

THE ESSENTIAL GUIDE TO

Physics for Flash Games, Animation, and Simulations

DEV RAMTAL AND
ADRIAN DOBRE

friendsof ™
an Apress® company

The Essential Guide to Physics for Flash Games, Animation, and Simulations

Dev Ramtal
Adrian Dobre



The Essential Guide to Physics for Flash Games, Animation, and Simulations

Copyright © 2011 by Dev Ramtal and Adrian Dobre

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

ISBN-13 (pbk): 978-1-4302-3674-0

ISBN-13 (electronic): 978-1-4302-3675-7

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logos, or image we use the names, logos, or images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

Distributed to the book trade worldwide by Springer Science+Business Media LLC., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com.

For information on translations, please e-mail rights@apress.com or visit www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales—eBook Licensing web page at www.apress.com/bulk-sales.

The information in this book is distributed on an “as is” basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author(s) nor Apress shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

The source code for this book is freely available to readers at www.apress.com in the Downloads section.

Credits

President and Publisher: Paul Manning
Coordinating Editor: Corbin Collins

Lead Editor: Chris Nelson
Copy Editor: Nancy Sixsmith

Technical Reviewer: RJ Owen
Compositor: Apress Production (Christine Ricketts)

Editorial Board: Steve Anglin, Mark Beckner,
Ewan Buckingham, Gary Cornell,
Jonathan Gennick, Jonathan Hassell,
Michelle Lowman, James Markham,
Matthew Moodie, Jeff Olson, Jeffrey Pepper,
Frank Pohlmann, Douglas Pundick,
Ben Renow-Clarke, Dominic Shakeshaft,
Matt Wade, Tom Welsh
Indexer: SPI Global
Artist: SPI Global
Cover Artist: Anna Ishchenko
Cover Designer: Corné van Dooren

To my mother, who made me possible; and to my wife and daughter, who supported me in making this book possible.

–DR

To my family who have been so understanding while I have been away from them working on this book.

–AD

Contents at a Glance

Contents at a Glance	iv
Contents	vi
About the Authors	xvii
About the Technical Reviewer.....	xviii
About the Cover Image Artist.....	xix
Acknowledgments	xx
Preface	xxi
Part I: The Basics	
Chapter 1: Introduction to Physics Programming.....	3
Chapter 2: Selected ActionScript 3.0 Topics	13
Chapter 3: Some Math Background	45
Chapter 4: Basic Physics Concepts.....	91
Part II: Particles, Forces and Motion	
Chapter 5: The Laws Governing Motion	135
Chapter 6: Gravity, Orbits, and Rockets.....	157
Chapter 7: Contact and Fluid Forces	181
Chapter 8: Restoring Forces: Springs and Oscillations	221
Chapter 9: Centripetal Forces: Rotational Motion	251
Chapter 10: Long-Range Forces	281

Part III: Multi-particle and Extended Systems

Chapter 11: Collisions 321

Chapter 12: Particle Systems 359

Chapter 13: Extended Objects 395

Part IV: Building More Complex Simulations

Chapter 14: Numerical Integration Schemes 443

Chapter 15: Other Technical Issues 461

Chapter 16: Simulation Projects 481

Index 517

Contents

Contents at a Glance	iv
Contents	vi
About the Authors	xvii
About the Technical Reviewer.....	xviii
About the Cover Image Artist.....	xix
Acknowledgments	xx
Preface	xxi
What this book will (and won't) teach you	xxi
Overview of this book	xxii
Source code and examples.....	xxii
Part I: The Basics	
Chapter 1: Introduction to Physics Programming.....	3
Why model real physics?	3
Creating realistic animation effects	4
Creating realistic games.....	4
Building simulations and models	4
Generating art from code	4
Can't I just use a physics engine?.....	5
What is physics?.....	6
Everything behaves according to the laws of physics.....	6
The laws can be written as math equations	6
Predicting motion.....	7
Programming physics	7
The difference between animation and simulation.....	8
The laws of physics are simple	8
Hence, they can be readily coded up!	8
The four steps for programming physics.....	9
A simple example	9
A bouncing ball: the physics.....	10
Coding up a bouncing ball in 2D	10
Summary	12

