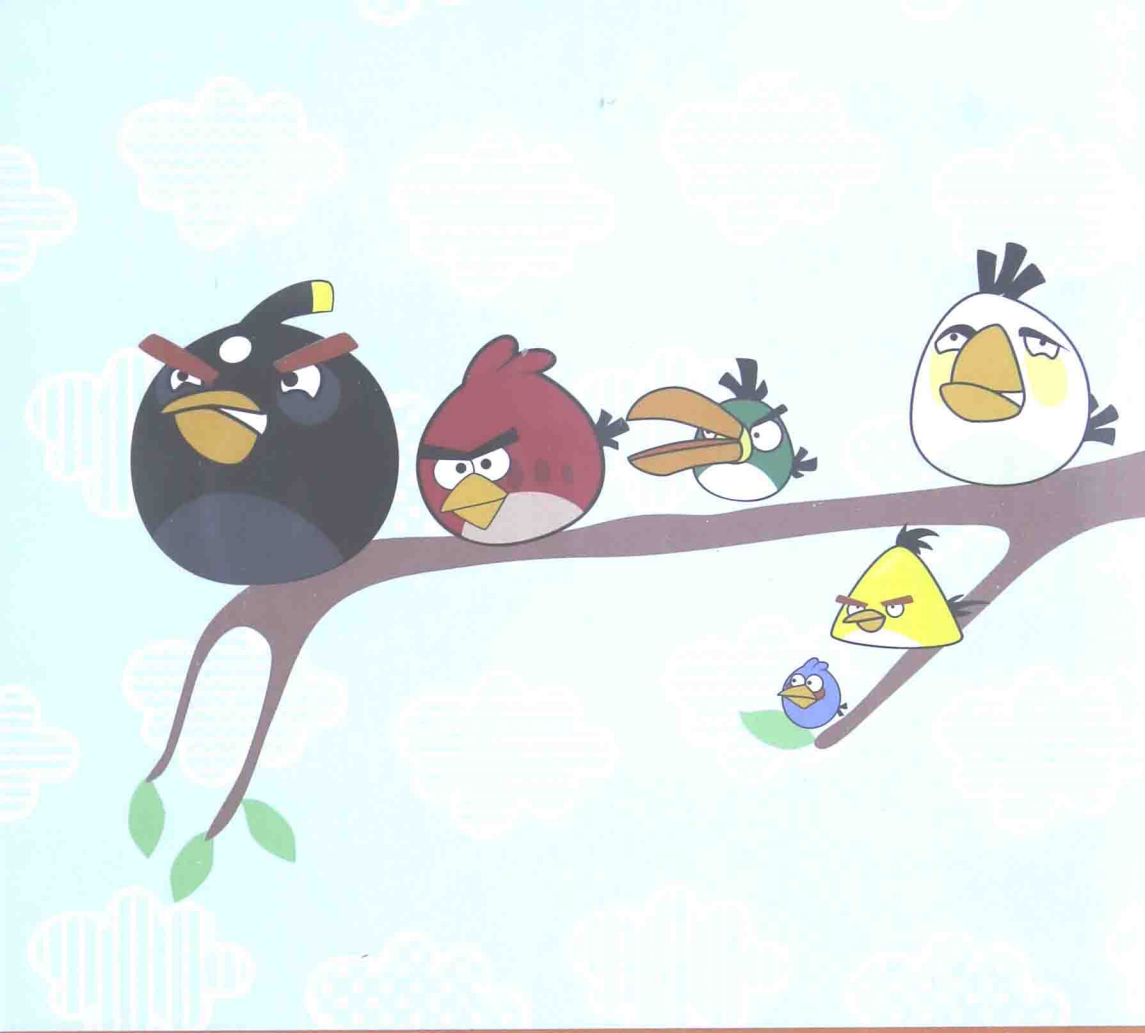

社区精华提炼

Box2D 物理游戏编程

初学者指南

陈文登 编著

再现经典游戏的技术细节：

《愤怒的小鸟》、《叫醒盒子》、《超越重力》、《星星大盗》、《割绳子》……

科学出版社

Box2D 物理游戏编程

初学者指南

陈文登 编著

科学出版社
北京

内 容 简 介

本书系统地梳理学习Box2D物理游戏编程的各个知识点，并通过图解、问答、举例等形式深入浅出地讲解初学者觉得晦涩难懂的概念、术语。最后，通过大量的游戏效果模拟，直截了当地介绍相关知识点的实际应用，这些游戏包括《愤怒的小鸟》《叫醒盒子》《超越重力》《星星大盗》《割绳子》等。

本书基于Box2D 2.3.0进行讲解，内容包含b2WheelJoint、b2MotorJoint、SetTangentSpeed、Raycast等网络上相对较少的教程，是目前为止市面上最全面、最详细的Box2D教程。

