# 第15章 在游戏中添加对Game Center的支持

到目前为止，学习的内容都是关于如何开发单机游戏的。本章将介绍苹果官方提供的社交游戏网络Game Center、使用Game Kit API启用Game Center、如何使用Leaderboards（排行榜），以及如何使用Achievements（成就系统）。

为了方便学习，本章将使用苹果官方提供的示例项目GKTapper进行示范，并对其中的代码做适当调整。GKTapper项目的下载地址如下：

http://developer.apple.com/library/ios/#samplecode/GKTapper/Introduction/Intro.html

## 什么是Game Center

随着Facebook平台游戏的持续火爆，游戏的社交化已成大势所趋。苹果公司（以下简称苹果）当然不会错过这样的机会，从iOS 4.1开始，苹果推出专属的社交游戏网络Game Center。

### Game Center的功能

Game Center的功能由三部分实现：

1. 苹果在iOS 4.1及以上版本中向用户提供了系统级别的Game Center应用，玩家可以使用此应用创建自己的账号。使用该应用，玩家可以设置状态消息、添加好友、查看好友经常玩的游戏、邀请好友共同玩一个游戏。通过自动匹配启动一个多玩家游戏，比较玩家的成就，跟踪玩家的得分等。
2. 苹果向开发者提供了Game Kit的相关API，使用这些API，开发者可以创建支持Game Center的游戏。通过这些API，游戏可以对玩家进行验证，并获取好友信息，报告并显示得分，解锁成就系统，提供多玩家功能，设置管理玩家间的语音聊天。
3. 苹果同时提供一个在线的Game Center服务，该服务支持玩家验证，提供排行榜和成就系统数据，提供对多玩家游戏的自动匹配功能，以及其他的更多服务。

由于SDK版本的限制，早期的苹果移动设备并不支持Game Center。支持Game Center的设备包括：iPod Touch 2代~4代、iPhone 3GS、iPhone 4、iPhone 4S、iPad、iPad 2。总之，只要支持iOS 4.1的设备都支持Game Center。

### Game Center的特点

使用Game Center有以下好处：

* 给游戏添加Game Center功能非常简单。在没有Game Center之前，给游戏添加在线排行榜或成就系统，需要涉及大量的网络编码工作，同时还需要开发服务器端的应用。而使用Game Center，只需简单的调用几个API即可。
* 社交游戏系统可以增加玩家黏度，并吸引他们重返游戏。除了简单的通关，很多玩家想要更深入的体验游戏，可以给他们提供各种成就系统。排行榜功能也很不错，因为它会吸引玩家重新玩游戏，以便获得更高的分数，从而击败朋友！
* Game Center可以帮助开发者卖出更多的应用。支持Game Center的游戏会在App Store中提供一个专区，玩家可以通过Games选项卡上的Find Game Center Games直接找到支持Game Center的游戏。
* 在Game Center中添加的功能越多，就会吸引越多的玩家体验游戏，并分享给自己的好友。

关于Game Center更详细的介绍，可以参考苹果的官方文档：

Getting Started with Game Center：http://developer.apple.com/devcenter/ios/gamecenter/

## 在项目中添加对Game Center的支持

在本节内容中将学习如何在应用中启用对Game Center的支持。和本书中用到过的其他API不同，要启用Game Center需要按照以下的步骤进行：

1. 注册iOS 开发者计划账号；
2. 为应用创建并设置App ID；
3. 在iTunes Connect中注册应用；
4. 开启对Game Center的支持；
5. 如果需要，在iTunes Connect中添加成就系统；
6. 如果需要，在iTunes Connect中添加积分排名系统；
7. 在项目中添加相关的代码。

本节将介绍前6个步骤的相关内容。

1. 注册iOS开发者账号

为了在应用中启用Game Center，需要在iTunes Connect和iPhone Developer Portal中注册应用。以上两点都需要成为iOS开发者计划的注册会员。

为了完成本章的学习，并实际体验Game Center，首先注册苹果的iOS开发者计划，并为此支付一定数额的美元。注册iOS开发者计划的地址是<http://developer.apple.com/programs/ios/>。

1. 为应用创建并设置App ID

一旦注册iOS开发者计划成功，下一步就可以为应用创建并设置一个App ID。所谓的App ID是应用的唯一标示符，iOS通过App ID判断两个应用是否相同。

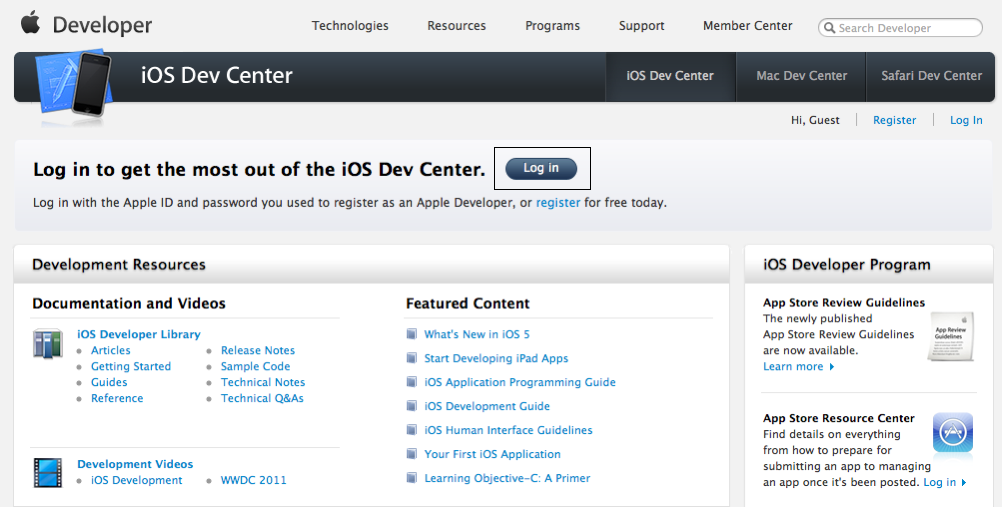
App ID通常分成两个部分：

1. **Bundle Seed ID**：由苹果生成的10字符随机字符串。如果有两个应用使用相同的App ID，那么它们会共享一个Keychain Access（钥匙串访问，在iOS中存储信息的一种安全方式）。如果不确定是否需要该信息，可以对所有的应用使用相同的Bundle Seed ID，以便今后访问钥匙串时使用。
2. **Bundle Identifier**：由开发者指定的字符串。虽然可以指定任何的字符串，但通常会使用网站域名作为前缀，以免和其他开发者输入的字符串冲突。当然，在该字符串的后面最好输入和应用名称相关的内容。

了解了App ID的构成，就可以给自己的应用创建一个App ID了。以下是相关的步骤。

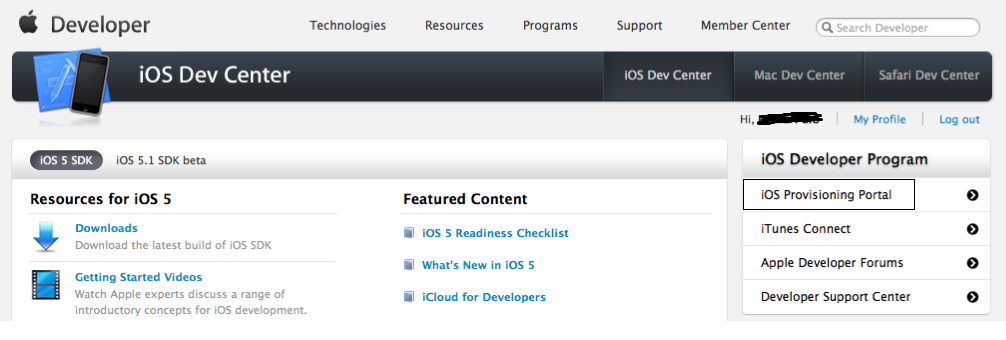
#### 登录iOS Dev Center

登录<http://developer.apple.com/devcenter/ios/index.action>，单击“Log in”按钮，并输入自己的开发者账号和密码，如图15-1所示。



