

Autodesk® Scaleform®

AMP 用户指南

本文介绍 Scaleform 内存和性能分析器 (AMP) 以及如何使用该系统来分析应用程序。

作者: Alex Mantzaris
版本: 3.03
上次编辑时间: 2013 年 4 月 10 日

Copyright Notice

Autodesk® Scaleform® 4.2

© 2012 Autodesk, Inc. All rights reserved. Except as otherwise permitted by Autodesk, Inc., this publication, or parts thereof, may not be reproduced in any form, by any method, for any purpose.

Certain materials included in this publication are reprinted with the permission of the copyright holder.

The following are registered trademarks or trademarks of Autodesk, Inc., and/or its subsidiaries and/or affiliates in the USA and other countries: 123D, 3ds Max, Algor, Alias, AliasStudio, ATC, AUGI, AutoCAD, AutoCAD Learning Assistance, AutoCAD LT, AutoCAD Simulator, AutoCAD SQL Extension, AutoCAD SQL Interface, Autodesk, Autodesk Homestyler, Autodesk Intent, Autodesk Inventor, Autodesk MapGuide, Autodesk Streamline, AutoLISP, AutoSketch, AutoSnap, AutoTrack, Backburner, Backdraft, Beast, Beast (design/logo) Built with ObjectARX (design/logo), Burn, Buzzsaw, CAiCE, CFdesign, Civil 3D, Cleaner, Cleaner Central, ClearScale, Colour Warper, Combustion, Communication Specification, Constructware, Content Explorer, Creative Bridge, Dancing Baby (image), DesignCenter, Design Doctor, Designer's Toolkit, DesignKids, DesignProf, DesignServer, DesignStudio, Design Web Format, Discreet, DWF, DWG, DWG (design/logo), DWG Extreme, DWG TrueConvert, DWG TrueView, DWFx, DXF, Ecotect, Evolver, Exposure, Extending the Design Team, Face Robot, FBX, Fempro, Fire, Flame, Flare, Flint, FMDesktop, Freewheel, GDX Driver, Green Building Studio, Heads-up Design, Heidi, Homestyler, HumanIK, i-drop, ImageModeler, iMOUT, Incinerator, Inferno, Instructables, Instructables (stylized robot design/logo), Inventor, Inventor LT, Kynapse, Kynogon, LandXplorer, Lustre, MatchMover, Maya, Mechanical Desktop, MIMI, Moldflow, Moldflow Plastics Advisers, Moldflow Plastics Insight, Moondust, MotionBuilder, Movimento, MPA, MPA (design/logo), MPI (design/logo), MPX, MPX (design/logo), Mudbox, Multi-Master Editing, Navisworks, ObjectARX, ObjectDBX, Opticore, Pipeplus, Pixlr, Pixlr-o-matic, PolarSnap, Powered with Autodesk Technology, Productstream, ProMaterials, RasterDWG, RealDWG, Real-time Roto, Recognize, Render Queue, Retimer, Reveal, Revit, RiverCAD, Robot, Scaleform, Scaleform GFx, Showcase, Show Me, ShowMotion, SketchBook, Smoke, Softimage, Sparks, SteeringWheels, Stitcher, Stone, StormNET, Tinkerbox, ToolClip, Topobase, Toxik, TrustedDWG, T-Splines, U-Vis, ViewCube, Visual, Visual LISP, Vtour, WaterNetworks, Wire, Wiretap, WiretapCentral, XSI.

All other brand names, product names or trademarks belong to their respective holders.

Disclaimer

THIS PUBLICATION AND THE INFORMATION CONTAINED HEREIN IS MADE AVAILABLE BY AUTODESK, INC. "AS IS." AUTODESK, INC. DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE REGARDING THESE MATERIALS.

如何联系 Autodesk Scaleform:

文档	AMP 用户指南
地址	Autodesk Scaleform Corporation 6305 Ivy Lane, Suite 310 Greenbelt, MD 20770, USA
网站	www.scaleform.com
电子邮箱	info@scaleform.com
电话	(301) 446-3200
传真	(301) 446-3199

目录

1	引言	1
1.1	AMP 服务器	1
1.2	AMP 客户端	2
2	AMP 入门	10
2.1	进行连接	10
2.2	分析性能指标	11
2.3	分析渲染指标	12
2.4	分析内存消耗	13
2.5	使用透支分析	19
2.6	使用批量分析	20
3	平台和集成说明	21
4	构建注释	22
5	添加 AMP 支持	23
5.1	Debug 文件生成	23
5.2	应用程序控制	23
5.3	连接状态	24
5.4	标记	24
6	与 AMP 相关的常见问题解答	25
7	更多信息	26