本书适合零基础、想继续深入学习Box2D物理游戏开发的读者，可作为游戏开发人员的参考书，也可用作高等院校相关专业的教学用书。

图书在版编目（CIP）数据

Box2D物理游戏编程初学者指南/陈文登编著.—北京：科学出版社，2015.4

ISBN 978-7-03-043434-0

Ⅰ.B… Ⅱ.陈… Ⅲ.游戏…软件设计 Ⅳ.TP311.5

中国版本图书馆CIP数据核字（2015）第036043号

责任编辑：喻永光 杨 凯 / 责任制作：付永杰 魏 谨

责任印制：张 倩 / 封面设计：付永杰

北京东方科龙图文有限公司 制作

http://www.okbook.com.cn

科 学 出 版 社 出版

北京东黄城根北街16号

邮政编码：100717

http://www.sciencep.com

中国科学院印刷厂 印刷

科学出版社发行 各地新华书店经销

*

2015年4月第 一 版 开本：720×1000 1/16

2015年4月第一次印刷 印张：20 1/2

印数：1—3 500 字数：600 000

定价：78.00元

（如有印装质量问题，我社负责调换）

致　谢

从起初开始执笔写这本书，到最终稿件的完成，中间经过了一年多的时间，在这段时间里，很多人都从不同的方面帮助我、支持我把这本书写完，我想应该在本书开始之前，由衷地对这些帮助过我的人表示感谢。

首先感谢我的每一位读者。从最初在“9ria 天地会”社区发布 Box2D 系列教程开始，就有大量的网友对我的教程表示肯定，并给予褒奖，也有部分读者玩笑式提议“你可以写一本书啦！”这句不经意的玩笑话，正是我写作本书的初衷。大家的不断支持，也成了我坚持下去的无限动力。

其次要感谢我的爱人和父母。在写作本书期间，我的家庭有了一次重大的升级——女儿出生了。然而就在她最需要父爱和母爱的时候，我却一头扎进了本书的写作中，爱人接过了我身上的担子，感谢她在我身后的默默支持！我不仅没有尽到一位父亲的责任，还当上了“啃老族”，父母悉心照顾我的饮食起居，全力配合我写作，父爱如山、母爱如水，感谢我的父母！

另外还要特意感谢两位素未谋面的外国友人——Emanuele Feronato 和 Chris Campbell，他们的个人网站 *www.emanueleferonato.com* 和 *www.iforce2d.net* 上发布了大量的精品文章，为我的写作带来了很多灵感，我也将秉承他们的这种无私精神，把知识分享给广大的读者和正在阅读的你。

最后，感谢出版社的每一位工作人员，让这本书最终沉甸甸地来到我们的手上。衷心地希望本书能为你的物理游戏开发之路带来一些帮助和启示。

前　言

在这个智能手机、平板电脑风靡世界的年代，与其坐在计算机前劳力伤神地打上一天的怪物，不如上下班路上、午休期间掏出手机玩一两局简单的益智游戏来得放松。基于 iOS 和 Android 两大主流移动平台系统，加入多样化的触屏体验，更趋于真实的交互也使得人机连接更紧密。

看看时下比较流行的游戏，《愤怒的小鸟》《捣蛋猪》《小鳄鱼爱洗澡》《猴子也疯狂》……大都是模拟真实物理碰撞类游戏，本书将向你介绍一个专门用于此类物理游戏开发的开源引擎——Box2D。它也是业界游戏开发者常用的一个物理引擎，可以帮我们可以省去复杂的物理公式计算，快速创建自己的物理游戏！

网上搜索一下 Box2D，相关开发教程层出不穷。Box2D 官方网站包含了正式的开发者使用手册，笔者也曾试着去学习这些教程，但总是找不到切入点，刚体啊、世界啊，毫无头绪，剪不断，理还乱。直到我看到了 Emanuele Feronato 网站，他的教程浅显易懂，让我真正入了门，进而有了“9RIA 天地会”的 Box2D 系列教程及本书，与各位初学者分享我的 Box2D 学习经验，在全面巩固自己的物理游戏开发技能的同时，希望也能对有类似开发需求的你带来些许帮助。

是否需要物理知识

学习编程，是不是数学要非常好？学习物理引擎，是不是要精通物理？很多初学者有过类似的顾虑，在没有 Box2D 之前，要做出“愤怒的小鸟”这样的游戏，肯定要有扎实的物理力学知识基础，而且还要用计算机语言表示出来。但是现在有了 Box2D，这个集成了所有物理运动、碰撞、反弹等各种现象的强大引擎，我们可以放下物理知识的思想包袱了。