1. iOS开发者中心的登录页面

成功登录后，如图15-2所示。



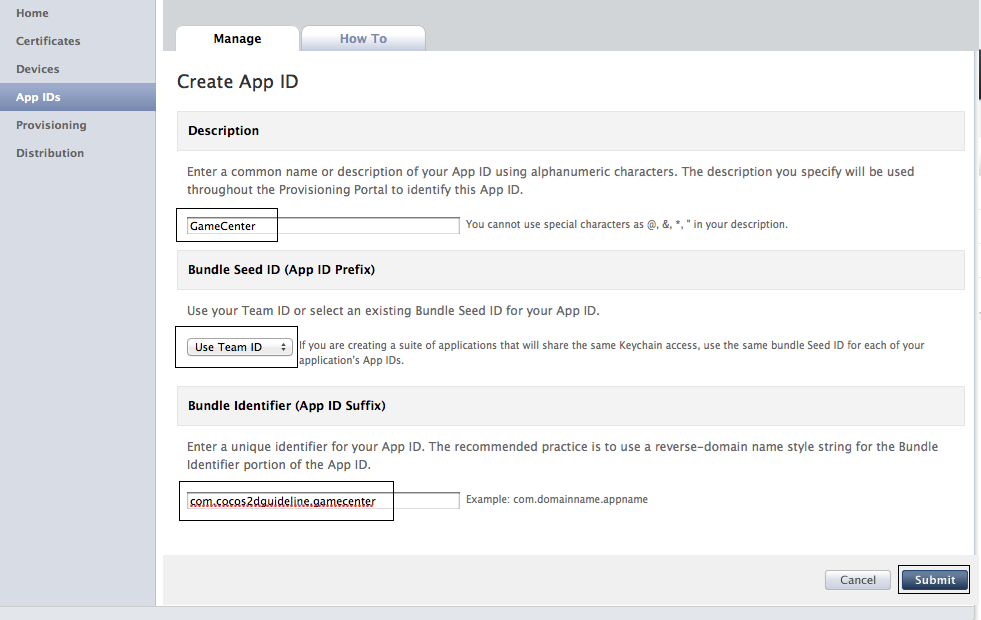
1. 成功登录iOS开发者中心

#### 创建App ID

单击图15-2右侧的“iOS Provisioning Portal”，在新打开的界面单击左侧的App IDs →New App ID按钮。按照以下的提示为应用创建一个新的App ID：

* Description：输入有助于区别App ID的信息，比如Game Center。
* Bundle Seed ID：选择和之前应用相同的Bundle ID，以便共享钥匙串访问，或者选择Team ID。
* Bundle Identifier：输入一个独特的名称，比如“com.cocos2dguideline.gamecenter”。如果没有域名，可以用自己的名字全称或其他独特名称。

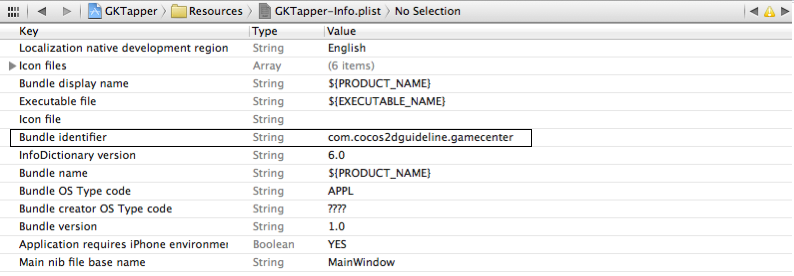
完成后单击“Submit”，如图15-3所示。



1. 创建App ID

#### 设置项目的Bundle Identifier

完成创建App ID的工作后，使用Xcode打开GKTapper项目，选择Resources→Info.plist，将Bundle identifier的内容修改为App ID的第二部分（10个随机字符串之后的部分），这里是“com.cocos2dguideline.gamecenter”，如图15-4所示。



1. 在info.plist中设置Bundle identifier

现在已经成功设置了App ID，接下来就是在iTunes Connect中注册应用，并启用Game Center。

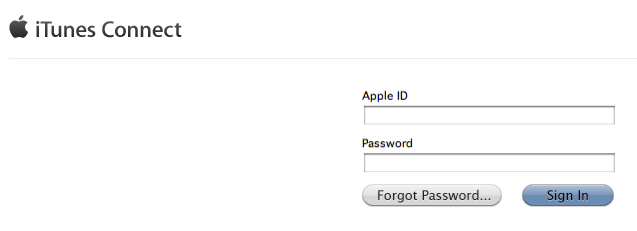
1. 在iTunes Connect中注册应用

要使用Game Center的特性，必须先在iTunes Connect中注册应用，然后编写和Game Center相关的代码。

以下是在iTunes Connect中注册应用的详细步骤。

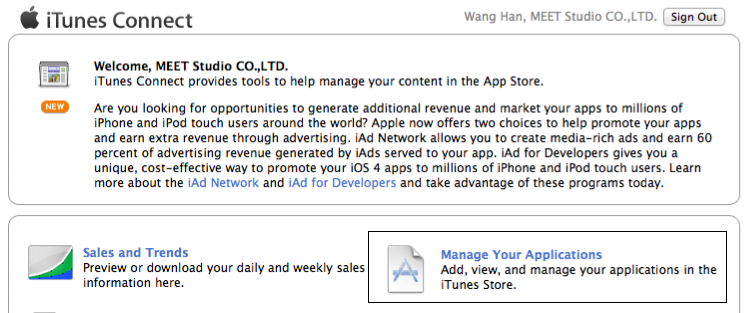
#### 访问iTunes Connect

登录<https://itunesconnect.apple.com>，iTunes Connect的登录界面如图15-5所示。



1. iTunes Connect的登录界面

使用账号登陆，如图15-6所示。



1. 登录成功界面

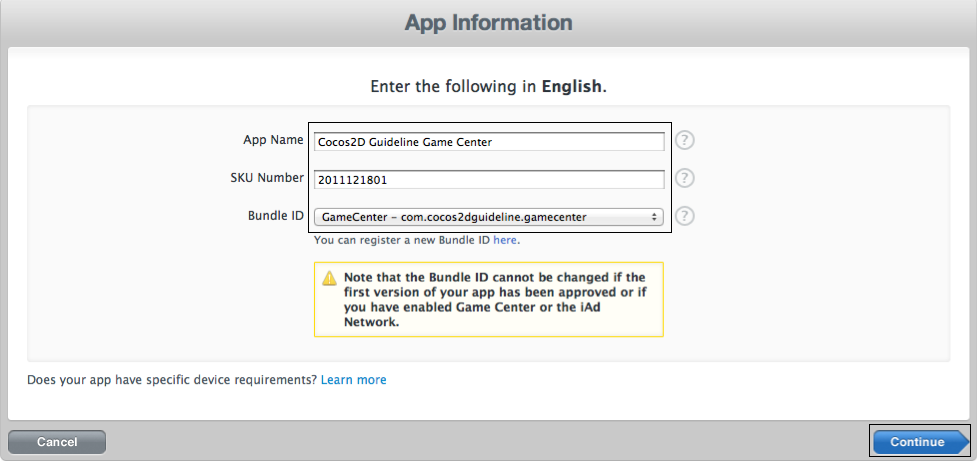
#### 应用管理

单击图15-6中的“Manage Your Applications”进入该模块。单击左上方的蓝色按钮“Add New App”。

接下来请根据以下说明填写相关信息：

* App Name：选择一个独特的应用名称，App Name和项目的名称不必相同。
* SKU Number：可以输入自己习惯的一种编号方式。
* Bundle ID：选择刚才使用的Bundle Identifier。

