 458
23.5 Exercises . . . 459

24 Game Physics . . . 461
24.1 Jumping and Falling . . . 461
24.1.1 Applying Gravity . . . 462
24.1.2 Letting the Character Jump . . . 462
24.1.3 Finishing Touches and Design Choices . . . 463
24.2 General Collision Detection . . . 464
24.2.1 Collision Detection with Bounding Volumes . . . 464
24.2.2 Limitations of Bounding Volumes . . . 466
24.3 Adding Collision Detection to Tick Tick . . . 467
24.3.1 Communication Between the Character and the Level . . . 467
24.3.2 Giving the Character a Custom Bounding Box . . . 468
24.3.3 Adding the HandleTileCollisions Method . . . 468
24.3.4 Finding the Range of Tiles to Check . . . 469
24.3.5 Dealing with Level Boundaries . . . 470
24.3.6 Collision Detection with a Single Tile . . . 471
24.3.7 Responding to a Collision with a Tile . . . 472
24.3.8 Dealing with False Positives . . . 473
24.4 Adding Slippery Ice and Friction . . . 474
24.4.1 Detecting the Type of Surface Below the Player . . . 474
24.4.2 Adding Friction to the Game Physics . . . 475
24.5 Pixel-Precise Collision Detection . . . 476
24.5.1 Getting and Storing Transparency Information . . . 477
24.5.2 Checking for Overlap Between SpriteGameObject Instances . . . 477
24.6 What You Have Learned . . . 478
24.7 Exercises . . . 479

25 Intelligent Enemies . . . 481
25.1 A Simple Moving Enemy . . . 481
25.1.1 Preparing the Level and Player Classes . . . 481
25.1.2 Overview of the Rocket Class . . . 482
25.2 Enemies with Timers . . . 484
25.2.1 The Turtle Class . . . 484
25.2.2 The Sparky Class . . . 486
25.3 Patrolling Enemies . . . 486
25.3.1 The Basic PatrollingEnemy Class . . . 487
25.3.2 Subclasses for Variants of the Enemy . . . 487
25.3.3 Adding the Enemies to the Game World . . . 489
25.4 What You Have Learned . . . 490
25.5 Exercises . . . 490

26 Finishing the Game . . . 491
26.1 Level Progression . . . 491
26.1.1 Collecting Water Drops . . . 491
26.1.2 Checking If the Level Has Been Completed . . . 492
26.1.3 Celebrating a Victory . . . 492
26.2 Life and Death . . . 493
26.2.1 Changing the Player Class . . . 493
26.2.2 Changing Other Classes . . . 494
26.2.3 Resetting the Level . . . 494

26.3 Adding a Timer 495
26.3.1 The BombTimer Class: Maintaining and Showing the Time 495
26.3.2 Integrating the Timer into the Game 497
26.3.3 Letting Hot Tiles Speed Up the Timer 497
26.4 Finishing Touches 498
26.4.1 Adding Mountains in the Background 498
26.4.2 Adding Moving Clouds 498
26.4.3 Adding Sound Effects 499
26.5 What You Have Learned 500
26.6 Exercises 501
26.7 What's Next? 503

Glossary 505

Index 509