Chapter 2: Selected ActionScript 3.0 Topics	13
ActionScript 3.0 classes	14
Classes and objects	14
Structure of an AS3.0 class	15
Functions, methods and constructors	16
Properties	16
Static methods and properties	17
Inheritance	18
ActionScript 3.0 language basics	18
Variables and constants	18
Data types	19
Operators	22
Math	23
Logic	24
Loops	25
Events in ActionScript 3.0	27
Event listeners and handlers	27
Events and user interaction	28
Drag and drop	28
The Flash coordinate system	29
2D coordinates	29
3D in Flash	30
The Flash drawing API	32
Drawing lines and curves	32
Creating fills and gradients	33
Example: Bouncing ball in a box	34
Producing animation using code	36
Using the built-in frame stepping as a clock	36
Using the Timer class	37
Using getTimer() to compute elapsed time	38
Precalculating motion	40
Collision detection	40
Using the hitTestObject() method	41
Using the hitTestPoint() method	41
Distance-based collision detection	41
Advanced collision detection	43
Summary	43
Chapter 3: Some Math Background	45
Coordinates and simple graphs	46

Building a plotter: the Graph class	46
Plotting functions using the Graph class	47
Straight lines.....	48
Polynomial curves	49
Things that grow and decay: exponential and log functions	50
Making an object move along a curve.....	52
Distance between two points.....	59
Basic trigonometry.....	60
Degrees and radians	60
The sine function	61
The cosine function	63
The tangent function.....	63
The inverse trig functions	64
Using trig functions for animation	65
Vectors and basic vector algebra.....	70
What are vectors?	70
Vectors and scalars.....	71
Adding and subtracting vectors.....	72
Resolving vectors: vector components.....	73
Multiplying vectors: Scalar or dot product	76
Multiplying vectors: Vector or cross product	77
Building a Vector class with vector algebra.....	79
Simple calculus ideas.....	80
Slope of a line: gradient.....	81
Rates of change: derivatives	82
Doing sums: integrals.....	87
Summary	89
Chapter 4: Basic Physics Concepts.....	91
General physics concepts and notation	92
Physical quantities and units	92
Scientific notation	92
Things: particles and other objects in physics.....	93
What is a particle?	93
Particle properties.....	94
Building a Particle class.....	94
Moving particles: the Mover class	98
Extending the Particle class	100
Describing motion: kinematics.....	103
Concepts: displacement, velocity, speed, acceleration.....	104

Combining vector quantities	107
Describing motion using graphs	108
Equations of motion for uniform acceleration	108
Applying the equations to projectile motion	110
More motion-related concepts: inertia, mass, and momentum	114
Predicting motion: forces and dynamics	114
The cause of motion: forces	115
The relationship between force, mass, and acceleration	115
Types of forces	116
Combining forces: force diagrams and resultant force	117
Forces in equilibrium	118
An example: object falling under gravity and drag	119
Energy concepts	122
The notion of work in physics	123
The capacity to do work: energy	124
Energy transfer, conversion, and conservation	125
Potential and kinetic energy	126
Power	126
Example: a rudimentary “car” simulation	127
Summary	131

Part II: Particles, Forces and Motion

Chapter 5: The Laws Governing Motion	135
Newton’s laws of motion	136
Newton’s first law of motion (N1)	136
Newton’s second law of motion (N2)	137
Newton’s third law of motion (N3)	138
Applying Newton’s laws	139
General method for applying $F = ma$	139
The Forcer class	139
The Forces class	141
A simple example: projectile with drag	142
A more complicated example: floating ball	144
Newton’s second law as a differential equation	146
Taking a deeper look at $F = ma$	146
An example: Fall under gravity and drag revisited	147
The principle of energy conservation	148
Conservation of mechanical energy	149
An example: Energy changes in a projectile	149

