•游戏开发经典丛书•

David Weller
(美) Alexandre Santos Lobão 著
Ellen Hatton
李 化 方小永 译

.NET 游戏编程入门经典
—C#篇

Beginning .NET Game Programming in C#

清华大学出版社

.NET 游戏编程入门经典

—— C#篇

(美) David Weller
Alexandre Santos Lobão 著
Ellen Hatton

李 化 方小永 译

清华大学出版社

北 京

EISBN：1-59059-319-7
Beginning .NET Game Programming in C#
David Weller, Alexandre Santos Lobão, Ellen Hatton
Original English language edition published by Apress L. P., 2560 Ninth Street, Suite 219, Berkeley, CA 94710 USA.

图书在版编目(CIP)数据

.NET 游戏编程入门经典——C#篇/(美)威勒(Weller, D.)，(美)罗伯(Lobão, A. S.), (美)海顿(Hatton, E.)著；李化，方小永译. —北京：清华大学出版社，2006.2
书名原文：Beginning .NET Game Programming in C#
ISBN 7-302-12104-4

I. N… II. ①威… ②罗… ③海… ④李… ⑤方… III. 游戏—应用程序—程序设计 IV. G899

中国版本图书馆 CIP 数据核字(2005)第 132956 号

出 版 者：清华大学出版社 **地 址**：北京清华大学学研大厦
http://www.tup.com.cn **邮 编**：100084
社 总 机：010-62770175 **客户服务**：010-62776969
组稿编辑：曹 康
文稿编辑：于 平
封面设计：康 博
版式设计：康 博
印 装 者：三河市春园印刷有限公司
发 行 者：新华书店总店北京发行所
开 本：185×260 **印张**：19.25 **字数**：493 千字
版 次：2006 年 2 月第 1 版 2006 年 2 月第1 次印刷
书 号：ISBN 7-302-12104-4/TP·7825
印 数：1～4000
定 价：39.00 元

前　言

游戏始于一个好的构想

尽管现在发布的游戏越来越倾向于图形化，但是游戏的主要特点“可娱乐性”，却在有些时候被遗忘了。

尽管游戏中那些惊险的图形画面、精彩的情景切换和 3-D 图形世界简直让我们眼花缭乱，但是它们有很多内容其实都不适合用户。有时候，即使一个游戏得到的反响很好，但游戏的可娱乐性却是不明确的或者是比较失败的。

您觉得旧版本的 Pac-Man 游戏怎么样呢？凭借其华丽的游戏环境，Pac-Man 游戏以及早期版本的 Mario Brothers on Nintendo 游戏在孩子们中间仍然十分受欢迎。

这里，我们的目的不是让您忘记所有的事情而过分地关注这些基本点。相反，我们希望您能够牢记：一个好的游戏总是始于一个好的构想，有时候，仅这一点就已经足够了。

引用最为普遍的一个游戏是 Tetris，它是由一个俄国编程人员单独开发的。这些年过去了，这个程序依然非常有意义，而且，我们也在本书的第 1 章复制了一个 Tetris 游戏程序——我们的“Hello World”程序。

当然，您可能会说“Tetris”只是成千上万个游戏程序中的一个，这一点我们也同意。但是如果您说“只有当一个游戏像 Tetris 那样的简单，一个人才能够比较好地单独创建这个游戏”，则我们将不同意您的这种观点。年长一些的人们可能还记得 Another World 游戏，此游戏具有一个续集，称为 FlashBack。该游戏具有的图形界面和音效对于那个时代而言非常好，并且它还具有漂亮的人物动画，以及各种各样的情景切换，这些情景切换通过“在不同视角显示游戏人物和奇异的世界”来完成游戏情节，而这个游戏也是由一个人单独设计的，设计者就是法国的一个程序员。

今天，我们可以找到很多关于业余程序员创建的游戏的网站。这些游戏中有些确实还不错，它们具有高质量的图形处理界面和音效；而且更重要的是，它们几乎都具有很好的可娱乐性，当然，这也许是因为他们只是喜欢创建和玩这些游戏，而不是为了赚钱。