完成后如图15-7所示，输入完成后单击“Continue”。

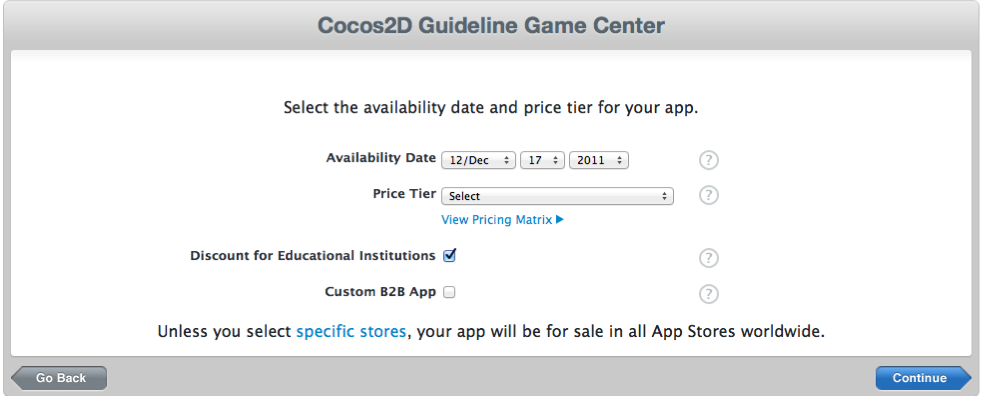


1. 输入应用的相关信息

#### 设置价格

紧接着会看到Rights and Pricing页面，如图15-8所示。

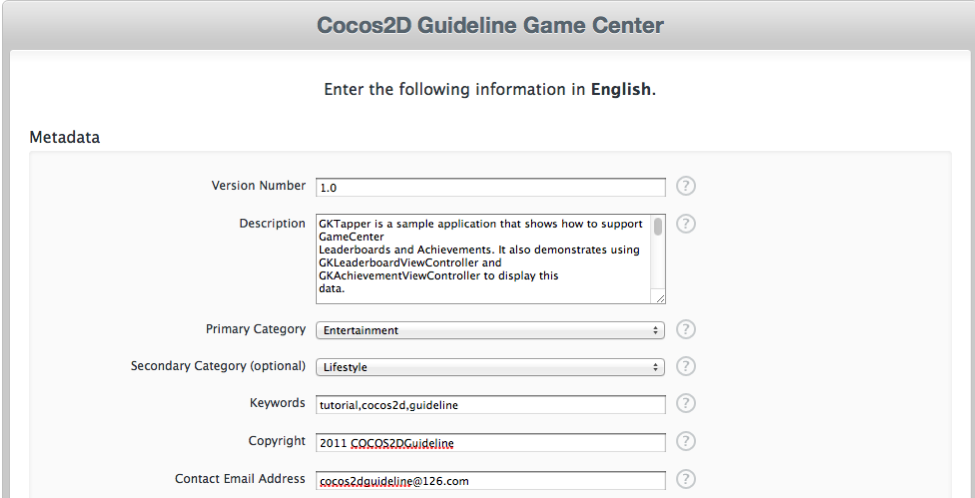
这里可以先随便选择一个价格，在正式提交审核前都可以对这些信息进行修改。信息输入完成后，单击“Continue”。



1. 选择应用的价格

#### 完善应用信息

Version Information页面如图15-9所示。

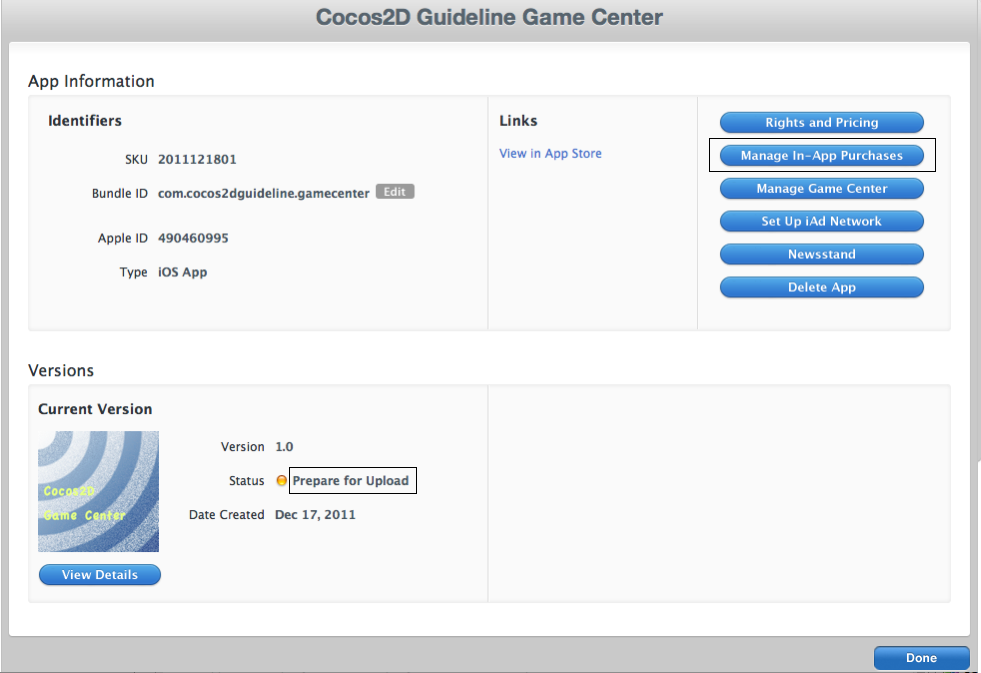


1. 输入更多和应用相关的信息

在以上需要填写或选择的地方输入一些临时性的信息，包括版本编号（Version Number）、应用的简单描述（Description）、应用的主分类（Primary Category）、应用的次分类（Secondary Category）、应用的键词（Keywords）、版权说明（Copyright）、联系邮件（Contact Email Address）等。

注意 这里还需要提供一个512\*512的临时图标，以及至少一张iPhone、iPod Touch或iPad的截图。

信息输入完成后，单击“Save”。如果一切顺利，会看到应用处于“Prepare for Upload”状态，如图15-10所示。

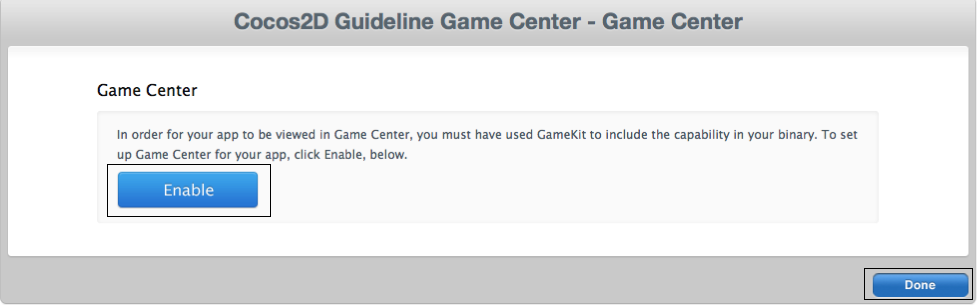


1. 应用处于“Prepare for Upload”状态

至此，在iTunes Connect已经成功注册应用。

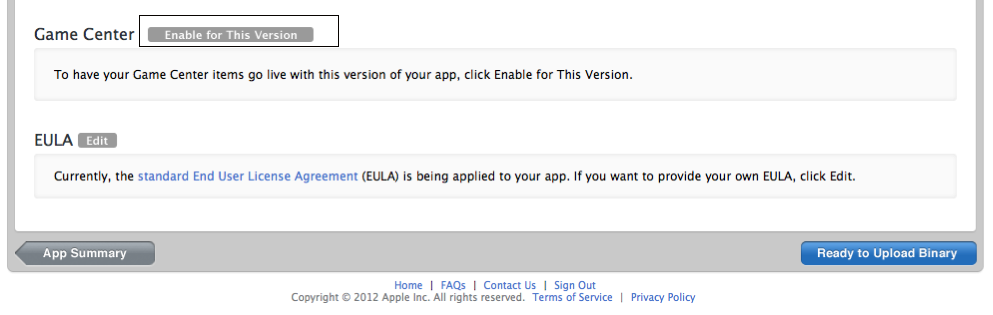
1. 开启对Game Center的支持

单击图15-10所示页面中右上角的蓝色Manage Game Center按钮，得到界面如图15-11所示，单击Enable按钮，然后单击Done。



1. 设置对Game Center的支持

此外，还要返回图15-11所示页面，单击View Details按钮，在页面下方单击Enable for This Version按钮，以启用对Game Center的支持，如图15-12所示。