适用对象

Box2D 物理引擎基于 C++ 开发，书中的教程是基于 Flash ActionScript 3.0 语言编写的。无论是 C++ 还是 ActionScript 3.0，它们都是面向对象的编程语言。在学习过程中，你会遇到类、对象、继承、静态方法等大量面向对象的概念和术语，所以你必须有一定的面向对象编程的基础。

而实际上，现在常用的编程语言基本都是面向对象的了，所以可以说学习 Box2D 没有任何门槛，如果你对物理游戏开发有着浓厚的兴趣，那就赶紧加入我们吧。

需要什么

Box2D 引擎

本书基于 Box2D 2.3.0 版 API 编写，请根据你所用的开发语言，在 Box2D 官方网

站 *http://box2d.org/links/* 下载对应版本的 Box2D 引擎。

注意：书中所用 ActionScript 3.0 对应的 Box2D 最新版本为 2.1a，笔者基于 C++ Box2D 2.3.0 版的 API 对 Flash Box2D 2.1a 版进行了扩充，扩充后版本为 Box2D 2.1a Plus。Flash 开发者在使用书中示例时，请以本书配套源文件中 *lib* 文件夹下的 Flash Box2D 引擎为准。

编译环境

虽然本书内容是基于 ActionScript 3.0 的，但重点在于 Box2D 中的各个类对象，以及相关 API 的讲解与说明，而这些内容在各语言版本的 Box2D 中都是相同的，受 ActionScript 3.0 语言限制并不多，所以你不必专门安装 ActionScript 3.0 的编译和调试环境。在自己所用开发环境下，跟随书中的教程编写代码，学习相关的 Box2D 知识，完全没有问题。因此，书中对 Box2D 引擎的安装与配置不做介绍，具体步骤请自行上网搜索。

浏览器

本书所附的示例文件，均为 ActionScript 3.0 编译生成的 *swf* 文件，如果你的电脑上安装了 PC 版 Flash Player，可直接开启查看。如果没有，可将 *swf* 文件直接拖至浏览器中打开查看。

阅读引导

针对初学者经常遇到的一些问题和疑惑，书中会以“小菜”的身份进行提问，由“大叔”来解答！

Q 小菜：我会问一些初学者经常问到的问题哦！

A 大叔：有求必应！

当你看到下面的格式时，说明这是一段代码：

```
when you see this format, that means this sentence is source code.
```

本书配套源文件中，另一类是文件名以“Whatis”开头的示例文件与程序源文件，为阐述性演示文件，目的在于演示相关知识点的动态效果，书中不对此类文件进行讲解，读者无须探究此类文件的实现方法。不过，如果你对演示效果感兴趣，在相应章节中同样可以找到对应的源文件。

目 录

第 6 章 关 节

第 7 章 Box2D 工具

第 8 章 游戏中的 Box2D 应用

附录 向量运算

第 1 章
认识 Box2D 世界

也许在这之前，你对 Box2D 没有太多的了解，那么让我带你认识一下 Box2D。

笔者有一个习惯，在玩游戏的时候，喜欢看游戏开发者信息，也就是游戏中常设的“关于”或“Credits”按钮，关注这些成功案例所使用的游戏引擎。在这个过程中，我发现越来越多的物理游戏都使用了 Box2D，其中也不乏像图 1.1 中这些我们耳熟能详、天天把玩在手中的游戏。

图 1.1　应用 Box2D 引擎实现物理模拟的游戏

实际上，Box2D 早已得到了业界的公认，并被广泛应用于 PC、Web、iOS、Android 等多平台的 2D 物理游戏开发当中。因此它也从最初的 C++ 版本，被广大的开发者们转译成了 C#、Java、JavaScript、ActionScript 等多个语言版本。随着移动游戏产业的兴起，各类游戏开发引擎不断推陈出新，Box2D 也随之繁衍，结合 Cocos2D、HTML5、EaselJs、Starling、Unity3D 等各大主流游戏引擎（图 1.2），开发出一个又一个优秀的作品。

益智类游戏是女生的最爱，有没有想过把《愤怒的小鸟》改编一下，把“小鸟”换成你心仪的女生，“猪头”换成你，然后用这个游戏作为定情信物送给她？想象一下：

图 1.2　支持 Box2D 的几大主流游戏开发引擎

她玩着“女友版”《愤怒的小鸟》，开心地笑着，感动地哭着……谁说程序员不会浪漫！还愣着干什么，快来学习 Box2D 吧！

本章将带领你进入 Box2D 的世界，初步认识一下这个强大物理引擎。你将会学到创建一个简单 Box2D 应用的基本流程，了解什么是 b2World 物理世界，这个世界里的基本作用力——重力，以及如何开启 Box2D 调试试图。