The principle of momentum conservation	152
Example: 1D elastic collision between two particles	154
Laws governing rotational motion.....	156
Summary	156
Chapter 6: Gravity, Orbits, and Rockets	157
Gravity	157
Gravity, weight, and mass	158
Newton’s universal law of gravitation	158
Creating the gravity function.....	159
Orbits	161
Orbiter class	162
Escape velocity.....	165
Two-body motion	166
Local gravity	169
The force of gravity near the Earth’s surface	169
Variation of gravity with height	170
Gravity on other celestial bodies	172
Rockets.....	173
It <i>is</i> rocket science!.....	173
Modeling the thrust of a rocket.....	173
Building a rocket simulation.....	174
Summary	180
Chapter 7: Contact and Fluid Forces	181
Contact forces	182
Normal contact forces.....	182
Tension and compression	183
Friction	183
Example: Sliding down a slope	185
Pressure	191
The meaning of pressure	191
Density.....	192
Variation of pressure with depth in a fluid	193
Static and dynamic pressure	194
Upthrust (buoyancy)	194
Archimedes’s Principle	195
Apparent weight.....	196
Submerged objects.....	196
Floating objects	197

Example: Balloon.....	197
Drag.....	199
Drag law for low velocities.....	200
Drag law for high velocities.....	201
Which drag law should I use?.....	202
Adding drag to the balloon simulation.....	203
Example: Floating ball.....	203
Terminal velocity.....	207
Example: Parachute.....	209
Lift.....	212
Lift coefficients.....	213
Example: An airplane.....	213
Wind and turbulence.....	216
Force due to the wind.....	216
Wind and drag.....	217
Steady and turbulent flow.....	217
Example: Air bubbles in a steady wind.....	217
Modeling turbulence.....	219
Summary.....	220
Chapter 8: Restoring Forces: Springs and Oscillations.....	221
Springs and oscillations: Basic concepts.....	221
Spring-like motion.....	222
Restoring force, damping, and forcing.....	222
Hooke's law.....	223
Free oscillations.....	224
The spring force function.....	224
Creating a basic oscillator.....	224
Simple harmonic motion.....	226
Oscillations and numerical accuracy.....	229
Damped oscillations.....	233
Damping force.....	233
The effect of damping on oscillations.....	234
Analytical solutions for oscillations with damping.....	236
Forced oscillations.....	237
Driving forces.....	238
Example: A periodic driving force.....	238
Example: A random driving force.....	239
Gravity as a driving force: bungee jumping.....	240
Example: Driving force by user interaction.....	243

Coupled oscillators: Multiple springs and objects	245
Example: A chain of objects connected by springs	246
Summary	250
Chapter 9: Centripetal Forces: Rotational Motion	251
Kinematics of uniform circular motion	251
Angular displacement	252
Angular velocity	253
Angular acceleration	253
Period, frequency, and angular velocity	254
Relation between angular velocity and linear velocity	255
Example: A rolling wheel	256
Particles with spin	258
Example: Satellite around a rotating Earth	259
Centripetal acceleration and centripetal force	263
Centripetal acceleration	263
Centripetal acceleration, velocity and angular velocity	264
Centripetal force	264
Common misconceptions about centripetal force	265
Example: Revisiting the satellite animation	266
Example: Circular orbits with gravitational force	267
Example: Car moving around a bend	270
Non-uniform circular motion	274
Tangential force and acceleration	274
Example: A simple pendulum	275
Summary	279
Chapter 10: Long-Range Forces	281
Particle interactions and force fields	281
Interaction at a distance	282
From particle interactions to force fields	282
Newtonian gravitation	283
Gravitational field due to a particle	283
Gravity with multiple orbiters	284
Gravity with multiple attractors	286
Particle trajectories in a gravity field	289
Building a simple black hole game	292
Electrostatic force	299
Electric charge	299
Coulomb's law of electrostatics	300

Charged particle attraction and repulsion.....	301
Electric fields	303
Electromagnetic force.....	306
Magnetic fields and forces.....	306
The Lorentz force law	308
Other force laws	310
Central forces	310
Gravity with a spring force law?	314
Multiple attractors with different laws of gravity.....	316
Summary	318

Part III: Multi-particle and Extended Systems

Chapter 11: Collisions	321
Collisions and their modeling	322
Bouncing off horizontal or vertical walls	323
Elastic bouncing	323
Implementing energy loss due to bouncing.....	326
Bouncing off inclined walls	327
Collision detection	327
Repositioning the particle	329
Calculating the new velocity	330
Velocity correction just before collision	332
An example of a ball bouncing off an inclined wall.....	334
Example of ball bouncing off multiple inclined walls	339
Collisions between particles in 1D.....	340
Repositioning the particles	340
Elastic collisions	343
Inelastic collisions.....	347
Collisions between particles in 2D.....	349
An example: 2D collisions between two particles	350
Example: multiple particle collisions.....	353
Example: multiple particle collisions with bouncing.....	354
Summary	358
Chapter 12: Particle Systems	359
Introduction to particle system modeling.....	360
Creating animated effects using particles	361
A simple example: splash effect with particles.....	361
Creating a particle emitter	364