1. 启用对Game Center的支持

到这步为止，应用已经启用了对Game Center的支持。

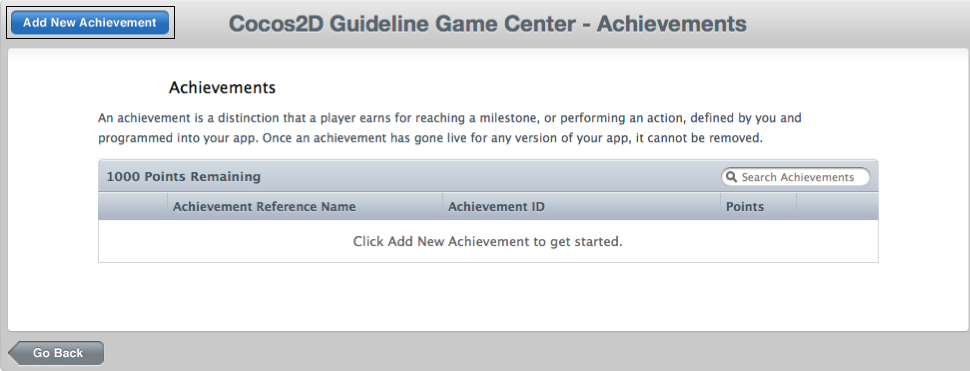
1. 在iTunes Connect中添加成就系统

通过成就系统，可以给玩家设置更多的挑战，比如跑100米、获得300个金币、消灭6个敌人或者满生命值过关等方式。通过记录玩家的成就，可以向用户提供分值或者称号的奖励。玩家也可以第一时间了解到他们已获取的成就，以及通过努力可以获取的成就。具体设置怎样的成就，不仅仅是开发人员的工作，更需要和游戏策划和设计人员紧密配合。

在项目中实现成就系统之前，必须在iTunes Connect中进行相应的设置，具体步骤如下：

#### 创建Achievements

登录地址<https://itunesconnect.apple.com>。单击“Manage Your Application”，选择自己的应用，单击“Manage Game Center”，然后单击Achievements下面的“Set up”，此时会看到信息如图15-13所示。

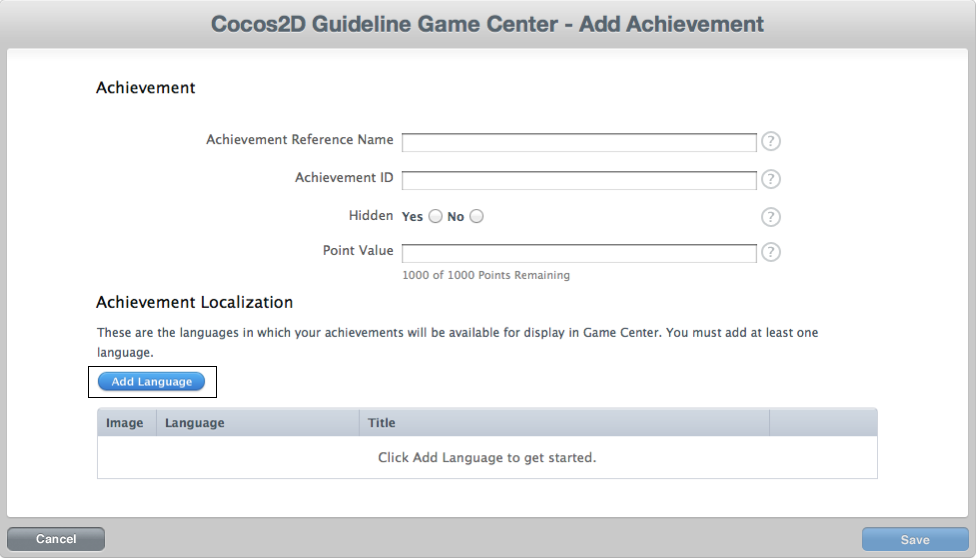


1. Achievements的设置界面

注意 以上有文字提示“1000 Points Remaining”，这是因为在设置成就系统的时候，会给每个成就指定一定的分值。不论要添加多少成就，最后的总分值不能超过1000分。因此，为了确保今后可以添加更多的成就，最好先使用较小的分值，从而给以后留下余地。

#### 填写Achievement信息

单击图15-13中的“Add New Achievement”，打开Add Achievement的页面如图15-14所示。



1. 设置新的Achievement

每处信息的含义如下：

* Achievement Reference Name：此处需要输入一个字符串，用于内部记录成就（玩家永远不会看到这个信息）。但需要注意的是，该名称在自己所开发的所有应用中都必须是独一无二的。
* Achivement ID：此处需要输入一个独特的ID，用于记录成就。和Bundle Identifier类似，可以在此处输入任何信息，但最好使用一个可以确保独一无二的名称。注意，这里的ID必须和项目中AppSpecificValues.h的宏定义一一对应。
* Hidden：如果选择Yes，那么玩家在获得该成就前无法知道游戏中有这样的成就设置。而如果选择No，则玩家可以在获得该成就前就知道有这样的设置。如果想要给玩家带来意外的惊喜，可以选择Yes，而如果想要一开始就吸引玩家去解锁更多的成就，可以选择No。
* Point Value：该处需要输入成就所对应的分值，前面对此做过说明。

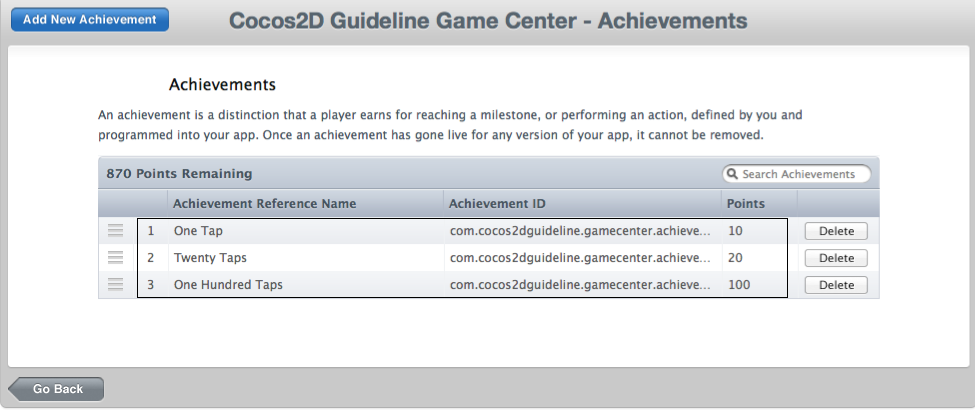
填写上述信息，单击图15-14中的“Add Language”蓝色按钮，输入至少一种语言所对应的信息。单击该按钮后，会提示输入信息：成就的名称、在用户获取成就前后的描述，以及对应该成就的512\*512分辨率大小的图片。

注意 在Game Center中，成就所对应的图像是圆形的徽章，有点类似硬币。可以直接制作并上传一个圆形徽章，也可以制作一个方形图像，Game Center将自动添加圆边和背景色（如果该图片含有透明度信息的话）。

#### 创建成就

这里分别创建了3个成就，分别是单次触碰、20次触碰和100次触碰。

设置完成后，单击“Save”，如图15-15所示。



1. 设置完成后的Achievements管理界面

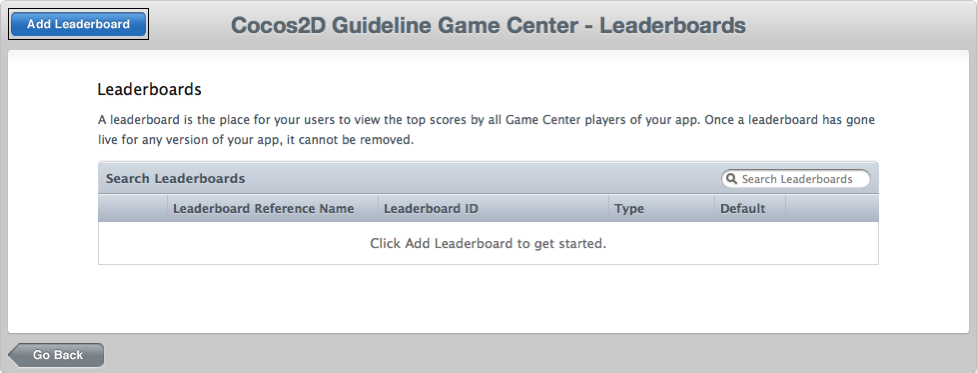
如图所示，Achievement Reference Name和Achievement ID下面的内容都是开发者自行创建的不同成就及其对应的编号。

