Autodesk®
Scaleform®

手机游戏工具箱概述

本文介绍 Scaleform 4.2 手机游戏工具箱（Scaleform 4.2 Mobile Game Kit) 的架构、原代码和内容。此工具箱是一个跨平台的、基于触摸/手势的游戏演示，专用于采用 Scaleform 手机播放器的手机和平板电脑硬件。

作者： Nate Mitchell
版本： 1.01
上次编辑： 2012 年 6 月 5 日

Autodesk®
GAMEWARE

版权声明

Autodesk® Scaleform® 4.2

How to Contact Autodesk Scaleform:

Document	Mobile Game Kit Overview
Address	Autodesk Scaleform Corporation 6305 Ivy Lane, Suite 310 Greenbelt, MD 20770, USA
Website	www.scaleform.com
Email	info@scaleform.com
Direct	(301) 446-3200
Fax	(301) 446-3199

目录

1 引言

Scaleform 手机游戏工具箱提供触摸屏手机游戏的最佳做法实现示例，这种触摸屏手机游戏是将 Autodesk® Scaleform® 作为引擎和渲染器构建的。游戏《Starforce Battlement》是完全采用 ActionScript 3 编写的，并利用了 Scaleform 的“可装运的手机播放器”(Shippable Mobile Player) 实现跨平台部署。本概述详细介绍具体实现方法。尽管该游戏是专门针对移动设备设计的，但它也可以在支持鼠标或触屏输入的任何平台（包括 PC、Mac、Linux、iOS 和 Android）上播放。

此工具箱包括针对游戏和主菜单的完整 ActionScript 3 源代码，包括支持多分辨率的示例框架、声音播放、保存和加载到设备、成绩以及 iOS 游戏中心 (iOS Game Center) 集成。此工具箱包括游戏中使用的所有 Adobe® Flash® 内容和素材 (Asset) 。开发者可以重复利用此工具箱的所有资源，并且/或者作为其采用 Scaleform 的游戏的架构和实现的最佳示例。

尽管设计此游戏的目的是利用 Scaleform 在移动设备上播放，但此游戏也可以在采用 Adobe Flash Player 的 Web 浏览器上播放，这使其成为 Scaleform 在各种平台上的内存占用空间和性能的一个很好的基准。

下面从开发/运行时观点对构成 Starforce Battlement 的核心组件进行简单细分：

1. Flash Professional 用于互动艺术、动画、UI 布局和关卡编辑。
2. 游戏逻辑是用 ActionScript 3 编写的（FlashDevelop 4 是我们的推荐的编辑器）。
3. 通过 Scaleform 在设备上播放 Exported .SWF 文件。
4. Scaleform 的渲染器负责将播放结果绘制到设备 GPU 上，因而可以有效地显示在屏幕上。