1.1 什么是 Box2D 引擎

Box2D 不是一种计算机编程语言，不需要又从变量、数据类型……一大堆让人头疼的基础知识学起；也不是一种游戏编程软件，不需要复杂的安装与配置。它是一个强大的开源物理游戏引擎，用来模拟 2D 刚体物体的运动和碰撞，是 Erin Catto 早在 2007 年用 C++ 语言开发的。

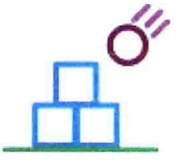

图 1.3 Box2D 的 Logo

Box2D（图 1.3）集成了大量的物理力学和运动学的计算，并将物理模拟过程封装到类对象中，将对物体的操作以简单友好的接口提供给开发者。我们只需要调用引擎中相应的对象或函数，就可以模拟现实生活中的加速、减速、抛物线运动、万有引力、碰撞反弹等各种真实的物理运动。

Q 小菜：报告，有问题！什么是开源？

A 大叔：开源就是开放源代码，任何人都可以看到甚至修改核心内容的代码，开源是一种分享精神，是一种人人为我、我为人人的奉献精神。

Q 小菜：那什么是引擎？还有什么框架，貌似很高深呢！

A 大叔：相信大多数初学者，都会有这样的疑问，对一些专业名词似懂非懂，看大牛们讲得口若悬河，小鸟们却看得云里雾里。引擎是集成了特定功能的一个工具类库，如我们现在学习的 Box2D，它的功能就是仿真物理运动。框架是集成了某些常用开发流程的标准化类库，如 MVC 框架，它将开发过程分成视图、数据和控制，可以用来开发游戏、网页、应用等，不局限于某一种功能。

Q 小菜：什么是刚体？

A 大叔：别着急，第 2 章会详细介绍刚体的知识。

下面我们来看看从哪里开始下口，吃掉 Box2D 这个大蛋糕：创建一个简单的 Box2D 世界。

1.2 创建 Box2D 世界

世界本是混沌的一体，盘古的神斧砍下去之后才有了天和地，形成了真正的世界，之后才有了山川、河流、花草、树木。Box2D 也一样，现在在我们的脑中也是混沌的一片，在创建物体或进行物理模拟之前，我们也要先创建世界。

Box2D 中用 b2World 类（图 1.4）来表示世界。它是 Box2D 的核心类之一，集成了 Box2D 对所有对象的创建、删除、碰撞模拟的相关接口。

那么，要创建什么样的世界呢？太阳系里的星球数不胜数，人类也已经成功登上了月球，但是我们不希望过着每天打个喷嚏都能弹出去的日子——还是地球住着比较安全，因为地球上有重力！

图 1.4 Box2D 中的 b2World

1.2.1 重 力

早在 1666 年，坐在树下的牛顿被掉下来的苹果砸到脑袋之后，向大家揭示了重力的存在（图 1.5）。2015 年的今天，在本书中，我们认识了 Box2D 中的重力。

图 1.5 苹果的掉落，让我们认识了重力

重力，简单地讲，就是使物体自然下落的力，Box2D 中用 b2Vec2 向量类来模拟作用力。b2Vec2 是“Box2D Vector2D”的缩写，表示二维空间中的一个向量，b2Vec2 类的构造函数中只有 x 和 y 两个参数，分别表示 x 轴和 y 轴方向上的分量。

```
function b2Vec2(
  x : Number,
  y : Number
  ):void
```

Q 小菜：打住！什么是向量？

A 大叔：向量是 2D 游戏中模拟几何运算时常用的一个概念，常用来表示物体的坐标和速度。Box2D 中大量模拟计算都是基于向量的，所以你要对向量有一定的认识。更多关于向量的定义和公式，请参考附录。

Box2D 中的重力同样是用 b2Vec2 对象表示的。我们知道重力都是垂直向下的，根据 Flash 中自左至右、自上而下的坐标系统（图 1.6），创建一个 b2Vec2 对象来模拟重力的代码通常是下面这个样子的：

```
var gravity : b2Vec2 = new b2Vec2(0, 9.8);
```