本书将介绍很多提示和技巧来帮助开发人员单独设计自己的游戏。然而，如果您可以找到一些人来帮助您开发游戏的话，那还是请他(她)和您一起工作吧。

当然，除了一个好的构想之外，还有很多内容与游戏开发有关。

一个游戏并不仅仅是一个好的构想

尽管一个游戏必须始于一个好的构想，但是还有很多其他内容与游戏编程有关，而这些内容又往往超出我们的想象。现在，让我们来看一看在创建一个游戏项目时必须牢记的几点要求，即：

- 音效：尽管您也可以总是使用“嘟嘟”声等简单的音效来创建一个游戏，但是如果使用优美的背景音乐以及恰当的音效为游戏行为(射击、死亡、获取游戏得分，等)配音的话，则游戏效果会更好。即使您不打算在开发团队配备一个音乐方面的专家，但也不应该忘记，必须花大量的时间来寻找合适的环境音乐，以及在 Internet 或 CD 库的成千上万个音效中寻找最为合适的一个。
- 绘图：关于这一点，有一种不好的做法是：从别人的游戏中剪切下一些图像，然后把它们用到自己的游戏中。这样的做法将使游戏缺乏原创性，并且常常造成侵权事件。因为不是每个人都可以绘制精美图像的，所以最好还是在开发团队里配备一个(或者几个)艺术工作者。
- 色彩：在计算机上着色不同于在纸张上着色。因此，如果您的艺术工作者不会使用作图工具着色的话，则需要聘请一个会使用作图工具的艺术工作者。
- 动画：一个动画图形的创建与一个静态图形的创建有所不同。比如，任何人都可以绘制一棵效果逼真的树，但是如果想要绘制一个正在行走的人或者一只正在飞翔的小鸟的话，则需要那些具有动画制作经验的人来完成。即使您的游戏没有使用动画 sprite，但是也不要忘记：您可能还需要一个动态的游戏简介界面或者动态的情景切换。
- 代码：当然，如果没有这个内容，看到的将是关于游戏的书籍。
- 游戏级别设计：游戏级别的设计者是那些“确保游戏者能够选择到最优的可娱乐性和最好玩的游戏经历的”人。
- 质量保证：如果付不起钱聘请一个非常好的质量保证团队的话，那么最好还是不要进行游戏开发。对于一个游戏开发公司来说，最糟糕的事情莫过于一个错误众多的游戏程序。
- 项目管理：众多具有不同技术、不同性格的人在一起工作必需有一种有效的管理方式来保证他们中的每个人都保持最佳状态。即使是您自己单独开发一个项目，也千万不要轻视了好的项目管理方法的重要性，如：如果没有为项目设置一些“里程碑事件”来控制项目开发进度的话，则可能永远都在为该项目而工作，并且永远也看不到任何好的结果。关于如何管理项目，这方面的内容已经远远超出了本书的范围，但是，如果您从来没有在一个团队工作过的话，我们强烈建议您查看一下这方面的资料。最重要的是，您应该学会一些好的、有助于使您成为团队一部分的项目开发原则。
- 其他：还有很多有意义的内容，通常而言，必须做好准备以便解决所有难以预料的新问题。

如今，开发一个商业游戏绝不是一件简单的事情。那种“单凭一个程序员就可以成功开发新游戏并因此而发财的”时代基本上已经过去了。然而，我们还是要切记：这本书面向的对象是那些喜欢游戏设计的人、那些喜欢开发游戏并且以“让别人一起分享自己的思想”为乐的人。如果您希望创建一些专业性比较强的游戏，或者希望学习 Managed DirectX 技术的话，则这本书对于您而言确实只是一本较好的入门教程，但是，如果您真正想进入“游戏”这个领域工作的话，还必须学习更多的内容。

正如我们已经看到的那样，如今，要想开发一个一鸣惊人的游戏必须付出巨大的努力才可以做得到，但是不要被这些困难吓倒。记住：也许，您的游戏就是下一个“吃豆”、“俄罗斯方块”或者“飞行模拟器”。

切记：一个成功的游戏始于一个好的构想！

如何阅读本书

这本书是针对游戏编程具体实践的指导性书籍，为了从中获取最重要、最有用的价值，我们建议通过运行每一章的示例程序来学习该章的内容，这些示例程序的代码可以从本书合作网站 http://www.tupwk.com.cn 下载。使用 Visual Studio .NET 打开项目，然后编译并运行它。体验一下游戏，并仔细观察游戏的细节，从而使得您在阅读每一章内容之前，可以提前知道该章将要介绍的内容。