Creating a smoke effect.....	367
Creating a fire effect	371
Creating fireworks.....	373
Particle animations with long-range forces	379
Particle paths in a force field	379
Building a wormhole	382
Interacting particle systems.....	385
Multiple particles under mutual gravity	385
A simple galaxy simulation	389
Summary	394
Chapter 13: Extended Objects	395
Rigid bodies.....	395
Basic concepts of rigid body modeling	396
Modeling rigid bodies.....	402
Rotational dynamics of rigid bodies.....	405
Simulating rigid body dynamics.....	409
Example: a simple wind turbine simulation	412
Example: Rolling down an inclined plane.....	415
Rigid body collisions and bouncing	422
Example: Simulating a single bouncing block	426
Example: Colliding blocks	430
Deformable bodies	432
Mass-spring systems.....	432
Rope simulation	434
Cloth simulation	438
Summary	440
Part IV: Building More Complex Simulations	
Chapter 14: Numerical Integration Schemes	443
General principles	444
Statement of the problem	444
Characteristics of numerical schemes.....	446
Types of integration schemes	447
Modifying Forcer to allow for different integration schemes.....	448
Euler integration	449
Explicit Euler.....	449
Implicit Euler.....	450
Semi-implicit Euler.....	450

Comparing the explicit and semi-implicit Euler schemes	451
Why use Euler and why not?	452
Runge-Kutta integration	452
Second-order Runge-Kutta scheme (RK2)	453
Fourth-order Runge-Kutta scheme (RK4)	454
Stability and accuracy of RK2 and RK4 compared with Euler	455
Verlet integration	456
Position Verlet	456
Velocity Verlet.....	458
Testing the stability and accuracy of the Verlet schemes	458
Summary	459
Chapter 15: Other Technical Issues.....	461
Doing physics in 3D.....	462
3D versus 2D physics.....	462
3D math	462
Creating classes for 3D objects and motion.....	468
Creating 3D models.....	470
Example: A rotating cube	471
Integrating with 3D engines.....	474
Looking ahead to Stage3D.....	475
Building scale models.....	475
Scaling for realism	475
A simple example	475
Choosing units.....	476
Scaling factors and parameter values	476
Rescaling equations	477
Building accurate simulations.....	478
Using Number for calculations	478
Choosing an appropriate integration scheme	479
Using an appropriate timestep	479
Using accurate initial conditions	479
Dealing with boundaries carefully.....	480
Summary	480
Chapter 16: Simulation Projects.....	481
Build a submarine.....	481
Brief review of the physics.....	482
The visual setup	482
The setup code.....	483

The basic motion code	484
Adding controls and visual effects	484
The full mover code	486
Further ideas	488
Build a flight simulator	488
Physics and control mechanisms of aircraft	488
What we will create	493
Creating the visual setup	493
Coding up the physics	495
Implementing the controls	499
Displaying flight information	499
Test flying the simulator	499
Further things to do	500
Create an accurate solar system model	500
What we will create	501
The physics	501
Coding up an appropriate integration scheme	501
Building an idealized single-planet simulation	503
Choosing appropriate scaling factors	505
Obtaining planetary data and initial conditions	506
Basic solar system model	507
Incorporating accurate initial conditions	512
Comparing the model results with NASA data	512
Other things to do	515
Summary	515
Index	517

About the Authors



Dev Ramtal has been coding physics for more than 20 years. Alongside his academic research background in mathematical and computational physics, he has a long history of web development experience since building his first website back in 1997. Dev has been working with the Flash platform for the past six years, using it as a research and teaching tool, for general web programming, and just for some serious fun. He is also increasingly interested in game programming. Dev has a BSc and a PhD in Physics from the University of London. He lives in the UK where he works as a research scientist in academia. He can be contacted at www.physicscodes.com.