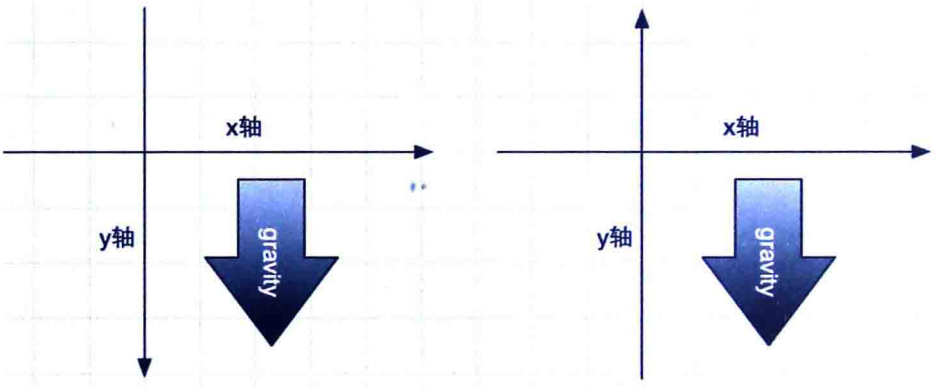

图 1.6 ActionScript（左）和 C++（右）坐标系统下的重力

不同语言环境下坐标系统的坐标轴方向可能不同，如 C++ 中的 y 坐标轴是向上的，那么模拟重力的 b2Vec2 创建代码应该是：

```
b2Vec2 gravity(0.0f, -9.8f);
```

Q 小菜：为什么模拟重力的代码中设置 x 为 0？而 y 是 9.8，不是 10 或者其他值。

因为重力是垂直向下的，在水平方向没有任何力，所以 x 设置为 0，这个不难理解。至于 y 的参数值 9.8，实际上，这里的 y 指的是重力加速度，我们读高中时都学过，理论重力加速度是 9.8 m/s²（米每平方秒）。Box2D 中的坐标系统的单位也是米（m）——Flash 中的像素 (px)。为了便于说明，这里设置为 9.8。在实际开发中，为了方便计算，通常设置为 10。

实际上，Box2D 中的重力 gravity 可以设置为任意方向和任意大小，来模拟不同朝向的“重力”。甚至可以设置为 new b2Vec2(0,0)，模拟出一个类似太空的完全失重状态，这也是利用 Box2D 开发俯视类游戏时，经常使用的设置。

A 大叔：注意！Box2D 中的计量单位是米（m），而目前大部分的平台渲染都是基于像素 px 的，所以在进行渲染计算时，要注意对坐标或形状尺寸进行转换，通常 1m=30px。假设 Box2D 中某个点的坐标是 (a,b)，那么它在渲染时，这个点的位置应该是 (a × 30, b × 30)。反过来讲，要在舞台的 (c,d) 位置绘制一个点，对应 Box2D 中的坐标位置应该是 (c/30,d/30)。

1.2.2 创建世界

“世界”在 Box2D 中用 b2World 类来表示。b2World 类的构造函数中有两个参数：一个是 b2Vec2 类型的 gravity，也就是上一节我们讲的重力；另一个是叫作 doSleep 的布尔值参数，表示 Box2D 引擎是否将静止不动的刚体设置为睡眠状态。

Q 小菜：sleep？睡眠？什么意思？

任何运动物体，因为摩擦力、能量转换等因素，最终都会慢慢停止运动。Box2D 中的刚体也是一样，正常情况下，Box2D 会遍历世界中的所有刚体，通过复杂的公式计算，模拟物理运动。但是对于静止不动的刚体来说，这时的物理运动模拟计算是没有必要的。当参数 doSleep 为 true 时，Box2D 会将静止的刚体标记为睡眠状态，在遍历时直接跳过，不进行运动模拟计算，这样就提升了 Box2D 的计算效率，节省 CPU 开支。

Q 小菜：又是一堆刚体，到底什么是刚体？

A 大叔：简单地讲，刚体就是 b2World 中的一个“东西”。具体我们会在第 2 章详细讨论。

讲完了这两个参数就可以创建 Box2D 的 b2World 世界了。在你的主程序中键入下面的代码：

```
private var world:b2World;
public function DemoSimpleBox2DWorld()
{
  var gravity:b2Vec2 = new b2Vec2(0,9.8);
  var doSleep:Boolean = true;
  world = new b2World(gravity,doSleep);
}
```

world 对象稍后还会用来创建和管理刚体等其他对象，所以要设置为全局变量。而 gravity 和 doSleep 在世界创建完成之后就基本用不到了，所以可以设置为局部变量，或者直接在 b2World 的构造函数中定义。

不过，b2World 并不是一个具体的物体，现在编译源文件是看不到任何效果的。

1.3 开启 Box2D 模拟

世界创建好了，下一步是让引擎动起来。当一组连续的画面按照固定的频率，连贯地出现在我们眼前时，就形成了动画，每个画面叫作一帧。每种语言中都有自己的帧频事件，或者计数器事件。我们在事件处理函数中不断更新某个对象的坐标，可以实现对象的移动。如 Flash 中经常用到的 Event.ENTER_FRAME 事件，在对应的事件处理函数中不断更新 rect 对象的 x 坐标，可以让它慢慢地动起来，如下面的代码所示：

```
stage.addEventListener(Event.ENTER_FRAME,loop);
private function loop(e:Event):void
{
  rect.x+=10;
}
```