1. 在iTunes Connect中设置积分排行榜

设置排行榜和设置成就系统的流程是相似的。具体的操作会按照以下步骤进行：

#### 创建Leaderboards

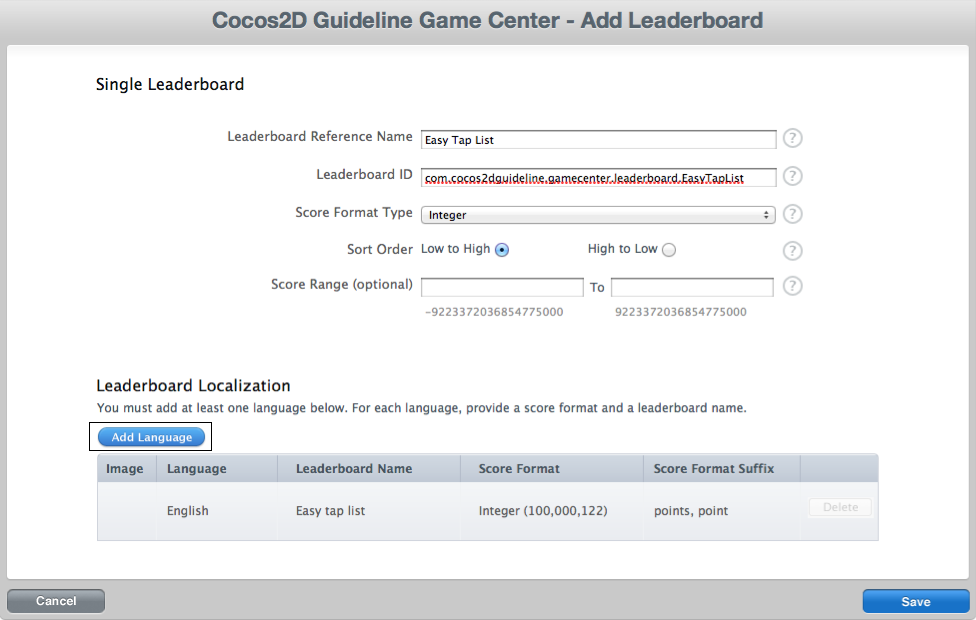
登录地址<https://itunesconnect.apple.com>。单击Manage Your Application，选择需要添加排行榜的应用，单击Manage Game Center，然后单击Leaderboards下面的Set up，此时页面如图15-16所示。



1. Leaderboards的设置界面

#### 填写Leaderboard信息

单击图15-16中的“Add Leaderboard”以添加新的排行榜。此时会让你选择Single Leaderboard或是Combined Leaderboard，这里需要选择Single Leaderboard。单击Add Language以设置其他的选项。设置完成后，单击Save以保存该排行榜，如图15-17所示。



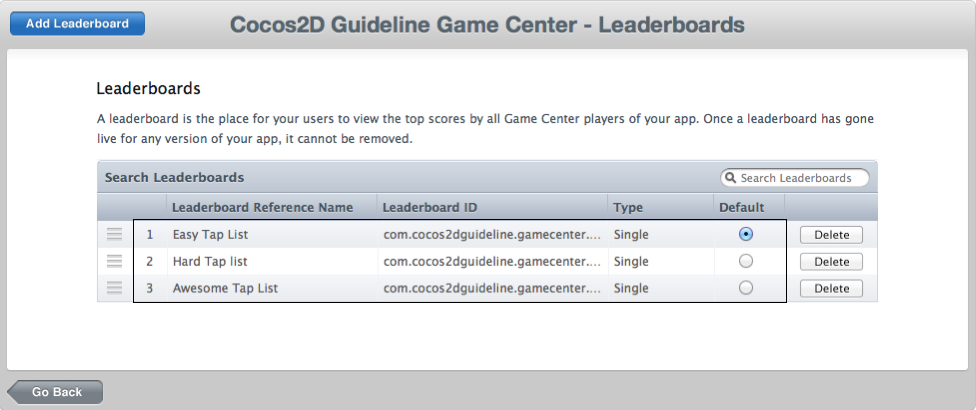
1. 设置新的Leaderboard

输入以下设置信息：

* Leaderboard Reference Name：此处仅是一个内部名称，用于记录排行榜，与Achievement Reference Name类似。本例为“Easy Tap List”。
* Leaderboard ID：此处的信息和Achievement ID类似，需要输入排行榜的独特名称，可以输入类似com、mydomain、mygame、leaderboard、escape等名称。注意，这里的ID必须和项目中AppSpecificValues.h的宏定义一一对应。
* Score Format Type：排行榜的数值通常是整数值，但也允许使用不同的方式指定整数值的格式。
* Sort Order：选择“Low to High”。
* Score Range：可选项，如果填写，必须是灰色数字的范围以内。

#### 添加排行榜

添加两个新的排行榜，“Hard Tap List”和“Awesome Tap List”。完成后如图15-18所示。



1. 设置完成后的Leaderboards管理界面

至此，已经为项目成功设置3个新的排行榜。

## 示例项目GKTapper详解

在iTunes Connect中完成相关的设置之后，就可以在项目中添加对应的代码实现对Game Center特性的支持。

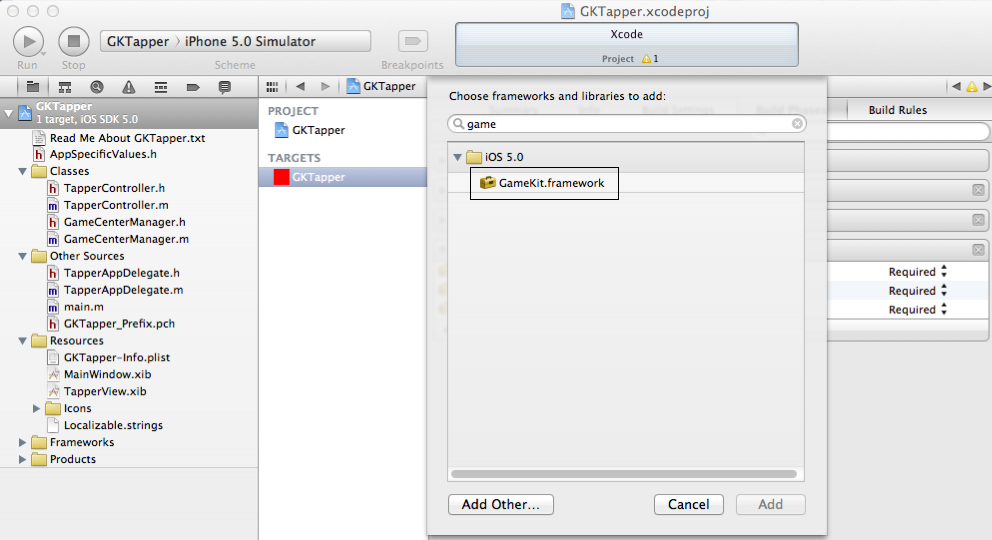
为了帮助开发者在项目中支持Game Center的特性，苹果官方提供的示例项目GKTapper中提供了GameCenterManager类。使用GameCenterManager.h和GameCenterManager.m可以帮助开发者轻松实现用户验证、积分排行榜和成就系统。开发者可以直接在项目中使用GameCenterManager类。

本节将详细介绍GameCenterManager类及其使用方法。

1. 添加Game Kit框架

首先在项目中添加Game Kit框架，该类库中包含Game Center API的相关代码。

在Xcode中打开项目，单击TARGETS下的项目名称“GKTapper”，在右侧单击“Build Phases”选项卡，单击Link Binary with Libraries下三角以扩展该选项。单击对话框底部的“+”按钮，选择iOS 5.0\GameKit.framework，单击“Add”。如图15-19所示。



1. 在项目中添加Game Kit框架

然后在Link Binary with Libraries的展开项中找到GameKit.framework，将类型设置为Optional（而非Required），

在Xcode中打开GKTapper项目后，会看到GameCenterManager.h和GameCenterManager.m这两个文件。

在Xcode中切换到GameCenterManager.h，其中的代码如代码清单15-1所示。

1. GameCenterManager.h内容

#import <Foundation/Foundation.h>

@class GKLeaderboard, GKAchievement, GKPlayer;

@protocol GameCenterManagerDelegate <NSObject>

@optional

- (void) processGameCenterAuth: (NSError\*) error;

- (void) scoreReported: (NSError\*) error;

- (void) reloadScoresComplete: (GKLeaderboard\*) leaderBoard error: (NSError\*) error;

- (void) achievementSubmitted: (GKAchievement\*) ach error:(NSError\*) error;

- (void) achievementResetResult: (NSError\*) error;

- (void) mappedPlayerIDToPlayer: (GKPlayer\*) player error: (NSError\*) error;

@end

@interface GameCenterManager : NSObject

{

NSMutableDictionary\* earnedAchievementCache;

id <GameCenterManagerDelegate, NSObject> delegate;

}

//This property must be attomic to ensure that the cache is always in a viable state...