1.1 Starforce Battlement

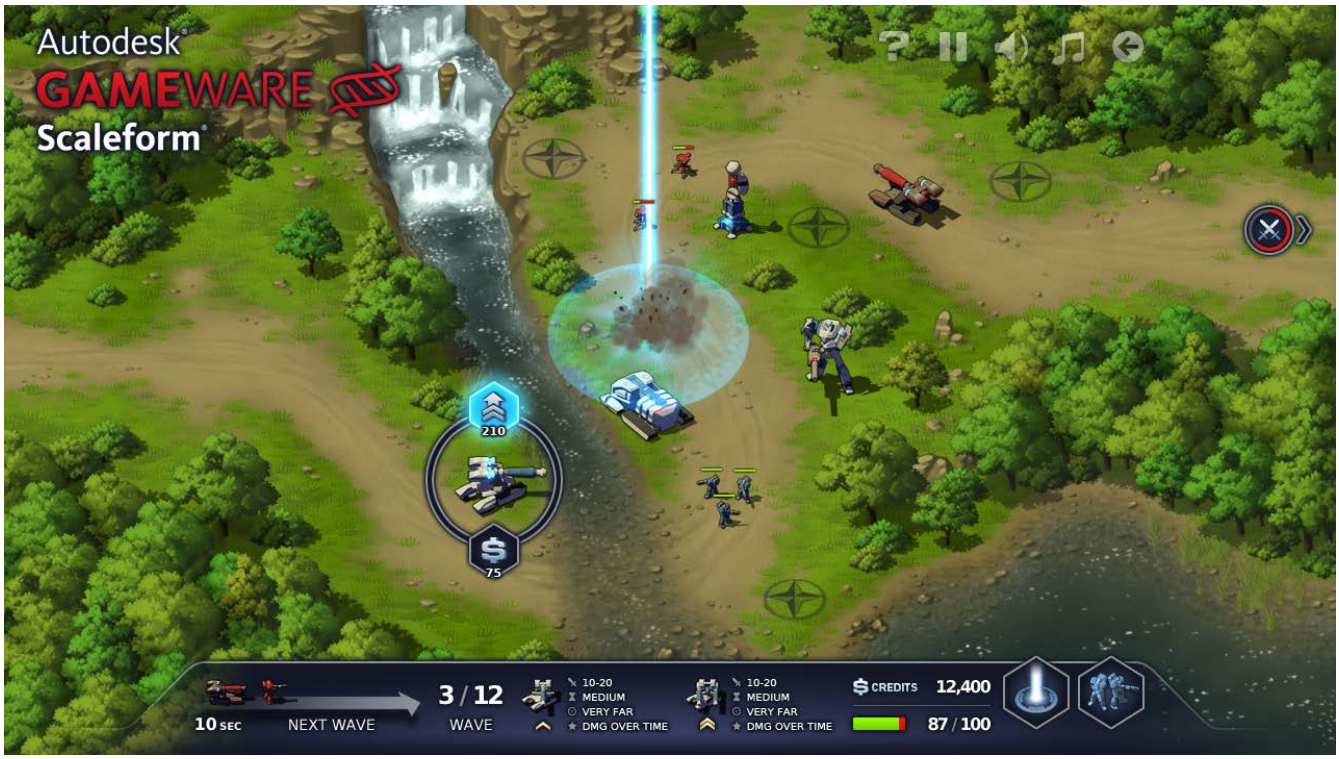

图 1：Starforce – 关卡 1 屏幕截图

Starforce Battlement 是一个包含角色扮演元素的城防游戏 (Tower Defense Game)。每个关卡的主要目的是防止一波波逼近的敌人到达其目的地。每当一个敌人到达目的地，就从玩家生命值中减去一分。如果足够敌人到达目的地，玩家生命值就减为零，因而游戏结束。

玩家可以在地图上指定位置构建军队/城堡 (Unit/Tower)，用来攻击逼近的敌人，阻止他们到达目的地。消灭敌人玩家可赢得积分 (Credit)，这些积分可花费在构建新的城堡上，也可以花在升级现有的城堡上。消灭所有各波敌人后，关卡就通过了。

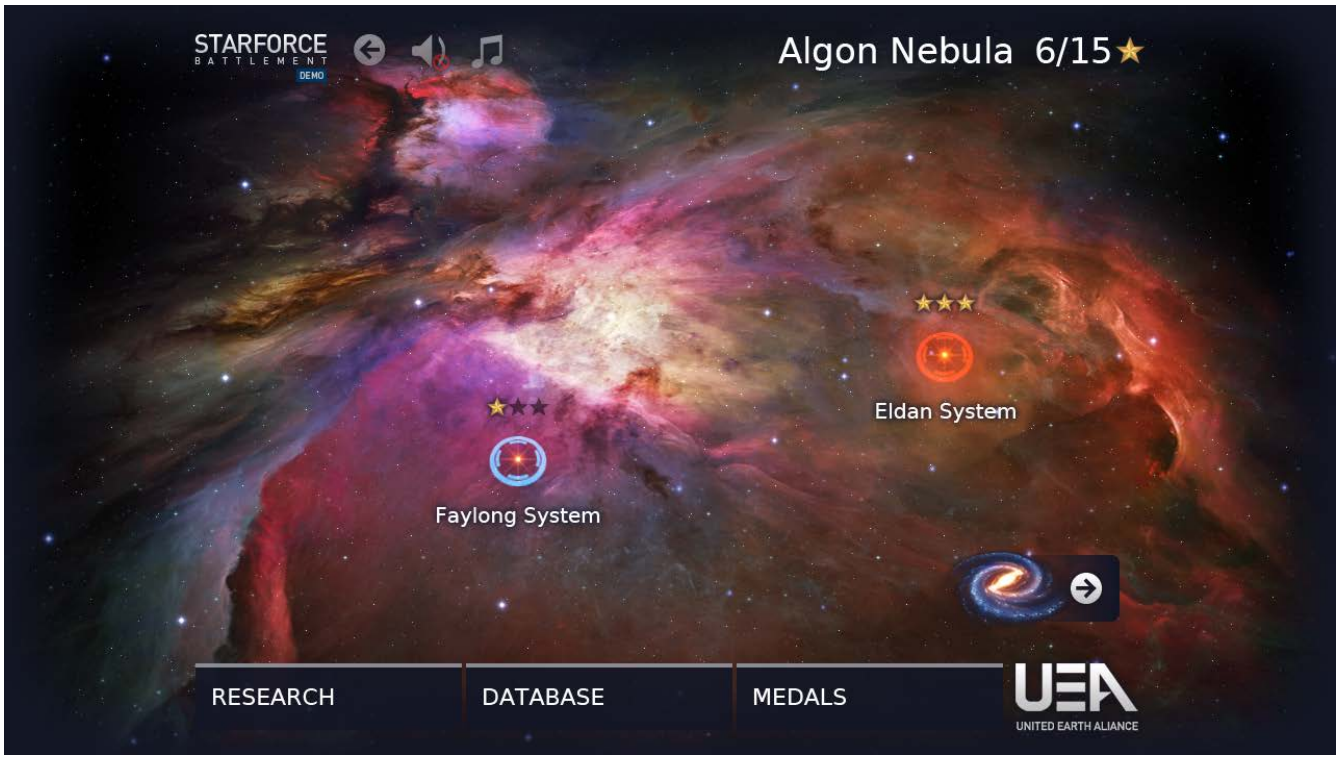

图 2：Starforce – 主菜单

作为一个完整的游戏演示版本，Starforce Battlement 还包括一个用于关卡选择、教程和查看玩家统计数据/成绩的主菜单，开发者可以在其自己的游戏中重复利用它们。