Adrian Dobre has more than 15 years of experimental and computational modeling experience in fluid dynamics. His experience in scientific research and teaching brought his attention to ActionScript and Flash, where programming combines with nice visual output. From here his dedication over more than five years for using Actionscript as a tool to model physical processes and create virtual laboratory platforms. Adrian holds a BSc in Aeronautical Engineering from University *Politehnica* in Bucharest, Romania and a PhD in Engineering Science from the University of Western Ontario, Canada. He currently lives in Bucharest, Romania, with his family.

About the Technical Reviewer



RJ Owen is a senior software architect at Effective UI. RJ has been working with Flex since version 2 and has worked on many projects of varying sizes and complexities, but always favors those with great user interfaces. He is a regular blogger and has spoken at many events including 360|Flex, Adobe MAX, and O'Reilly's Web 2.0. RJ enjoys advocating for teaching developers how to design and branch into skills beyond software. He has an MBA degree in Physics and Computer Science and lives in Colorado with his family.

About the Cover Image Artist



Corné van Dooren designed the front cover image for this book. After taking a brief hiatus from friends of ED to create a new design for the Foundation series, he worked at combining technological and organic forms, the results of which now appear on this and other book covers.

Corné spent his childhood drawing on everything at hand and then began exploring the infinite world of multimedia—and his journey of discovery hasn't stopped since. His mantra has always been, "The only limit to multimedia is the imagination"—a saying that keeps him constantly moving forward.

Corné works for many international clients, writes features for multimedia magazines, reviews and tests software, authors multimedia studies, and works on many other friends of ED books. You can see more of his work at and contact him through his website at www.cornevandooren.com.

Acknowledgments

We are extremely grateful to all the individuals, at Apress and elsewhere, who have helped turn this book from an idea into physical reality. An enormous amount of work goes into producing a book, of which writing it is only one part. We were fortunate to have the support of such a fantastic team—we won't mention any names for fear of missing anyone.

We are also grateful to the Flash and programming community for sharing their knowledge and expertise so openly through blogs, forums, websites, and books. It is a great pleasure to be able to give something back after having gained so much from the community.

Finally, we would like to thank our respective families for their constant support and patience during the intense period of writing this book.

Preface

We hope you will enjoy reading this book as much as we enjoyed writing it. Doing physics with Flash is a lot of fun—there is a sense of satisfaction in writing a few lines of code and then seeing how they make objects behave like they do in the real world. The ability to do this gives us great power as programmers. The aim of this book is to provide you with tools that will make you feel that sense of power. Whether you want to build convincing animations, games that behave realistically, or accurate simulators, you will find herein plenty of tools, examples and ideas that should be of help.

The inspiration for this book came from the excellent book authored by Keith Peters, *Foundation ActionScript Animation*. This was the first book we read when we were still getting acquainted with Flash and ActionScript several years ago. The book contained a good amount of physics-related topics and showed us the power of Flash in creating interactive animations with physics that were as fun to code as they were to play with. Since then we became addicted to Flash and ActionScript, using them to create numerous physics-based simulations and other applications. Along the way we learned or figured out a lot of things that seemed worth sharing with the Flash community. Keith Peters's book provided a great introduction to physics-based animation. But it seemed to us that there was also a need for a book that explored physics further and in greater depth, and that catered for more demanding applications such as accurate simulators or more complex game programming. After all, there are several “game physics” books written for other programming languages such as C++ and Java, so why not ActionScript? The present book is meant to fill this gap.

We should make it clear at the outset that this is primarily a book about physics, and secondarily about Flash. Therefore, the focus is less on producing attractive animations, and more on modeling real physics using the Flash environment. Some of the simulations in the book may not be very pretty or smooth from a visual perspective, but they do contain a lot of physics goodness! The approach we adopt attempts to make serious physics accessible. But although we don't shy away from going into technical stuff, with all the accompanying math, we hope to have done so in a way that is simple and straightforward.

Inevitably, due to space and time restrictions, there are topics that we were not able to include or that we didn't explore in detail. Nevertheless, the book covers an awful lot of ground, taking you from coding a simple bouncing ball animation in a few lines of code in the first chapter to a highly accurate simulation of the solar system in the final chapter. We hope you enjoy the journey in between!