本书内容

本书中，我们创建了 4 个不同的游戏，这 4 个游戏跨越了好几章内容，另外在本书接近尾声的地方增加了一章附加内容。本书的代码也是按章节组织的，并且很多情况下是按照操作步骤递增的方式进行组织的。程序是采用 DirectX9.0(它是 2003 年夏季更新的 DirectX 版本)和 Visual Studio 2003 进行编写和测试的。还需要从 http://msdn.microsoft.com/directx 单独下载 DirectX SDK，并且如果决定使用不同的编辑工具的话，则必须使用“该工具能够支持的格式”来创建项目文件。完全可以使用.NET 和 DirectX SDK，以及一个简单的文本编辑器，如记事本(Notepad)，来编辑/运行所有这些游戏程序。但是，如果有可能的话，我们还是推荐使用 Visual Studio 或者其他更智能化的编辑器。

接下来的部分中，我们将对每一章的内容给出简短的描述。

第 1 章：.Nettrix：GDI+和冲突检测

在第 1 章内容中，我们引入了游戏中的冲突检测概念，介绍了一些简单的算法来管理游戏对象之间的冲突检测，然后引入了 GDI+的一些基本概念，以及一些图形库，这些图形库可以被.NET Framework 用来执行简单的图形操作。

本章，我们创建了一个 Tetris 克隆版的游戏.Nettrix，藉此来说明上述这些概念的使用。

第 2 章：.Netterpillars：人工智能和 Sprite

在本章中，我们回顾了面向对象编程的概念，并把相关的术语组织成一个术语表。我们也介绍了“创建游戏类库”的思想，这些类将在进一步的游戏开发工作中得到应用，以此来提高游戏程序的质量和推进游戏项目的开发进度。

在本章中，我们也对游戏中涉及到的人工智能进行了简要的介绍，说明了一些必须在游戏程序中进行处理的典型问题，并提出了一些如何解决这些问题的建议。本章的游戏示例是.Netterpillars，它是一个 Snake 克隆版的游戏程序，我们使用它来讨论本章介绍的这些概念。这里，我们介绍了如何创建一个可重用的类——一个基于 GDI+的 Sprite，它是本书中第一个可重用的类。

第 3 章：Managed DirectX 基础知识：Direct3D 基本概念、DirectX 与 GDI+

第 3 章介绍了 Managed DirectX 9.0，讨论了一些基本概念，如矩阵变换的使用、透明纹理的处理、有色光的处理等。这里，我们还讨论了如何基于游戏类型来选择图形库(DirectX 或者 GDI+)的问题。

本章没有任何游戏程序，只是包含一些简单的应用程序，我们将通过这些应用程序来说明本章所介绍的每个概念。

第 4 章：Space Donuts 游戏

在第 4 章中，我们讨论了如何使用 Managed DirectX 中的一个特殊类来创建 Sprite。我们也介绍了 DirectSound 和 DirectInput 的基本概念。

我们将使用本章所讨论的类和概念来引导读者创建一个 Asteroids 克隆版的游戏，即 Space Donuts。

第 5 章：Spacewar

这里，我们讨论了 DirectX 早期版本中所使用的“代码重写”技术，特别关注了 DirectDraw 库。另外，我们还介绍了 DirectPlay 的概念，它可以使您能够编写基于网络的多人游戏。

本章创建了 Spacewar 游戏的一个实现版本，Spacewar 游戏是第一个在计算机上开发成功的游戏，并且它至今仍具有很好的娱乐性。

第 6 章：Spacewar3D：网格、缓冲区和纹理

截止到本章内容之前，我们已经创建了 Spacewar 游戏，并进入了 Direct3D 世界。本章不但涵盖了许多新的 3-D 概念，而且还介绍了如何扩展 Spacewar 游戏的 2-D 版本程序代码。

第 7 章：为 Spacewar3D 添加视觉效果

本章深入讨论了如何使用 point sprite 编写游戏程序，这是一个相对高级的专题知识，但它也是一种可以产生特殊视觉效果的技术。

第 8 章：进一步学习游戏编程

第 9 章：附加内容：把.Nettrix 游戏移植到 Pocket PC 上

本章，我们讨论了程序员在“把游戏程序移植到不同的设备上”时所面临的问题，同时介绍了.NET Compact Framework。

使用这些知识，我们介绍了如何“通过将第 1 章创建的示例程序移植到 Pocket PC 系统上”来创建 Tetris 游戏的第二版本。

附录

为了让读者对“专业游戏开发人员如何考虑游戏的设计问题”有一个认识，我们在本书末尾附加了四个附录，其中三个附录取自三个专业人士所撰写的文章，他们都是在游戏领域长期工作的技术人员，另外一个附录是建议阅读的书籍。这四个附录是：