Box2D 引擎模拟物理运动的原理也大致如此。每一帧基于大量的物理运动公式，计算出刚体的坐标、角度、速度等，并更新对象相应的属性。然后，不断地快速更新这些计算，形成动态的物理模拟效果。好在我们不用自己动手去编写这些复杂的公式，Box2D 已经把这个复杂的过程集成到了一个叫作 step() 的函数中。

在编写具体代码时，我们只要在 Event.ENTER_FRAME 或 TimerEvent.TIMER 事件处理函数中，调用这些复杂物理公式计算的接口 b2World.step() 就可以了。step() 方法的结构如下：

```
public function Step(
  delta:Number,
  velocityIterations:int,
  positionIterations:int
):void
```

- delta:Number ：Box2D 在进行物理模拟计算时，每次更新计算之间经过的秒数（或者说延迟）。或者简单地讲，在 Box2D 应用中，我们看到的画面都是前一个画面经过 delta 秒之后的结果。

A 大叔：注意，Box2D 中的计时单位是秒，不是帧！

程序在每次调用这个 b2World.step() 函数时也要经过一定的时间，假设这个时间是 interval 秒（或者你可以理解成每一帧持续的时间即 1/fps 秒），当 delta 小于 interval 时，

呈现出来的是一个慢放的效果；当 delta 大于 interval 时，呈现出来的是一个快放的效果，如图 1.7 和图 1.8 所示。所以在编写游戏时，保证 delta=interval 是很重要的。

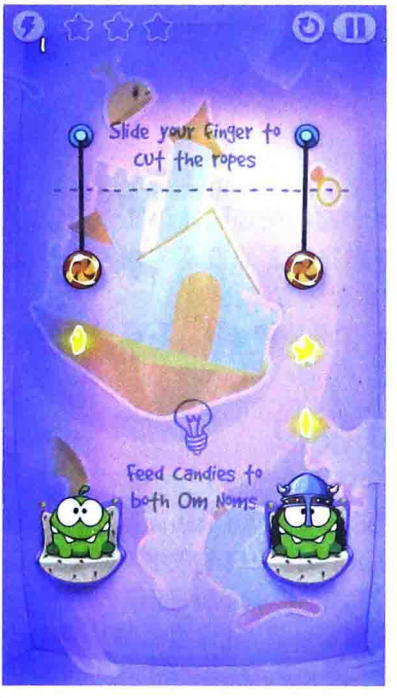

图 1.7 《割绳子》中的正常模拟速度

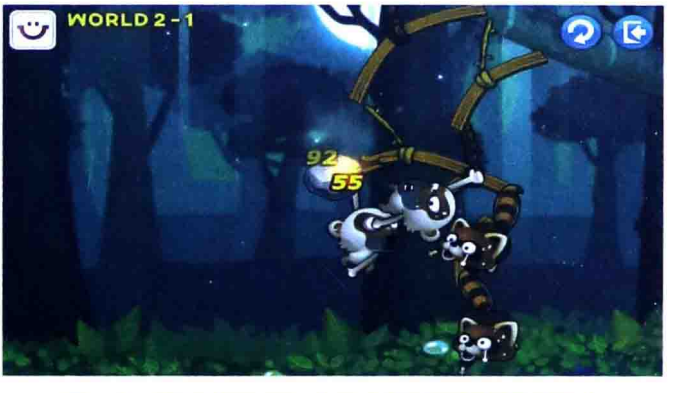

图 1.8 《疯狂的猴子》中敌人被打到时的慢放镜头

大部分的游戏引擎在渲染时，都是基于帧频（frame per second, fps）进行的，而且通常帧频要在 30 以上才能保证游戏的连贯和顺畅运行。此时两帧之间的时间间隔 interval=1/30 秒，那么这个 delta 参数就可以设置为 1/30。如果你的项目是基于 Timer 计时器驱动帧播放的就更简单了，Timer 的 delay 属性恰好指的是计时器延迟的毫秒数，delta 设置为 timer.delay*1000 即可。

开启本章示例文件 *WhatisDelta.swf*，舞台中有一个自由下落的小球、两个静止

的平台，以及一个吊着的挡板，如图 1.9 所示。示例中的帧频 fps=30，即 interval=1/30，初始情况下 step() 的 delta 参数也是 1/30，小球按正常速度下落；按下左方向键，delta 被设置为 1/90，在渲染经过了 1 s（即 30 帧）时，Box2D 只模拟了 1/3，因此形成了 1/3 慢放镜头效果；按下右方向键，此时 delta 被设置为 1/10，在渲染经过 1 s（即 30 帧）时，Box2D 却模拟了 3 s，形成了 3 倍快放镜头的效果。开启示例文件，查看动态的模拟效果。