What this book will (and won't) teach you

The Essential Guide to Physics for Flash Games, Animation, and Simulations teaches you how to include physics into your programming. It does not teach you how to program. It does not teach you ActionScript. And though this book teaches you how to implement physics into your games (as well as other projects), it is not about game programming *per se*.

We assume that you have at least some programming experience with ActionScript 3.0 (AS 3.0) as well as familiarity with an authoring tool for building Flash applications in AS 3.0, such as Adobe Flash (version CS3 or above) or Adobe Flash Builder (formerly Adobe Flex Builder). This book does not teach you how to

use those authoring tools. If you don't have this background we highly recommend picking up one of the many excellent introductory books on Flash and ActionScript available, such as *Foundation ActionScript 3.0 for Flash and Flex* by Darren Richardson and Paul Milbourne. We also highly recommend *Foundation ActionScript 3.0 Animation: Making Things Move!* by Keith Peters.

We do not assume any previous knowledge of physics, and only assume basic school level math knowledge. One chapter is dedicated to explaining some of the more difficult math concepts and tools that you are likely to need in the book. All the physics concepts and knowledge you will need are explained in a self-contained way. Numerous applications are given throughout the book with full source code to illustrate the application of the principles learnt.

Overview of this book

This book is divided into four parts.

Part I: "The Basics" (Chapters 1-4) introduces the necessary background in basic math and physics concepts upon which the rest of the book builds. For completeness, it also covers selected topics in ActionScript 3.0 that are most pertinent to physics programming.

Part II: "Particles, Forces, and Motion" (Chapters 5-10) begins by formulating the laws of physics that govern the motion of particles under any type of force (Chapter 5). The next chapters then apply those laws to make objects move under the action of a large variety of forces including gravity, friction, drag, buoyancy, wind, springs, central forces, and many more. The main focus in this section is on simulating the motion of individual particles and other simple objects.

In Part III: "Multi-Particle and Extended Systems" (Chapters 11-13), we show you how to model more complicated systems, including multiple interacting particles and extended objects. In these systems the constituent particles and objects do not simply co-exist but interact with each other, mutually influencing their motion. These interactions include collisions, short-range forces, and long-range forces. This part includes a discussion of particle systems, rigid bodies, and deformable bodies.

Part IV: "Building More Complex Simulations" (Chapters 14-16) is devoted to building more complex simulations, where accuracy and/or realism is especially important, not just visual effects. This part includes a discussion of numerical integration schemes and other numerical and technical issues such as scale modeling and 3D. Part IV ends with a chapter including some example simulation projects.

Source code and examples

All the code for this book can be downloaded from the book's page on the Apress website: www.apress.com.

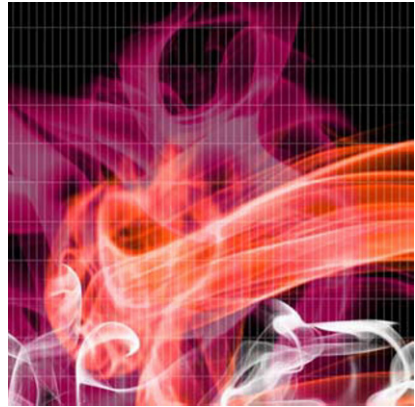
Source code in the book will work with Adobe Flash (version CS3 onwards; a few examples require CS4) and Adobe Flex/Flash Builder (version 3 onwards). All the code is pure ActionScript. We do not use MXML or the Flex framework.

We encourage you to modify and build upon the example codes given. While doing the technical review of this book, RJ Owen suggested it would be interesting to have a website where readers could share their

codes to see what people can build with the tools given in this book. We thought it was an excellent suggestion, and have therefore set up the following page on our website where you can find and contribute more code and examples: www.physicscodes.com/as3book.

Part I

The Basics



Chapter 1

Introduction to Physics Programming

Because you picked up this book, we assume that you are interested in implementing physics into your programming projects. But why would you want to do that? What can it do for you? And how difficult will it be? This chapter will provide answers to these questions.