2012 年 5 月，可在 iOS App Store 上免费下载 Starforce Battlement。欢迎开发者将该应用程序下载到 iOS 设备上，以便了解 Scaleform 在 iOS 上的工作情况。

1.2 特性

Starforce Battlement 展示下列 Scaleform 移动特性：

1. 跨平台移动支持
2. GPU 矢量和位图渲染
3. 全面支持 ActionScript 3
4. 多点触摸和手势输入处理
5. 保存和加载数据
6. 声音和音乐播放
7. iOS 游戏中心集成
8. 定位闭锁 (Orientation Lock) / 定位处理 (Orientation Handling)

此工具箱还为下列特性提供 ActionScript 3 框架示例：

1. 玩家成绩
2. 声音和音乐播放
3. iOS 游戏中心集成
4. 保存和加载数据

2 概述

2.1 文件位置和构建说明

与此演示关联的文件位于下列位置：

- *Bin/Data/AS3/Kits/StarforceTD/* - 包含 ActionScript 3 源代码、Flash 内容以及玩游戏期间使用的其它资源（图标、声音等）。
- *Projects/Win32 /{Msvc80, Msvc90, or Msvc10}/GFx 4.0 SDK* / – 包含 Visual Studio 项目，用来在 Windows 上构建和调试 FxPlayer Visual Studio 2005/2008/2010。请注意，此应用程序没有自动配置为播放 Starforce.
- *Projects/iPhone/Xcode4/GFx 4.0 iPhone SDK* – 包含 Xcode 4 项目，用于 iOS 上的 “可装运手机播放起”(Shippable Mobile Player)。请注意，此应用程序没有自动配置为播放 Starforce.
- *LocalApps\StarforceTD\Android* – 一旦您首次通过 make 成功构建 Scaleform，此目录就会包含一个正确配置的 Eclipse 项目，用于 “Shippable Mobile Player”（已经配置为在 Android 上播放 StarforceTD。

Bin/Data/AS3/Kits/ StarforceTD 中包含有用于 Windows 的演示的一个预构建可执行程序 StarforceTD.exe。它也可以通过 Windows 开始菜单或 Scaleform SDK 浏览器进行访问。

用于 FxPlayer 的项目/解决方案文件或 FxMobilePlayer 应用程序用来在不同平台上编译、部署和运行手机游戏工具箱。

在 Windows 上，确保将 Debugging（调试）的 “Working directory”（工作目录）设置为 *Bin/Data/AS3/Kits/StarforceTD* 目录，而将 “Command line arguments”（命令行参数）设置为 *StarforceTD.swf*。