- 附录 A　参考文献
- 附录 B　游戏的动机
- 附录 C　我是如何设计游戏的
- 附录 D　开发成功游戏的指导方针

目　录

第 1 章　.Nettrix：GDI+与冲突检测

本章我们将介绍 GDI+的基本概念、Windows 系统下本地图形操作的扩展库，并讨论游戏开发中的最重要方面之一：冲突检测算法(collision detection algorithm)。尽管游戏开发者使用 GDI+函数在屏幕上绘制图像，然而图像之间的交互作用却是由冲突检测算法决定的。这将使得程序能够判断什么时候一个图像位于另一个图像之上，并采取合适的操作。比如：当一个球撞击一面墙时，球将产生弹跳行为。

为了达到这些目标并阐明这些概念，我们将演示如何创建一个名为.Nettix 的游戏。学习一门新的编程语言时，“Hello World”总是首先要编写的程序。学习编程游戏时，Tetris 是首先要尝试编写的最佳游戏程序。在这个简单的游戏中，您可以看到我们使用了许多基本概念，例如，基本的图形例程、冲突检测以及用户输入处理。

首先，您将接触到基本的 GDI+概念，了解到冲突检测算法的思想，藉此您将具备必要的技术背景来编写本章中作为示例的一个游戏程序(参见图 1-1)。

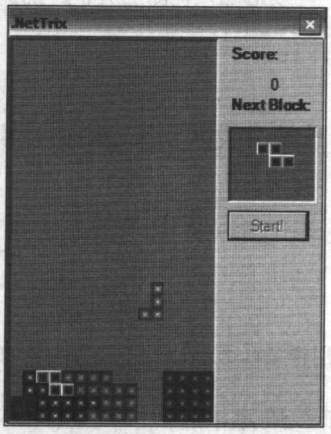

图 1-1　Nettrix，本章的游戏程序示例

1.1　GDI+基本概念

GDI+是新的.NET Framework 中基于类的二维图形、图像和版面的应用编程接口(API)。

与旧版本的 GDI 相比，GDI+具有一些实质性的改进，包括更好的性能、更多的功能，它甚至可以在 64 位计算机系统上运行，GDI+确实值得认真研究。关于 GDI+的新特点，我们将在以下章节进行讨论。

1.1.1　路径梯度

路径梯度(Path Gradient)允许程序使用梯度来更加灵活地填充二维图形，如图 1-2 所示。

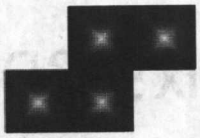

图 1-2 使用路径梯度

1.1.2 α 混合

GDI+是与 ARGB 色彩一同工作的，这意味着每一种色彩都是由红色值、绿色值和蓝色值混合并加上一个与透明度相关的 α 值来定义的。可以对透明度从 0(完全透明)到 255(不透明)进行赋值。0 到 255 之间的值使色彩在不同程度上部分透明，它们中的任一个值都将使得背景图像凸显出来。

图 1-3 显示了一个具有不同程度透明的长方形，如果把一个图像放在它下面，将会看到那个图像，就像透过玻璃观察一样。

图 1-3 在一个实心颜色位图上从 0 到 255 改变 α 值

1.1.3 基数样条函数

使用基数样条函数可以创建一些平滑线来连接给定的点集，如图 1-4 所示。

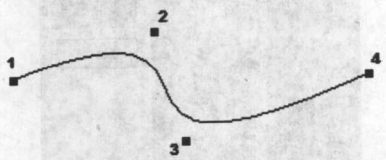

图 1-4 使用样条函数创建一条连接点集的平滑曲线

从图 1-4 中可以看到，该样条曲线确定了起点和终点(图 1-4 中，起点和终点用 1 和 4 标识)，另外两个点紧靠这条曲线，但样条曲线并不通过它们(点 2 和点 3)。