Topics covered in this chapter include:

- **Why model physics?** This section will explain some of the reasons why you might want to add physics to your projects.
- **What is physics?** Here we lift the veil of mystery and explain in simple terms what physics is. We also tell you, in a nutshell, what you'll need to know.
- **Programming physics.** Thankfully, programming physics is not as difficult as you might imagine, once you understand some basic principles. This section explains what you'll need to do.
- **A simple example.** As a concrete example, we'll code up a simple animation involving physics, using a minimum of code.

Why model real physics?

There are a number of reasons why you might be interested in modeling physics using Flash. Here are some of the most common reasons.

Creating realistic animation effects

One of the main strengths of working with Flash is the ability to create animations easily. With a little ActionScript and some physics, it is also possible to make animations that look and behave like the real thing. For example, suppose you are animating a scene in which someone kicks a ball and it bounces off the ground. If you're working with Adobe Flash, you could try and fake the animation with tweens, but however hard you might try it would probably look less than realistic. With just a little bit of coding and some knowledge of simple physics you could produce a far more realistic animation. And if, like the authors, you are programmers rather than designers, you might even find it easier! We'll show you just how easy it can be in the example at the end of this chapter.

Creating realistic games

Flash games are ever so popular on the Web. As the capabilities of the Flash Player continue to improve, better and more powerful games can be built. Hardware acceleration and native 3D support are just two of the recent developments that have the potential to improve the gaming user experience dramatically. But apart from performance and appearance, it is equally important for games to feel realistic. If you are throwing a ball, it has to fall according to the law of gravity; if you fire a torpedo underwater, it must move differently from a ball falling in air. In other words, your game needs to know physics.

How do you build physics awareness into your games? This book will show you how.

Building simulations and models

A *computer simulation* or *computer model* is a program that attempts to imitate certain key aspects of a system. Simulations vary in completeness or accuracy, depending on purpose and resources. Let's take a flight simulator program as an example. One would expect a flight simulator designed for training pilots to be much more comprehensive and accurate than one designed for a game. Simulations are extremely common in e-learning, training and in scientific research. In the final chapter of this book, you'll build simulations, including a submarine, a basic flight simulator, and a model of the solar system. In fact, many of the coded examples throughout the book are simulations, even if generally simpler.

Generating art from code

Generative art has gained popularity in recent years. ActionScript and Processing are popular tools used by artists to generate art from code. A lot of fun can be had with some simple physics—for example, elaborate effects and motions can be produced using particles and different kinds of forces.

We will explore the world of generative art and provide additional tools and algorithms that can be used to create original and interesting effects.

Can't I just use a physics engine?

You probably can. Physics engines are certainly useful for a lot of situations. But they may not necessarily be the panacea for all your problems. Here are a few reasons why the time you invest in learning how to program physics can pay off:

- You'll have more flexibility to do what you want. Physics engines cannot do everything.
- In some cases, it may be simpler to write code from scratch rather than to use a physics engine.
- Knowing how to use a physics engine doesn't guarantee that you'll know what physics to use for a particular application.
- Sometimes things may not work quite as they should with a physics engine. For example, problems may arise due to numerical instability or inaccuracy in some situations.
- You'll be able to build your own physics engine, matched to your own needs.
- It's fun!

The bottom line, though, is that it should not be an either/or lifetime decision whether to choose between a physics engine and coding your own physics. Experiment with both, and you'll become better at both: being able to code up your own physics will give you a deeper understanding of how physics engines work and how to use them more effectively; conversely, experimenting with physics engines should give you plenty of ideas that you can implement into your own coding.

There are now a number of Flash physics engines out there. Here is a brief list of some of them:

- **Box2DFlash:** Also known as Box2DAS3, this is an ActionScript 3.0 (AS3.0) port of Box2D, an open source 2D physics engine written by Erin Catto in C++. Box2DFlash is primarily a rigid body library, but with a host of other features. It is a popular choice among Flash game developers. Because Box2DFlash closely resembles Box2D, users can get help from the Box2D forums.
- **FOAM:** Another 2D rigid body physics engine. In the words of its creator, FOAM trades efficiency for modularity and extensibility, allowing a savvy developer to extend and repurpose FOAM to his own ends. FOAM is released under the MIT license.
- **Fisix:** The Fisix engine is an AS3.0 verlet-based physics engine aimed for use in games and relatively CPU-intensive applications. The website pledges to give good documentation and help, and states that the engine is currently free for both commercial and noncommercial applications until the next release.
- **APE:** APE stands for *Actionscript Physics Engine*, another AS3.0 open source 2D physics engine written by Alec Cove and released under the MIT license.
- **WOW-Engine:** This is an AS3.0 open source 3D physics engine written by Jerome Birembaut. It uses the Sandy library for all its 3D math computations and handles collisions and physics by extending APE, the 2D physics engine created by Alec Cove (see the previous bullet).