在 iOS 上，通过 Xcode 将应用程序构建并部署到设备后，使用 iTunes 把 SWF 文件和子目录（音频、图标）复制到应用程序。

2.2 演示用法

2.2.1 主菜单

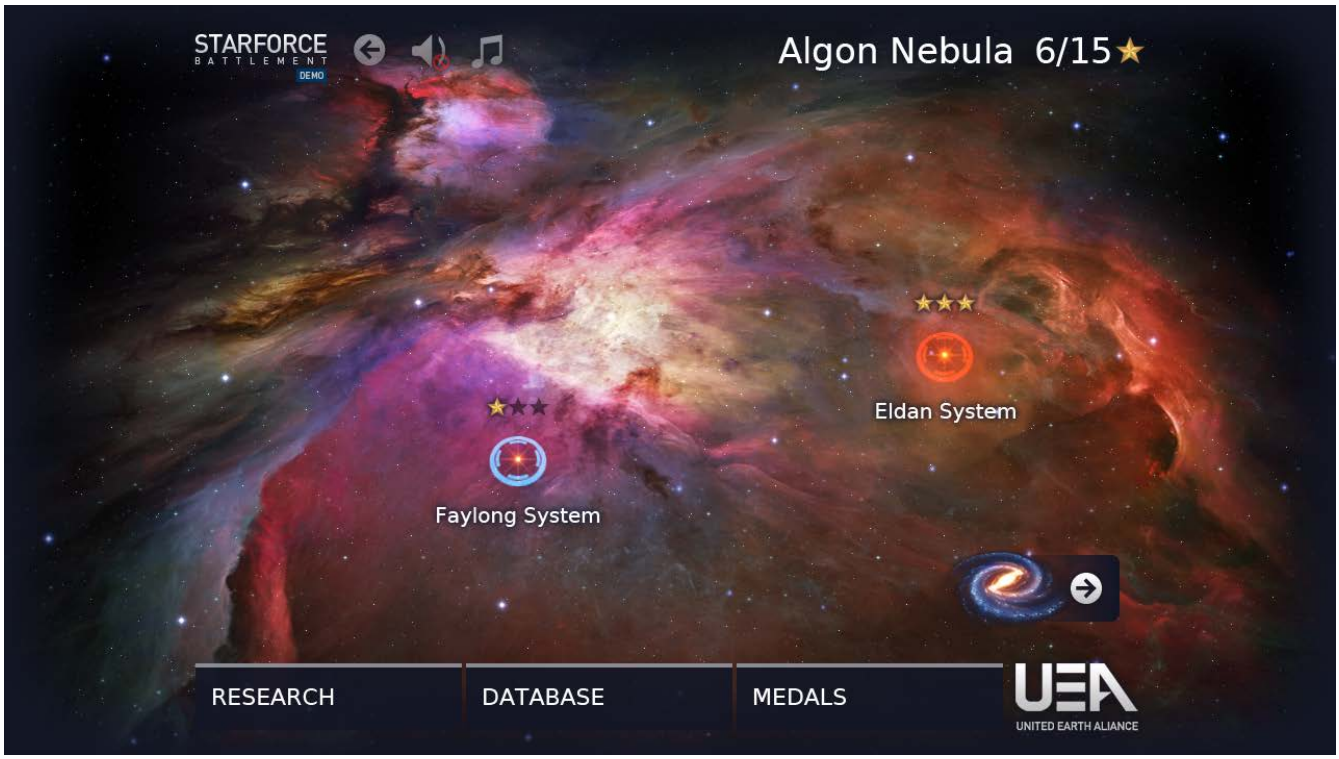

图 3：Starforce – 主菜单

StarforceTD.as 是游戏的进入点。此类是 StarforceTD.fla 的 Document Class（文档类），而且初始化时，会初始化 MainMainState。MainMainState 加载 StarforceMenu.swf。StarforceMenu.swf 包含“主菜单”美术素材，以及驱动主菜单的 ActionScript 3 类定义。

主菜单允许玩家选择任务，并提供额外选项，用来更改玩家的技能树，并查看教程、玩家统计数据历史记录以及玩家成绩。主菜单左上部的按钮允许玩家切换音乐、切换声音效果或退出游戏。

关卡选择包含指示器，这些指示器位于玩家可以选择的星系/星云 (galaxy/nebula) 图上。每个星云包含一组以线性级数解锁的任务。玩家可以在 2D 空间里随处拖动该图，以便在选择特定任务之前研究可用选项。