@property (retain) NSMutableDictionary\* earnedAchievementCache;

@property (nonatomic, assign) id <GameCenterManagerDelegate> delegate;

+ (BOOL) isGameCenterAvailable;

- (void) authenticateLocalUser;

- (void) reportScore: (int64\_t) score forCategory: (NSString\*) category;

- (void) reloadHighScoresForCategory: (NSString\*) category;

- (void) submitAchievement: (NSString\*) identifier percentComplete: (double) percentComplete;

- (void) resetAchievements;

- (void) mapPlayerIDtoPlayer: (NSString\*) playerID;

@end

在该文件中，首先导入GKLeaderboard、GKAchievement和GKPlayer三个类，然后定义一个GameCenterManagerDelegate协议，并在其中提供了6种可实施的方法。接下来定义GameCenterManager类，其中包含两个属性：一个可变词典属性用于存储已获取的成就缓存，以及一个协议对象。最后定义一个类方法和6个实例方法。其中，类方法用于判断当前设备是否支持Game Center，6个实例方法分别用于验证当前用户、将得分提交到Game Center、加载最高得分、将成就提交到Game Center、重置成就，以及将用户ID映射到另一个用户ID。

下面根据项目中的实际用途对这些方法加以介绍。

1. 检查Game Center是否可用

Game Center只支持iOS 4.1及更高版本。因此，如果要在应用中支持Game Center，一种可选方案是禁止应用在iOS 4.1之前的版本中运行。要实现这一点很简单，在编译设置选项中将应用的Deployment Target设置为iOS 4.1及以上版本即可。这样做无疑是最简单的方法，但问题在于，使用iOS 4.1之前版本的玩家将无法体验这款应用，也会因此失去很多的潜在玩家。

通常选择另外一种方式，就是在设计应用或游戏的时候，不要过度依赖Game Center。同时在尝试使用Game Center的特性之前先检查当前设备上是否可用。通过这种方式，代码仍然可以使用在iOS 4之前版本的设备上。

为此，需要加入一些代码用来检查Game Center是否可用。

代码清单15-2显示了GameCenterManager.m中所使用的检测方法。

1. GameCenterManager.m中所使用的检测方法

// 检查当前设备是否支持 Game Center API

BOOL isGameCenterAvailable

{

//检查GKLocalPlayer API是否存在

Class gcClass = (NSClassFromString(@"GKLocalPlayer"));

//检查当前设备是否支持iOS 4.1或更高版本

NSString \*reqSysVer = @"4.1";

NSString \*currSysVer = [[UIDevice currentDevice] systemVersion];

//比较当前设备的系统版本是否高于4.1

BOOL osVersionSupported = ([currSysVer compare:reqSysVer options:NSNumericSearch] != NSOrderedAscending);

return (gcClass && osVersionSupported);

}

在这个方法中，首先检查GKLocalPlayer类文件（Game Center API类中的一个）是否存在。然后检查当前设备的系统版本是否高于或等于4.1。如果两个条件都符合，就可以在应用中放心的使用Game Center。

经过上面的检测，应用支持Game Center，接下来需要对玩家进行验证。

1. 验证玩家是否登录Game Center

GameCenterManager.m中提供以下方法对玩家进行验证，代码如代码清单15-3所示。

1. GameCenterManager.m中对玩家进行验证的方法

- (void) authenticateLocalUser

{

if([GKLocalPlayer localPlayer].authenticated == NO)

{

[[GKLocalPlayer localPlayer] authenticateWithCompletionHandler:^(NSError \*error)

{

[self callDelegateOnMainThread: @selector(processGameCenterAuth:) withArg: NULL error: error];

}];

}

}

请注意，以上方法使用了completion handle（完成句柄），当玩家成功登录，或出现某种错误时，将执行一整块代码。

注意 如果之前没有接触过这种blocks(块语句)，请不要恐慌。这种语法形式看起来很怪异，但一旦对此熟悉，就可以使用块来保持代码的紧凑性。学习块语句的最佳方式是阅读苹果的官方指南（http://developer.apple.com/library/ios/#featuredarticles/ Short\_Practical\_Guide\_Blocks/）。

根据苹果官方的指南，需要在应用启动时尽早调用一次authenticateWithCompletionHandler。此时会完成以下3个任务：

* 如果玩家已经登录Game Center，会在应用的上面显示一个Welcome Back对话框。
* 如果玩家还没有登录Game Center，并且没有选择不使用Game Center，会提示玩家使用现有账户登录，或创建一个新账户。
* 如果玩家还没有登录Game Center，并且已经连续3次选择取消使用Game Center，那么应用中不会显示任何对话框，而且以后也是！玩家后续将需要自行打开Game Center登录。之所以会出现这种情况，是因为Game Center被设置为系统登录行为。

注意 禁止再次调用authenticationWithCompletionHandler。如果玩家想要再次登录Game Center，必须切换到苹果官方提供的Game Center应用完成此工作。可以参考苹果开发者论坛中的相关讨论https://devforums.apple.com/thread/71830。

在authenticateLocalUser方法中调用callDelegateOnMainThread方法，并使用processGameCenterAuth为参数。

callDelegateOnMainThread方法的相关代码如代码清单15-4所示。

1. callDelegateOnMainThread方法

- (void) callDelegate: (SEL) selector withArg: (id) arg error: (NSError\*) err

{

assert([NSThread isMainThread]);

if([delegate respondsToSelector: selector])

{

if(arg != NULL)

{

[delegate performSelector: selector withObject: arg withObject: err];

}

else

{

[delegate performSelector: selector withObject: err];

}

}

else

{

NSLog(@"Missed Method");

}

}

- (void) callDelegateOnMainThread: (SEL) selector withArg: (id) arg error: (NSError\*) err

{

dispatch\_async(dispatch\_get\_main\_queue(), ^(void)

{

[self callDelegate: selector withArg: arg error: err];

});

}

以上方法中出现的dispatch\_async或许令人困惑，为什么调用它呢？这行代码的作用在于确保在主线程运行块代码。为一个消息通知注册回调方法时，除非手动指定，并不能保证该方法会在主线程运行。由于该方法需要修改对象中的变量，因此应在主线程中运行，以避免多线程读取问题。dispatch\_async是解决此问题的简便方法，不过它仅适用于iOS 3.2及更高版本。

而processGameCenterAuth则是GameCenterManagerDelegate中定义的可选方法，在TapperController.m中可以找到该方法的实现，如代码清单15-5所示。

1. processGameCenterAuth方法代码

- (void) processGameCenterAuth: (NSError\*) error

{

if(error == NULL)

{

[self.gameCenterManager reloadHighScoresForCategory: self.currentLeaderBoard];

}

else

{

UIAlertView\* alert= [[[UIAlertView alloc] initWithTitle: @"Game Center Account Required"

message: [NSString stringWithFormat: @"Reason: %@", [error localizedDescription]]

delegate: self cancelButtonTitle: @"Try Again..." otherButtonTitles: NULL] autorelease];

[alert show];

}

}

以上方法中如果验证成功，会通过一个GameCenterManager类的实例对象调用reloadHighScoresForCatory方法，并使用currentLeaderBoard字符串作为参数。该方法的作用是加载当前排行榜的分数。如果验证失败，则会弹出一个警告框，提示用户输入Game Center账号。