- **JiglibFlash:** This is another AS3.0 open source 3D physics engine, ported from the C++ physics engine Jiglib. The engine has support for Away3D, Papervision3D and Sandy3D. Documentation appears sparse, but it seems to have an active forum.

A disclaimer: The selection and ordering of the list are essentially random and do not constitute a recommendation to use any of them in preference to any other.

What is physics?

Physics is the most fundamental of the sciences. In a broad sense, physics is the study of the natural laws that govern how things behave. More specifically, it concerns itself with space, time and matter (defined as any “stuff” that exists in space and time). One aspect of physics is to formulate general laws that govern the behavior of matter, its interactions, and its motion in space and time. Another aspect is to use these laws to make predictions of the way specific things move and interact—for example, the prediction of eclipses from the laws of gravity or how airplanes are able to fly from the laws of aerodynamics.

Physics is a vast subject, and in a book of this nature we cannot do more than scratch the surface. Fortunately, most of the physics that you will probably need to know fall within a branch known as *Mechanics*, which is one of the easiest to understand.

Everything behaves according to the laws of physics

Without getting too philosophical, it is fair to say that the laws of physics are truly universal, as far as physicists have been able to observe. What this means is that everything *must* behave according to physics. This is different from say, the laws of biology, which pertain only to living things. A stone thrown in the air, a planet orbiting the Sun, the workings of the human body, and the operation and motion of a man-made machine must all obey the laws of physics. Moreover, many seemingly diverse phenomena are governed by the same subset of laws. For example, a falling stone and a planet orbiting the Sun both obey the laws of gravity.

The laws can be written as math equations

The great thing is that the laws of physics can be written as mathematical equations. Okay, that may not sound too great if you don’t like math! But the point here is that for a law to be useful, it has to be made precise. And math equations are as precise as anything can be. There is no possible ambiguity in how to apply a law that is expressed mathematically, in contrast with the laws that are fought over in courtrooms! Second, this means that centuries of developments in mathematics prove to be applicable to physics, making it possible to solve many physics problems. Third, and what is of most relevance for us: math equations are readily convertible into code.

Predicting motion

Let's get more specific. As an ActionScript programmer, you are mostly interested in how things move. Much of physics deals with how things move under the action of different types of influences. These "influences" can be from other things and from the environment. In physics we have a special name for these influences: they are called *forces*. The really good news is that the forces have simple mathematical forms. Although the motion of objects is usually complicated, the underlying mathematical laws that describe the forces are usually quite simple.

The general relationship between force and motion can be written schematically as follows:

motion = function{**forces**}

This is a cause-and-effect relationship. Forces cause objects to move in different ways. In practical terms, it means this: specify the forces acting on an object and put them in a mathematical equation, and then you can calculate the motion of the object. Simple isn't it?

Motion is effect. Force is cause. The motion of an object is the result of the forces acting on it. The mathematical relationship between force and motion is known as the "Law of Motion."

Now to be able to put this principle to use, we need to know the following:

- **Definitions.** The precise definitions of *motion* and *force*.
- **The law of motion.** In other words, the precise mathematical form of the function that relates a force to the motion it produces.
- **Force laws.** In other words, how to calculate the forces. There are equations that tell you how to calculate each type of force.

So there are two kinds of laws you need to know about: laws of motion and force laws. You will also need to know the proper concepts (known as *physical quantities*) to describe and analyze motion and forces and the relationship between them. Finally, you will need to know the mathematics for manipulating and combining these quantities.

Programming physics

So, how do you code up physics? Do you program the motion or forces, or both? And what does it involve?

Once you know some basic physics (and some simple math), coding it is not much different or more difficult than what you are used to as a programmer—provided you do it in the right way. Let us take some time to explain what this "right way" is.