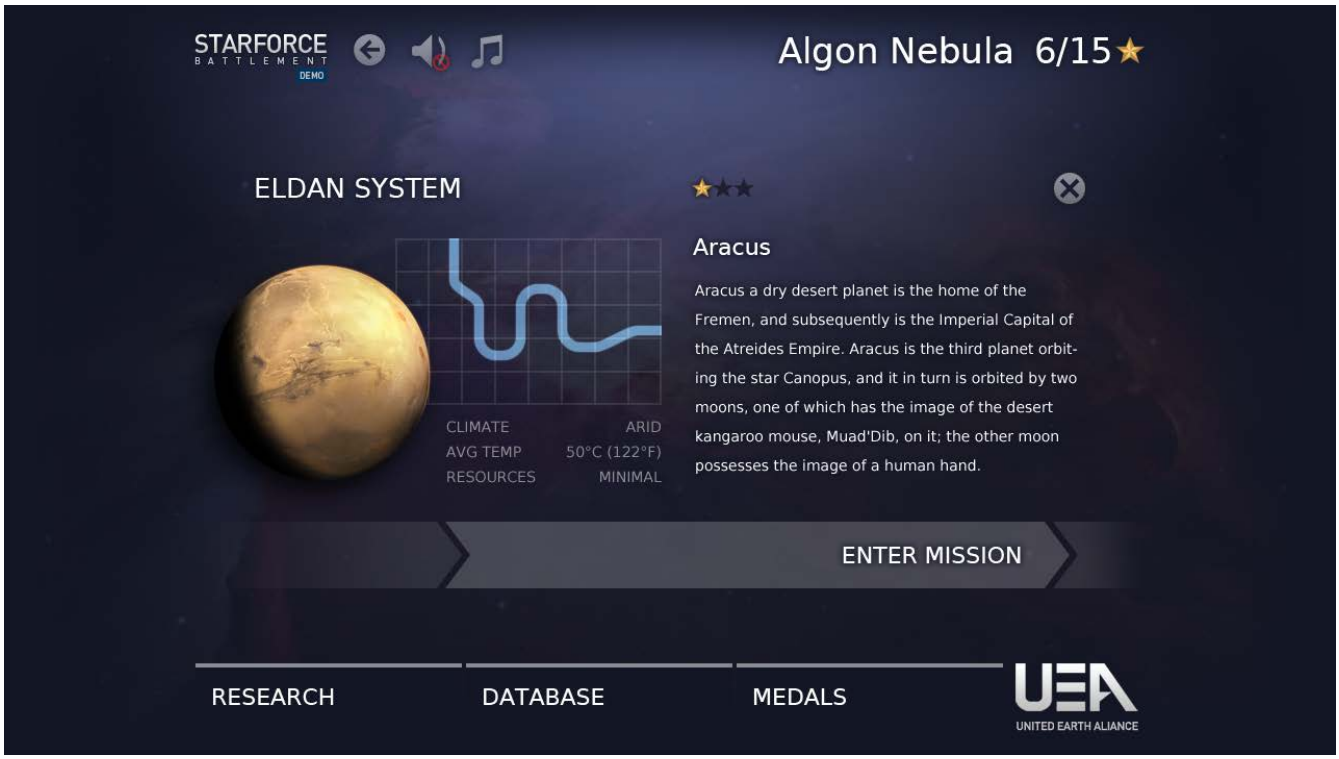

图 4：Starforce – 任务数据

当玩家选定一项可用任务时，就会显示“任务数据视图”(MissionDataView.as)。任务数据视图 (Mission Data View) 为玩家提供有关任务的详细信息。从此视图，玩家可以通过选择“进入任务”（Enter Mission) 按钮来启动任务，也可以通过在视图右上部选择 “X” 按钮来返回到“任务选择视图”(Mission Select View)。

2.2.2 游戏方式和 HUD

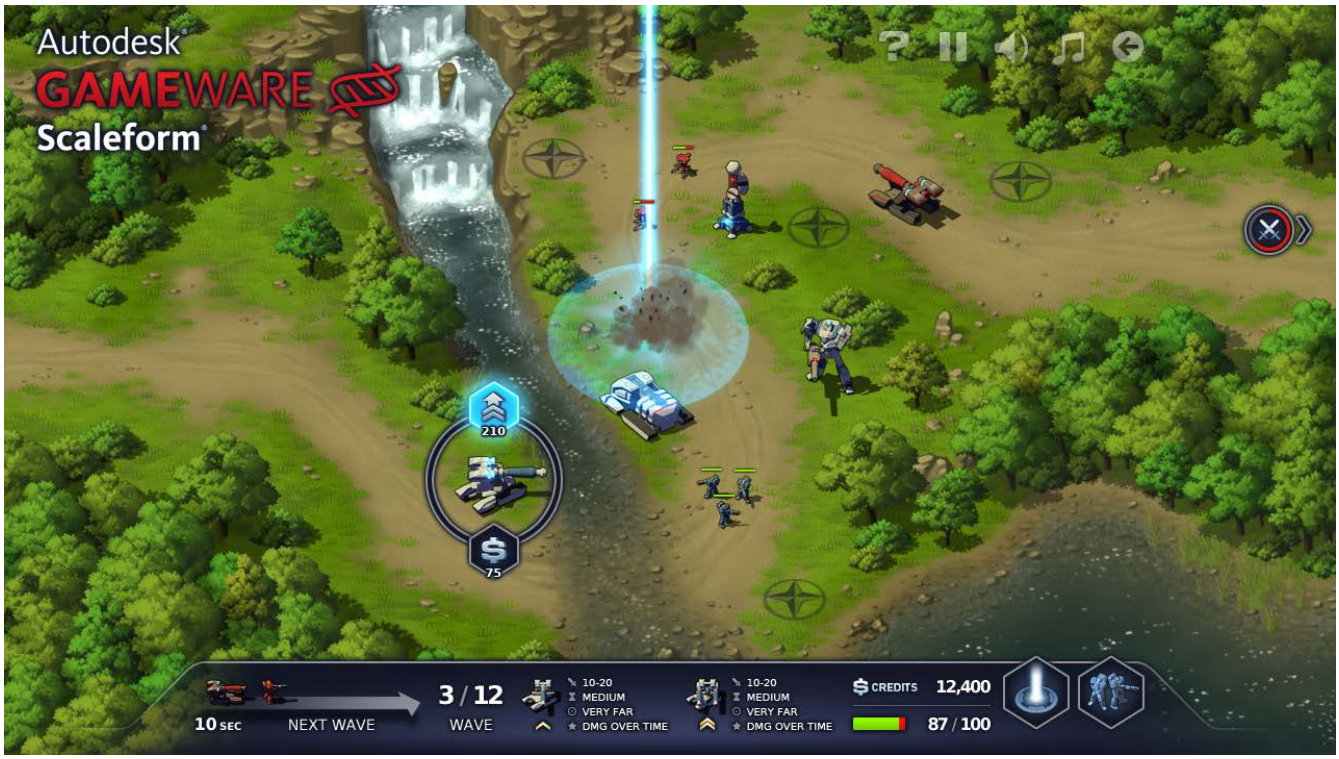

图 5 : Starforce - 一级游戏（有用户界面）

Starforce Battlement 的关卡和游戏方式封装在 StarforceTD.swf 中。 StarforceTD.swf 最初启动时，会将游戏关卡的所有数据加载到内存中，但在创建 LevelGameState 之后，才会初始化关卡。游戏细分为五个任务/关卡，每个都有一个唯一的设置。