注意 在GKTapper项目中，当启动项目后，首先会在viewDidLoad中调用isGameCenterAvailable方法，以判断当前设备是否支持Game Center。如果不支持，会弹出一个警告框，说明该应用必须支持Game Center特性才能继续；如果支持，则创建一个GameCenterManager的实例对象，将代理设置为自身，并调用authenticateLocalUser方法进行用户验证；如果验证通过，继续下一步，调用updateCurrentScore方法。

1. 实现成就系统

之前已在iTunes Connect中添加成就系统，本节首先说明在项目中实现成就系统的工作原理。

实际上非常简单，就是利用一个方法判断玩家是否已达到该成就的要求，如果达到该成就的要求，就会调用GameCenterManager类中的submitAchievement:方法，将成就提交到Game Center。

#### 提交成就

submitAchievement:方法的代码清单如代码清单15-6所示。

1. submitAchievement:方法

- (void) submitAchievement: (NSString\*) identifier percentComplete: (double) percentComplete

{

//当提交成就的时候，GameCenter 会检查是否属于重复的成就

//但如果开发者只想向用户提供新的成就

// 则需要检查该成就在提交前是否已达到

// 否则就会引起 loadAchievementsWithCompletionHandler

和reportAchievementWithCompletionHandler之间的紊乱状态

//为避免这种情况的出现，需要获取当前的成就列表

// 然后将其存入缓存，并使用任何新的成就将其更新

if(self.earnedAchievementCache == NULL)

{

[GKAchievement loadAchievementsWithCompletionHandler: ^(NSArray \*scores, NSError \*error)

{

if(error == NULL)

{

NSMutableDictionary\* tempCache= [NSMutableDictionary dictionaryWithCapacity: [scores count]];

for (GKAchievement\* score in scores)

{

[tempCache setObject: score forKey: score.identifier];

}

self.earnedAchievementCache= tempCache;

[self submitAchievement: identifier percentComplete: percentComplete];

}

else

{

//在加载成就列表时出错。此时需要在下次提交成就的时候再次尝试

[self callDelegateOnMainThread: @selector(achievementSubmitted:error:) withArg: NULL error: error];

}

}];

}

else

{

//在成就列表中搜索当前使用的成就ID...

GKAchievement\* achievement= [self.earnedAchievementCache objectForKey: identifier];

if(achievement != NULL)

{

if((achievement.percentComplete >= 100.0) || (achievement.percentComplete >= percentComplete))

{

//玩家已实现该成就，任务完成

achievement= NULL;

}

achievement.percentComplete= percentComplete;

}

else

{

achievement= [[[GKAchievement alloc] initWithIdentifier: identifier] autorelease];

achievement.percentComplete= percentComplete;

//将成就添加到成就缓存中...

[self.earnedAchievementCache setObject: achievement forKey: achievement.identifier];

}

if(achievement!= NULL)

{

//提交当前成就...

[achievement reportAchievementWithCompletionHandler: ^(NSError \*error)

{

[self callDelegateOnMainThread: @selector(achievementSubmitted:error:) withArg: achievement error: error];

}];

}

}

}

以上代码注释对该方法的使用做了详细说明。接下来需要了解在项目中该如何使用该方法。

#### 判断玩家是否达到某种成就

在Xcode中打开GKTapper，切换到TapperController.m，找到updateCurrentScore方法，其中调用了checkAchievements方法，该方法用于判断玩家是否达到了某种成就。这两个方法的代码如代码清单15-7所示。

1. updateCurrentScore方法和checkAchievements方法代码

- (void) checkAchievements

{

NSString\* identifier= NULL;

double percentComplete= 0;

switch(self.currentScore)

{

case 1:

{

identifier= kAchievementGotOneTap;

percentComplete= 100.0;

break;

}

case 10:

{

identifier= kAchievementHidden20Taps;

percentComplete= 50.0;

break;

}

case 20:

{

identifier= kAchievementHidden20Taps;

percentComplete= 100.0;

break;

}

case 50:

{

identifier= kAchievementBigOneHundred;

percentComplete= 50.0;

break;

}

case 75:

{

identifier= kAchievementBigOneHundred;

percentComplete= 75.0;

break;

}

case 100:

{

identifier= kAchievementBigOneHundred;

percentComplete= 100.0;

break;

}

}

if(identifier!= NULL)

{

[self.gameCenterManager submitAchievement: identifier percentComplete: percentComplete];

}

}

- (void) updateCurrentScore

{

[self checkAchievements];

[self.tableView reloadData];

}

当玩家得分为1时，解锁第一个成就；当玩家得分为10时，第二个成就的完成度达到50%；当玩家得分为20时，解锁第二个成就；以此类推。只要成就ID不为空，就可以通过submitAchievement:方法将成就完成信息提交给iTunes Connect设置的指定成就。

#### 重置玩家成就

除了使用submitAchievement:方法提交成就，GameCenterManager类中还提供了resetAchievements方法，用于重置玩家的成就，该方法的实现如代码清单15-8所示。

1. resetAchievements方法代码

- (void) resetAchievements

{

self.earnedAchievementCache= NULL;

[GKAchievement resetAchievementsWithCompletionHandler: ^(NSError \*error)

{

[self callDelegateOnMainThread: @selector(achievementResetResult:) withArg: NULL error: error];

}];

}

在以上代码中，首先将已获得的成就缓存清空，然后在主线程中调用GKAchievement类的resetAchievementsWithCompletionHandler:方法。

然后看看GKTapper项目中是如何使用该方法。

切换到TapperController.m，发现其中定义了方法如代码清单15-9所示。

1. TapperController.m定义的方法代码

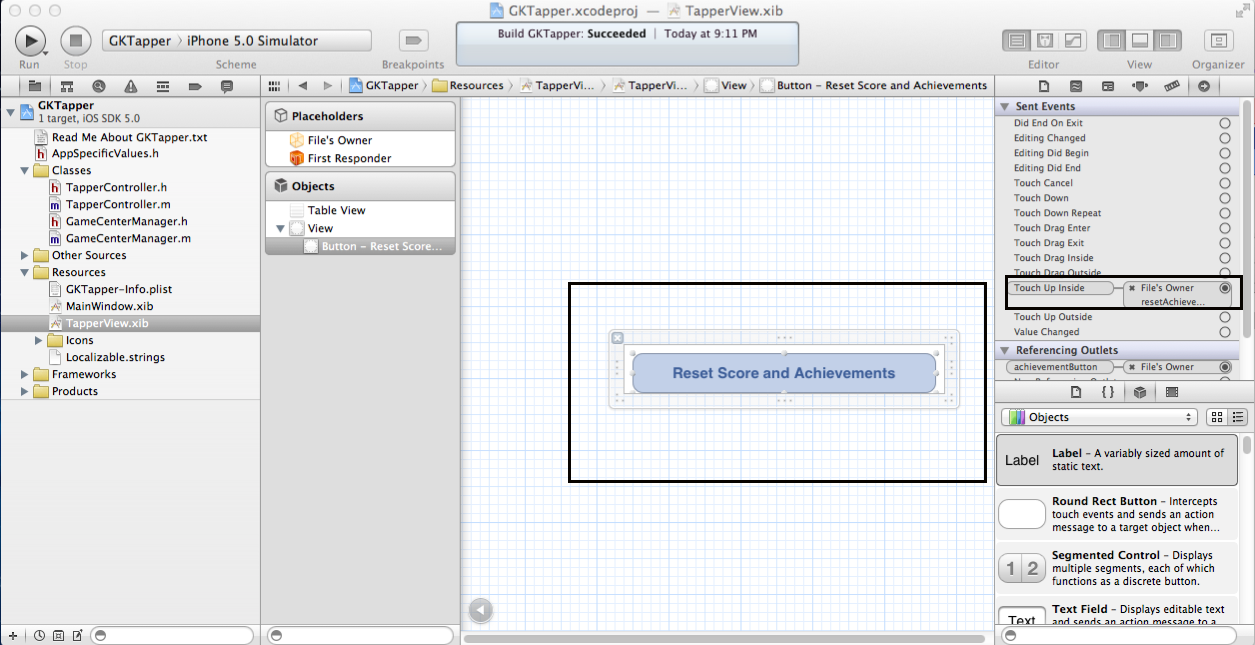
- (IBAction) resetAchievements: (id) sender