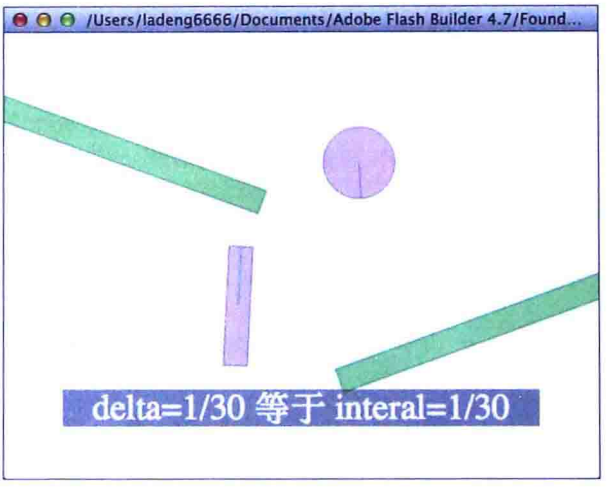

图 1.9 *WhatisDelta .swf* 中，不同 delta 模拟的快放和慢放效果

● positionDelta:int ：现实中两个物体发生碰撞时（如球和地面），为了缓解碰撞，物体都会发生一些形变。而 Box2D 中模拟的物体都是坚硬的“刚体”，无法产生形变，这时就会出现物体之间的重叠。当此类情况发生时，Box2D 会自动调用 b2ContactSovler 类的 SolvePositionConstraints() 函数，来矫正碰撞重叠。不过，这个函数通常无法一次性完全消除碰撞重叠，特别是当多个刚体之间发生堆叠时，就要再次调用 SolvePositionConstraints() 函数重复矫正（图 1.10）。这里，positionDelta 指的就是 Box2D 单次 step() 物理模拟计算中函数 SolvePositionConstraints() 的执行次数。

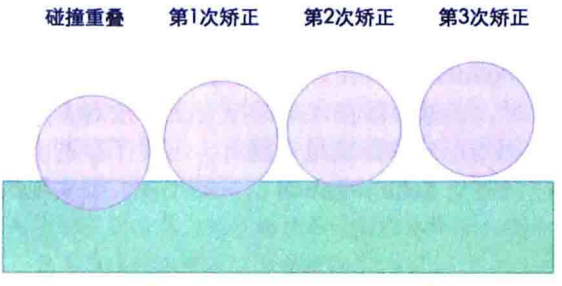

图 1.10 碰撞重叠矫正过程

本章示例文件 *WhatisPositionDelta.swf* 中，舞台上方会不断产生矩形刚体，并逐个堆叠在一起。按下空格键可以设置 positionDelta 在 1 和 10 之间切换。当多个刚体堆叠一起时，可以看到 positionDelta 为 1 时，刚体之间的重叠无法及时被矫正，出现了明显的挤压现象。开启示例文件，查看动态效果，如图 1.11 所示。

positionDelta 取值越大，单次 step() 物理模拟中碰撞重叠的矫正精度越高，但相应的所需资源消耗也会变大。通常设置为 10 即可满足游戏开发要求。

● velocityDelta:int ：当两个速度不同的运动物体发生碰撞时，根据物体的质量、速度大小和方向，速度会在碰撞物体上重新分配。和 positionDelta 类似，Box2D 也需要对这些不同的速度进行矫正，这个过程同样需要调用 b2ContactSolver 类的 SolveVelocityConstraints() 函数，进行多次矫正。velocityDelta 就表示单次 step() 物理模拟过程中，调用 SolveVelocityConstraints() 函数的次数——同样是一个 int 整型变量，

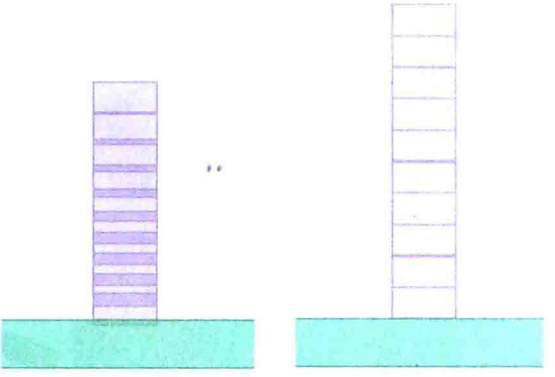

图 1.11 positionDelta 为 1（左）和 10（右）时的不同效果

取值越大，单次 step() 中的刚体速度矫正越精确，资源消耗越大，通常设置为 10 即可满足开发要求。

本章示例文件 *WhatisVelocityDelta.swf*，对取值为 1 和 10 的 velocityDelta 进行了对比，效果与图 1.11 类似，但碰撞叠加现象在起初两个刚体堆叠时就出现了。开启示例文件，查看动态效果。