只有在点击屏幕截图右手侧的脉动交叉剑按钮 -- “Rush Wave”（冲击波）-- 按钮后，才会启动第一波敌人。启动第一波后，敌人会在该关卡持续时间内以固定时间间隔到达，直到消灭所有敌人或者玩家被打败。HUD 左下角显示当前波数以及总波数，以及构成下一波敌人的预览图。

如果玩家成功消灭各波敌人，玩家将因其所过关卡而获得一个 1 星、 2 星或 3 星评级，具体星级取决于成功到达目的地的敌人的数量。这些星可用来发展玩家的技能树。

城堡构建位置（用地图上镶有钻石的黑圈表示）遍布整个关卡。选定后，允许玩家构建新城堡的位置上面就会出现一个上下文径向菜单。在一个城堡构建位置上一次只能构建一个城堡。只能在有效、可用的地点构建城堡，而且玩家必须拥有足够积分来负担城堡费用。玩家当前积分数显示在 HUD 的右边，就在玩家生命值 (Health) 上面。

一旦构建好一个城堡，最多可以升级城堡两次。选定一个城堡时，当前城堡等级以及下一等级的统计数据就会出现在 HUD 中心。

玩家有两项能力：“轨道炮打击”(Orbital Strike) 和“部署雇佣兵”(Deploy Mercenaries)，玩家可以通过点击 HUD 右下部的相应能力按钮随时使用这两项能力。激活一项能力后，玩家必须在地图上选择一个启动或定向该能力的位置。请注意，通过按住并拖动输入（鼠标或触摸屏），可以拖动已经启动的“轨道炮打击”。一旦使用一项能力，该能力就会“冷却下来”，并将在一限定时间段内处于禁用状态。通过能力按钮上方的一个深色调向用户指明冷却持续时间，该色调随着时间的推移而逐渐消失。

HUD 右上部的那组按钮允许玩家查看教程、暂停/恢复游戏、切换声音效果、切换音乐以及重启/退出当前任务。

3 架构

3.1 Flash 文件

用于游戏的多数美术素材均包含在 Flash .FLA 文件内，包括关卡、实体和 UI 以及游戏的多数动画。Starforce Battlement 主要使用“位图”(Bitmap) 美术，因为为了方便起见，单位和城堡的模型是用 3D 制作的。

包含游戏的美术（主要是 UI 中使用的图标）的一个子集不在这些 FLA 文件里面。运行时通过 ActionScript 加载这些外部图像。

Starforce Battlement 由三个不同的 FLA 文件组成：

1. **StarforceTD.fla** – 游戏的主要 .FLA。此文件包含 Starforce Battlement 的所有关卡、UI 和实体的 ActionScript 代码和美术素材。通过适当地加载和卸载“主菜单”(Main Menu) / “加载屏”(Loading Screen)，它还发挥游戏管理器的作用。
2. **StarforceMenu.fla** – 此文件包含“主菜单”和“任务选择”(Mission Select) 的 ActionScript 代码和美术素材。尽管它与 StarforceTD.swf 共享一些用户界面类定义，但它完全自包含，而且可以独立测试。
3. **LoadingView.swf** – 此文件包含“加载屏”（“游戏状态”转换期间显示）的 ActionScript 代码和美术素材。

3.2 ActionScript 代码

3.2.1 实现摘要

Starforce Battlement 游戏方式和 UI 的逻辑完全是用 ActionScript 3 编写的。这使游戏代码高度可移植，因为除 iOS Game Center 集成之外，再没有 Starforce 固有的针对设备的实现。

3.2.1.1 进入点

游戏的进入点是 StarforceTD.as 类，使用 Flash Document Class 属性，该类被捆绑到 StarforceTD.swf。有关“文档类”(Document Classes) 的更多信息，请参阅 Adobe 的 Flash 说明文档。

3.2.1.2 游戏状态

游戏分为三种 'GameState'（游戏状态），每种代表游戏可能处于的一个不同的状态。一次只能激活一种 GameState，而且由 StarforceTD 类所有，并且管理活动的 GameState。下面列出这三种游戏状态：

1. MenuGameState – 用于“主菜单”的游戏状态。
2. LoadingGameState – 用于“加载屏”的游戏状态。
3. LevelGameState – 用于关卡和游戏方式的游戏状态。

StarforceTD 的构造函数初始化子系统（数据、声音、多点触控），并加载 StarforceMenu.swf 和 MainMenuView.as 定义的主菜单。

3.2.1.3 主菜单实现

主菜单主要是用 com.scaleform.std.menu 程序包 (Package) 实现的（有关程序包的更多信息，请参阅下文）。主菜单建立在一个事件驱动型架构之上，在这个架构上，采用 ActionScript 3 的本机事件系统在各种主菜单视图之间传递事件。当用户作出选择时，就会发出并处理事件，以便打开和关闭窗口、读取数据并且/或者转换到全新的状态（例如，启动任务）。

启动一项任务时，从主菜单发出一个 GameEvent（游戏事件）。StarforceTD.as 类侦听此事件，并通过加载 LoadingGameState、卸载 MenuGameState 以及加载 LevelGameState，立即转换到该关卡。LevelGameState 加载选定的关卡，并且一旦完成加载该关卡，就立即开始游戏。