{

[gameCenterManager resetAchievements];

}

切换到TapperView.xib，选中“Reset Score and Achievements”按钮，在右侧面板选中“Show the Connection Inspector”，可以看到“Reset Score and Achievements”按钮及其对应的方法resetAchievements:，如图15-20所示。



1. “Reset Score and Achievements”按钮及其对应的方法

在项目运行时，当用户单击“Reset Score and Achievements”按钮时，就会调用resetAchievements:方法重新设置成就。

#### 显示成就

最后还需要了解在项目中如何显示成就。

切换到TapperController.h，可以看到TapperController继承自UITableViewController，并满足GKAchievementViewControllerDelegate协议，相关代码如代码清单15-10所示。

1. TapperController继承自UITableViewController

@interface TapperController : UITableViewController <UIActionSheetDelegate, GKLeaderboardViewControllerDelegate, GKAchievementViewControllerDelegate, GameCenterManagerDelegate>

切换到TapperController.m，找到两个方法如代码清单15-11所示。

1. 切换到TapperController.m找到的方法代码

- (void) showAchievements

{

GKAchievementViewController \*achievements = [[GKAchievementViewController alloc] init];

if (achievements != NULL)

{

achievements.achievementDelegate = self;

[self presentModalViewController: achievements animated: YES];

}

}

- (void)achievementViewControllerDidFinish:(GKAchievementViewController \*)viewController;

{

[self dismissModalViewControllerAnimated: YES];

[viewController release];

}

其中showAchievements方法中创建并初始化一个GKAchievementViewController实例对象，并在其中显示玩家的成就。而achievementViewControllerDidFinish则在显示完成后用平滑的动画关闭成就视图。

1. 实现积分排行榜

排行榜其实只是最佳分数的排序。当玩家达到某个积分时，可以创建一个GKScore对象，并调用方法将分数发送到Game Center。Game Center会记录每个玩家的最高分值。如果提交了较小的分值，或是重复发送某个分值，它会自动将其丢弃。

#### 发送分值

在GameCenterManager类中，使用reportScore:forCategory:方法将某个分值发送给Game Center，如代码清单15-12所示。

1. reportScore:forCategory:方法代码

- (void) reportScore: (int64\_t) score forCategory: (NSString\*) category

{

GKScore \*scoreReporter = [[[GKScore alloc] initWithCategory:category] autorelease];

scoreReporter.value = score;

[scoreReporter reportScoreWithCompletionHandler: ^(NSError \*error)

{

[self callDelegateOnMainThread: @selector(scoreReported:) withArg: NULL error: error];

}];

}

以上方法中，首先使用Game Kit API中的GKScore创建并初始化一个该类的实例对象，然后将其数值设置为传入的整数值，接着使用GKScore类的reportScoreWithCompletionHandler:方法将积分发送给Game Center。

接下来看看在GKTapper项目中如何使用该方法。

在Xcode中切换到TapperController.m，找到submitHighScore方法，如代码清单15-13所示。

1. submitHighScore方法代码

- (void) submitHighScore

{

if(self.currentScore > 0)

{

[self.gameCenterManager reportScore: self.currentScore forCategory: self.currentLeaderBoard];

}

}

而submitHighScore方法又在表视图的以下方法中出现如代码清单15-14所示。

1. submitHighScore方法在表视图中出现

- (void)tableView:(UITableView \*)tableView didSelectRowAtIndexPath:(NSIndexPath \*)indexPath

{

NSUInteger sectionNumber= [indexPath indexAtPosition: 0];

NSUInteger rowNumber= [indexPath indexAtPosition: 1];

switch (sectionNumber)

{

case kScoreHandlingSection:

{

if(rowNumber == 0)

{

[self submitHighScore];

}

else

{

if(rowNumber == 1)

{

[self addOne];

}

}

break;

}

case kLeaderBoardSection:

{

if(rowNumber == 0)

{

UIActionSheet\* selectLeaderboard= [[[UIActionSheet alloc] initWithTitle: @"Select Leaderboard" delegate:self cancelButtonTitle: @"Cancel" destructiveButtonTitle: NULL

otherButtonTitles: [NSString stringWithFormat: @"%@ Leaderboard", NSLocalizedString(kAwesomeLeaderboardID, NULL)],

[NSString stringWithFormat: @"%@ Leaderboard", NSLocalizedString(kHardLeaderboardID, NULL)],

[NSString stringWithFormat: @"%@ Leaderboard", NSLocalizedString(kEasyLeaderboardID, NULL)], NULL] autorelease];

[selectLeaderboard showInView: self.view];

}

[self.tableView deselectRowAtIndexPath: indexPath animated: NO];

break;

}

case kShowViewControllersSection:

{

if(rowNumber== 0)

{

[self showLeaderboard];

}

else

{

[self showAchievements];

}

break;

}

default:

[self.tableView deselectRowAtIndexPath: indexPath animated: NO];

break;

}

}

当sectionNumber为kScoreHandlingSection且rowNumber为0时，会调用submitHighScore方法，而submitHighScore方法又调用reportScore:forCategory:方法将得分发送到Game Center。

#### 加载积分排行榜

除了reportScore:forCategory:方法，GameCenterManager类中还提供了reloadHighScoresForCategory方法，其代码如代码清单15-15所示。

1. reloadHighScoresForCategory方法代码

- (void) reloadHighScoresForCategory: (NSString\*) category

{

GKLeaderboard\* leaderBoard= [[[GKLeaderboard alloc] init] autorelease];

leaderBoard.category= category;

leaderBoard.timeScope= GKLeaderboardTimeScopeAllTime;

leaderBoard.range= NSMakeRange(1, 1);

[leaderBoard loadScoresWithCompletionHandler: ^(NSArray \*scores, NSError \*error)

{

[self callDelegateOnMainThread: @selector(reloadScoresComplete:error:) withArg: leaderBoard error: error];

}];

}

在以上代码中，首先创建并初始化一个GKLeaderboard的实例对象，并设置其category、timeScope及range属性，然后调用GKLeaderboard的loadScoresWithCompletionHander方法加载积分排行榜。

在GKTapper项目中共有3处使用该方法，分别是最初对用户进行验证的processGameCenterAuth:方法，得分上传完成后的scoreReported:方法，以及表视图处理率的actionSheet:didDismissWithButtonIndex:方法。这里不再赘述。

最后，看看如何在项目中显示排行榜。

#### 显示排行榜

切换到TapperController.h，在@interface定义部分可看到，Tappercontroller继承自UITableViewController，并遵循GKLeaderboardViewControllerDelegate协议，相关代码如代码清单15-16所示。

1. @interface定义部分代码

@interface TapperController : UITableViewController <UIActionSheetDelegate, GKLeaderboardViewControllerDelegate, GKAchievementViewControllerDelegate, GameCenterManagerDelegate>

在TapperController.m中找到两个方法如代码清单15-17所示。

1. TapperController.m中的两个方法

- (void) showLeaderboard;

{

GKLeaderboardViewController \*leaderboardController = [[GKLeaderboardViewController alloc] init];

if (leaderboardController != NULL)

{

leaderboardController.category = self.currentLeaderBoard;

leaderboardController.timeScope = GKLeaderboardTimeScopeAllTime;

leaderboardController.leaderboardDelegate = self;

[self presentModalViewController: leaderboardController animated: YES];

}

}

- (void)leaderboardViewControllerDidFinish:(GKLeaderboardViewController \*)viewController

{

[self dismissModalViewControllerAnimated: YES];

[viewController release];

}

其中，showLeaderboard方法中创建并初始化了一个GKLeaderboardViewController实例对象，并在其中显示玩家的成就。而leaderboardViewControllerDidFinish则在显示完成后用平滑的动画关闭成就视图。

## 本章小结

本章介绍了苹果官方的社交游戏网络Game Center的功能和特点。详细讲解了如何在iTunes Connect中启用Game Center、设置成就系统和设置排行榜，以及如何使用GameCenterManager类实现对用户的验证、成就系统和排行榜。

第16章将介绍如何在游戏中添加IAP应用内购买。