1. 使用 3x3 矩阵变换对象

当处理一系列图形变换时，应用变换(旋转、平移或缩放)方法非常有用，它可以加速图形变换的执行。图 1-5 显示了图形变换的一个示例。

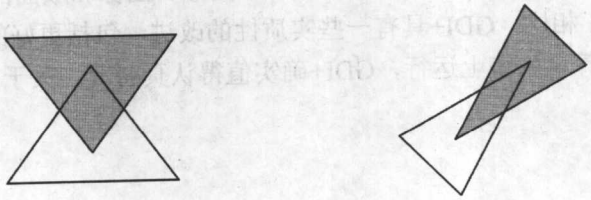

图 1-5 对图形实施旋转和缩放变换

2. 模糊

模糊用来对图形进行平滑处理以避免出现阶梯状外观，比如当对图像放大时。图 1-6 显示的是实施了该操作的一个示例。

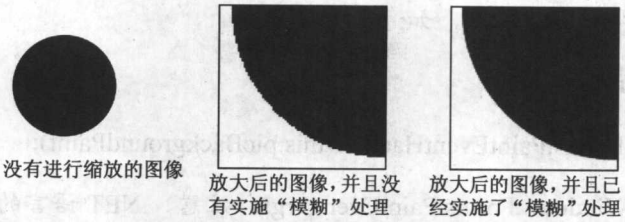

图 1-6 对图像实施模糊操作

注意

本书中，我们将给出 GDI+前两种新特点的示例：本章给出路径梯度的示例，下一章给出 α 混合的示例。对于 GDI+的其他特点，在.NET Framework 的 SDK 中有很多相应的示例代码。

1.2 使用 Graphics 对象执行图形操作

使用 GDI+的第一步总是先创建一个 Graphics 对象，它有助于图形操作的执行。Graphics 类提供了在特定的设备上下文中进行画图的方法。

有四种方法可以获得适当的 Graphics 对象：从使用 Paint 事件接收到的 e 参数获得、从窗口句柄中获得、从图像中获得，或者从指向设备上下文的特定句柄中获得。这些不同的方法实际上没有本质的区别，可以根据程序需要使用其中任何一种方法。例如，如果在窗体的 Paint 事件中编写画图函数的话，则可以使用 e 参数获得 Graphics 对象。但是，如果希望编写一个类实现在窗体上画图的话，则可能需要使用窗口句柄来创建 Graphics 对象。在接下来的章节中，我们将讨论每一种获得 Graphics 对象的方法。

1.2.1 使用 PaintEventArgs 参数创建 Graphics 对象

这种情况下，所有关于画图的代码必须与目标图像对象的 Paint 事件相关联。下面的代码展示了如何在屏幕的(10，20)(用像素表示)位置上绘制一个简单的红色长方形，长方形的高是 7(像素)、长是 13(像素)。

```
private PicSourcePaint(Object sender, Windows.Forms.PaintEventArgs e) {
    e.Graphics.FillRectangle(new SolidBrush(Color.Red), 10, 20, 13, 7);
}
```

注意

在代码段的前几行，可以看到.NET 如下的事件处理特点：

C#的每个事件处理过程都至少接收到两个参数：发送者对象(它是产生该事件的对象)和一个与该事件有关的对象(即 EventArgs 对象)。

事件处理过程通过事件与方法关联(通常在 InitializeComponent 方法中)而与对象相联系。它们的关联是通过+=操作符实现的，如：

this.PicSource.Paint += new

System.Windows.Forms.PaintEventHandler(this.picBackgroundPaint);

e 参数的类型是 Windows.Forms.PaintEventArgs。注意，.NET 语言的所有内容都集中在称作命名空间的托管代码单元里。在这个例子中，我们使用了 System.Windows.Forms 命名空间，该命名空间包含那些用于创建基于 Windows 的应用程序的类，这些应用程序使用了 Windows 操作系统的特征。在这个命名空间里，我们使用了 PaintEventArgs 类，它主要给出了 Paint 事件，以访问待更新的长方形数据结构(ClipRectangle 属性)和用作更新操作的 Graphics 对象。

Graphics 类和 SolidBrush 类由 System.Drawing 命名空间定义。该命名空间还有一些其他类，这些类提供了所有用于二维图形绘制、图像控制和版面设计的函数。在代码示例中，我们使用红色色彩(使用 Color 数据结构)属性创建了 SolidBrush 对象，使用该对象绘制了一个长方形，并使用 Graphics 对象的 FillRectangle 方法对长方形进行填充。