请注意，从技术的观点看，实际上不需要 LoadingGameState，因为将要绘制的关卡内容已经加载到 StarforceTD.swf 内的内存之中。不过，之所以在这里使用 LoadingGameState 转换，是为了向用户隐藏关卡初始化和 HUD，并且更加顺畅地过渡到游戏当中。

3.2.1.4 游戏实现

Level.as 是实现游戏的核心。此类充当游戏的所有关卡的基类。Level.as 类初始化和管理构成游戏的几乎所有元素，包括关卡、HUD、敌人、城堡和用户能力。Level 是所有关卡的基类。每个关卡有其自己的子类，这些子类定义敌人，并可能会覆盖其它行为和变量，以便使该关卡更加独特（例如，默认黄金数量、各波之间的时间、敌人类型、波数，等等）。

Level 负责创建、销毁和管理所有实体。一般情况下，游戏中的实体为城堡和敌人。所有实体都按关卡打勾，这使实体得以更新（即跨越地图、搜索敌人、攻击，等等）。

Flash Professional 便利于将一个 ActionScript 类绑定到“符号”(Symbol)，而符号由图形、动画和 ActionScript 构成。然后，在运行时，可以通过 ActionScript 对这些符号进行引用和实例化。这是 Starforce Battlement 在实例化游戏和菜单系统中的几乎所有图形内容时所使用的方法。有关将“符号”绑定到 ActionScript 和在运行时创建实例的更多信息，请参阅 Adobe 的 Flash 说明文档。

例如，TowerMech 符号可在 StarforceTD.fla 的 Flash Library 中找到。在 TowerMech 符号的属性中，请注意，该符号绑定到 com.scaleform.std.entities.TowerMech ActionScript 类。无论何时通过 ActionScript 实例化此类，都会创建此图形符号的一个实例，然后通过代码操控此实例。

Level.as 类使用一个事件驱动的框架管理游戏。该类侦听从实体、用户界面或任一其它系统框架（声音、数据、成绩）发出的事件，并相应地更新游戏和 HUD。例如，当一个城堡 (Tower) 攻击一个部队 (Unit) 时，该城堡不是直接修改其攻击的部队的生命值，而是发出 AttackEvent。全局地侦听 AttackEvents 的“关卡”(Level) 收到该 AttackEvent，处理攻击者和防御者，计算防御者应该受到的伤害的数量，并将该伤害情况应用到防御者。

游戏实体之间的大部分互操作性是作为从实体彼此内部提取各种类并将这些实体范围之外的数据包含在计算之内的一种手段来通过关卡的（例如，玩家的技能树，它可能会影响某些情况下防御者受到的伤害的数量）。此设计模式也便利于进行许多全游戏范围内的更改，因为它消除了冗余代码，并使实体类彼此独立。

3.2.1.5 游戏数据

com.scaleform.std.data 程序包中定义了游戏的多数数据，包括所有城堡、任务、能力和敌人的统计数据。在整个实现过程中，使用 GameData 静态类访问这些数据。这样，假如目前有外部变量影响这些数据（例如，玩家的技能树可能会更改对某个城堡的标准攻击力），在把数据返回到请求类之前，GameData 类就可以对这些数据进行修改。

理想的情况是，不把游戏的默认数据存储在 ActionScript 内，而是使用某种像 XML 或 JSON 这样的外部数据定义格式，然后在运行时加载。令人遗憾的是，Scaleform 4.0 不支持 XML，因此，最便捷的解决办法是将数据存储在类定义内。这一问题可能会在未来的版本中得到修正，以便反映最佳做法。

3.2.1.6 输入处理

对游戏的输入是使用 ActionScript 事件侦听器编写脚本的。这使输入逻辑可以跨平台工作，因为 Scaleform 播放器将会跨越平台提取鼠标/触摸输入，并将其正确地传递到 ActionScript。

游戏根据本机 Scaleform 应用程序当前是否支持触摸输入来侦听鼠标或触摸输入。假设 FxPlayer 设置正确，应用程序将会根据 MovieView 上是否安装了 MultitouchInputState，以及与触摸相关的适当操作系统功能是否可用，在运行时识别是否支持触摸功能。

尽管根据应用程序设置只积极使用一组侦听器，但 Starforce 中的所有互动元素均同时具有 TouchEvent 和 MouseEvent 的逻辑。TouchEvent 和 MouseEvent 的类型是不同的，虽然许多 TouchEvent 类型都具有非常相似的 MouseEvent 类型，反之亦然。例如，TouchEvent.TOUCH_TAP 可像 MouseEvent.MOUSE_CLICK 一样使用，而 TouchEvent.PRESS 也可像 MouseEvent.MOUSE_DOWN 一样使用。