另外，虽然通过设置不同的 delta 参数，可以实现镜头慢放和快放的效果，但是我们应当尽量避免 delta>interval，因为这样 Box2D 在前后两次 step() 中进行物理模拟计算的时长变大，物体移动的距离也比正常要远，很容易出现物体瞬间移动，而使得部分高速运动物体的碰撞无法有效检测到，出现穿透现象。

例如，在本章示例文件 *WhatisBugOfHighdelta.swf* 中，舞台上方会不断地落下小球，正常情况下，也就是 delta<=interval 时，小球会落在底部的平台上。按右方向键使 delta>interval，在模拟过程中，两帧之间的小球的距离越来越大，出现了穿透现象（图 1.12）。在实际的开发过程中，我们应当避免 delta>interval 情况的出现。要实现快镜头快放效果，应相应提高游戏的帧频。

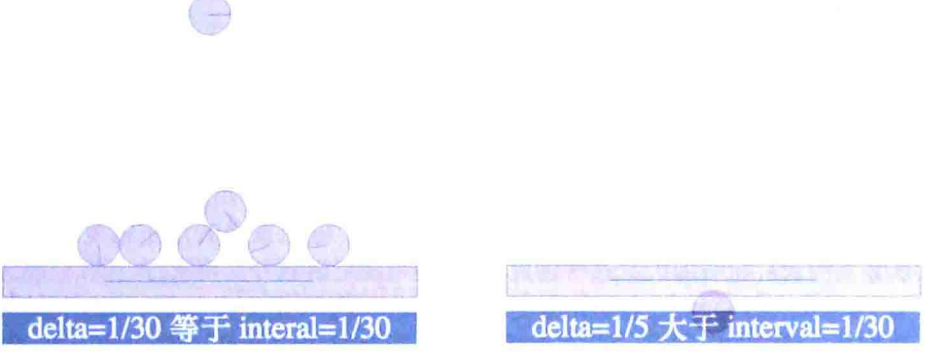

图 1.12 delta<=interval 时（左）模拟出来的正常碰撞，delta>interval 时（右）很容易出现穿透现象

所以，要让 Box2D 物理引擎顺畅地运行起来，只需要设置恰当的 delta、positionDelta 和 velocityDelta 参数，并在帧频事件处理函数或计数器事件处理函数中，不断地调用 b2World 对象的 step() 函数。

例如，在 Flash 中用 ActionScript 创建一个基本的 Box2D 应用程序，完整的源代码如下。你可以在本章示例文件 *SimpleBox2DWorld.as* 中找到对应的代码。

```
package
{
  import flash.display.Sprite;
  import flash.events.Event;

  import Box2D.Common.Math.b2Vec2;
  import Box2D.Dynamics.b2World;

  public class DemoSimpleBox2DWorld extends Sprite
  {
    private var world:b2World;
    public function SimpleBox2DWorld()
    {
      var gravity:b2Vec2 = new b2Vec2(0,9.8);
      var doSleep:Boolean = true;
      world = new b2World(gravity,doSleep);
    }

    protected function loop(event:Event):void
    {
      world.Step(1/30,10,10);
    }

  }
}
```

代码中的构造函数首先创建了 b2World 对象，并保存在全局变量 world 中，然后在 Event.ENTER_FRAME 事件处理函数中，不断地调用 world.step() 函数，开始进行物理模拟计算。你的第一个 Box2D 应用就这么完成啦！

Q 小菜：不对啊，我按照上面的代码一行一行的敲出来了，但是编译之后，怎么什么东西都没有啊？

A 大叔：别着急，现在的世界才刚刚从宇宙大爆炸中分离出来。苹果也没有，牛顿都还没出生，更不要谈什么重力、物理模拟了。

1.4 小 结

本章我们对 Box2D 引擎有了一个大体的了解，也初步学习了创建一个 Box2D 物理应用的主要步骤：

1. 定义 gravity 重力向量，以及 doSleep 是否将静止的刚体设置为睡眠状态；
2. 根据 gravity 和 doSleep 参数创建 b2World 对象；
3. 随帧频事件或计数器不断地调用 b2World.step()，让 Box2D 引擎运行起来。

这是一个好的开始，我们找到了学习 Box2D 的切入点，这也是值得庆祝的一大步。但现在不是休息的时候，要趁着一腔热情，继续学习。下一章，我们将用刚体在世界中添加一个掉落的苹果，一起感受牛顿发现重力时的喜悦！