1.2.2 使用窗口句柄创建 Graphics 对象

为了在 GDI+中创建图形图像，我们必须使用一个句柄来指向窗口(window)的可拖拽部分。这个句柄就是 Graphics 对象，它可以通过 Graphics.FromHwnd 方法(Hwnd 意为“Handle from a window”)获得。在示例代码中，Graphics.FromHwnd 是 System.Drawing.Graphics.FromHwnd 方法的快捷方式，该方法在给定句柄下创建一个 Graphics 对象，该对象用于在特定窗口或特定控件里进行图形绘制。本段代码涉及到一个名为 picSource 的 pictureBox 控件。

```
Graphics graph = new Graphics();
graph = Graphics.FromHwnd(picSource.Handle);
graph.FillRectangle(new SolidBrush(Color.Red), 10, 20, 13, 7);
```

1.2.3 从图像中创建 Graphics 对象

FromImage 方法是从一个给定的图像中创建 Graphics 对象，方法的使用如下所示：

```
Graphics graph = new Graphics();
graph = Graphics.FromHwnd(picSource.image);
graph.FillRectangle(new SolidBrush(Color.Red), 10, 20, 13, 7);
```

注意，上述代码只有当一个有效位图图像加载到了 pictureBox 控件时才可以正常执行。如果试图在一个空的 pictureBox 控件或者使用索引像素格式的图像(如 JEPG 图像)加载到 pictureBox 控件来执行上述代码的话，程序将会出错，将不会创建 Graphics 对象。

1.2.4 从一个指向设备上下文的特定句柄创建 Graphics 对象

类似前面提到的方法，在给定了特定设备上下文句柄的情况下，使用 Graphics.FromHdc 方法可以创建一个允许程序进行拖拽操作的 Graphics 对象。使用 GetHdc 方法，可以从另一个 Graphics 对象获得指向设备的句柄，如以下代码片段所示：

```
public void FromHdc(PaintEventArgs e) {
   // Get handle to device context.
   IntPtr hdc = e.Graphics.GetHdc();
   // Create new graphics object using handle to device context.
   Graphics newGraphics = Graphics.FromHdc(hdc);
   newGraphics. FillRectangle(new SolidBrush(Color.Red), 10, 20, 13, 7);
   // Release handle to device context.
   e.Graphics.ReleaseHdc(hdc);
```

1.3 创建梯度

在前面的章节中，我们给出了一些通过 SolidBrush 对象创建红色实心长方形的代码示例。GDI+允许程序员使用特殊的梯度画笔来创建线性和路径梯度以产生丰富的色彩，从而避免了程序只是使用单调色彩的情形，梯度画笔可以带来非常有趣的效果。

GDI+具有创建水平的、垂直的和倾斜的线性梯度的特点。可以创建色彩变化均一(默认行为)的线性梯度，也可以使用梯度画笔的 Blend 属性创建色彩变化不均一的梯度。

下面的示例代码展示了如何创建一个色彩变化均一的梯度画笔，并使用该画笔从左上角到右下角绘制了一个色彩从红色变换到蓝色的长方形。

```
Graphics graph;
Drawing2D.LinearGradientBrush linGrBrush;

graph = Graphics.FromHwnd(picSource.Handle);

linGrBrush = new Drawing2D.LinearGradientBrush(
    new Point(10, 20),              // Start gradient point.
    new Point(23, 27),              // End gradient point.
    Color.FromArgb(255, 255, 0, 0),    // Red
    Color.FromArgb(255, 0, 0, 255))    // Blue
graph.FillRectangle(linGrBrush, 10, 20, 13, 7);
```

注意

该示例代码最重要的部分是使用 Color 对象的 FromArgb 方法来定义色彩。正如我们所看到的那样，GDI+的每一种色彩都是由四个值定义的：红色值(red)、绿色值(green)和蓝色值(blue)(即 RGB 值)，它们由标准的 GDI 函数使用；另外一个值是 α 值，它定义了色彩的透明性。在前述的示例中，我们对两个色彩值均使用了 255 的 α 值，因此它们将全部是不透明的。如果使用 128 的 α 值，将会产生半透明的色彩，因而任何隐藏在长方形下面的图形都将会透过该长