要将单个函数声明同时作为 TouchEvent 和 MouseEvent 的侦听器，该功能的参数必须属于类型 Event（事件）。假如需要从该事件获取更多信息，例如，*target*（目标）或输入事件的坐标，则必须将该事件归到适当类型中。运行时识别类型的一个标准方法是 *"is"* 操作符，如下所示：

```
protected function onSetRallyClick( e:Event ):void {
    var point:Point;
    if (e is MouseEvent) {
        var me:MouseEvent = e as MouseEvent;
        point = new Point( me.stageX, me.stageY );
    }
    else {
        var te:TouchEvent = e as TouchEvent;
        point = new Point( te.stageX, te.stageY );
    }
}
```

3.2.2 ActionScript 程序包

根据所包含的类的功能和/或用途，游戏的 ActionScript 代码分成若干 ActionScript '程序包'（Package)。

所有下列程序包均前面带有 "com.scaleform.std"：

1. **abilities（能力）** – 玩家能力（轨道炮打击、雇用雇用兵）的类别定义和游戏逻辑。
2. **controls（控件）** – 整个过程中重复使用的 UI 元素和组件。
3. **core（核心）** – 状态流 (State Flow) 和数据存储/检索的基本组件的类定义。
4. **data（数据）** – 游戏的数据定义，包括玩家、实体、关卡和能力。
5. **entities（实体）** – 游戏的数据定义，包括玩家、实体、关卡和能力。
6. **events（事件）**– 所有游戏、UI 和状态事件的事件定义。
7. **fx** – 游戏中特殊效果（爆炸、死亡动画）的类定义。
8. **hud**– 游戏内 HUD 和游戏的类定义。

9. **levels（关卡）** – 每个关卡的类定义。请注意，Level（关卡）类是游戏的基本类之一。
10. **loading（加载）** – 加载屏 (Loading Screen) 的类定义。
11. **menu（菜单）** – “主菜单”的类定义，包括所有 UI 元素、任务选择和输入处理。
12. **system（系统）** – 执行声音播放、保存和加载以及定位处理等任务时用来从 ActionScript 与系统进行互动的框架。
13. **utils（实用工具）** – 整个代码库（但不局限于特定程序包）中使用的实用工具类、函数和定义。

3.2.3 框架

3.2.3.1 保存/加载框架

保存和加载框架（在 com.scaleform.std.system.SaveSystem.as 中定义）是在磁盘上读写数据的实现示例。尽管用来存储播放器进度的代码非常简单，但它可作为处理更复杂实现的基础。

此功能的 ActionScript 接口采用了 SharedObject 类。SharedObject 是一个标准 ActionScript 3 类，它使开发者可以在磁盘上存储数据，并且可以从 ActionScript 对象检索数据。Scaleform 对 SharedObject 类的实现支持图元类型 (Primitive Type) 和数组 (Array)。

每个 SharedObject 均由一个唯一字符串标识符定义，该标识符用来在运行时读写数据。在下面的代码中，SharedObject 的唯一字符串标识符为 “sb_player_data”。

```
// Local reference to the galaxy progress. Populated from the PlayerData.
public static var galaxyProgress:Array = null;
// Constant String for the property of the SharedObject we'll modify.
protected static const GALAXY_PROGRESS_SO_PROP:String = "galaxyProgress";
// Reference to the SharedObject once we retrieve it.
protected static var _playerDataSO:SharedObject = null;

public static function load( pd:PlayerData ):void {
    _playerDataSO = SharedObject.getLocal("sb_player_data");
    galaxyProgress = _playerDataSO.data[ GALAXY_PROGRESS_SO_PROP ];
    pd.setGalaxyProgress( galaxyProgress );
}

public static function save( pd:PlayerData ):void {
    // Write PlayerData into SharedObject.
    _playerDataSO = SharedObject.getLocal("sb_player_data");
    galaxyProgress = pd.getGalaxyProgress();
    _playerDataSO.data[ GALAXY_PROGRESS_SO_PROP ] = galaxyProgress;
    _playerDataSO.flush();
}
```

请注意，开发者并不局限于单个 SharedObject 实例。只要每个 SharedObject 都使用一个唯一名称，磁盘上就可以存储多个 SharedObject。这是一种把数据分成若干类别的便捷方式。

有关 SharedObject 类的更多详细信息，请访问：
http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/flash/net/SharedObject.html

3.2.3.2 声音框架

声音框架（com.scaleform.std.system.SoundSystem 中定义）是处理您的游戏内的声音效果和音乐播放的一个实现示例。该框架根据从游戏发出的 SoundEvent 加载和播放外部声音（在此案例中为 .mp3）。SoundEvent 可从带有对 stage 的引用的任何类发出。

```
dispatchEvent( new SoundEvent( SoundEvent.PLAY_SOUND, true, true, "infantry_attack.mp3",
                               1, false ) );
```

声音框架使用两个声音声道，一个用于后台，一个用于前台。SoundEvent 指定目标声音在哪个声道上播放。游戏的所有声音效果均在前台声道上播放，而后台音乐则在后台声道播放。这样就可以使用 SoundEvent 单独设定每个声道为静音或在二者之间进行切换。

```
dispatchEvent( new SoundEvent( SoundEvent.MUTE_SOUNDFX, true, true );
```