1 引言

AMP™ 是 Scaleform® 公司的远程分析系统，能够监控 CPU 使用状况、图形渲染以及 Scaleform 应用程序内存消耗情况。实时的图表提供一种快速查明内存和性能瓶颈的方法，而每帧统计数据功能允许深入分析和确定根源。AMP 利用其每帧 ActionScript 函数和源代码计时，还有助于查找脚本问题所在。

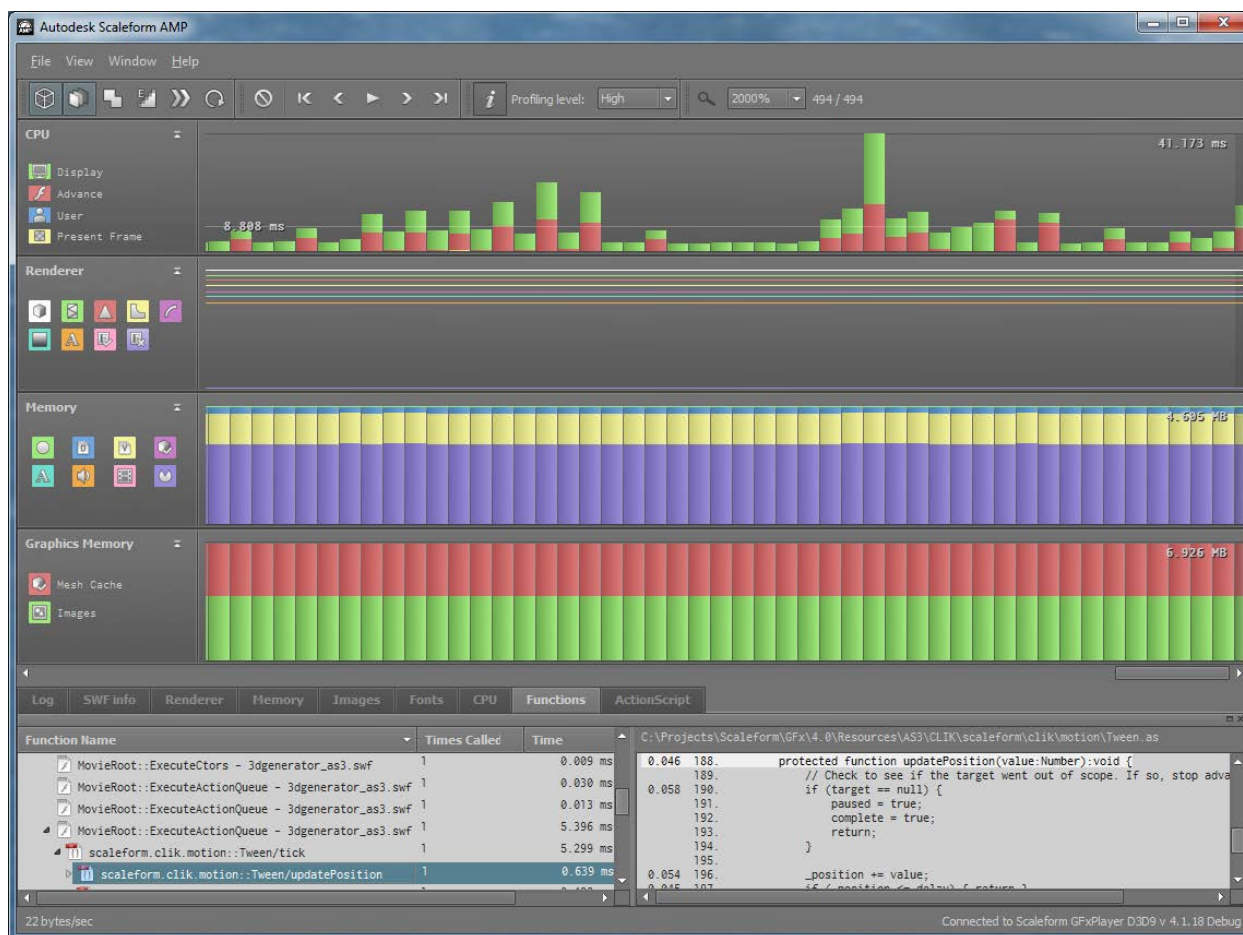


图 1: Scaleform AMP

每个 Scaleform 应用程序都可以充当一个 AMP 服务器，这就意味着，它可侦听分析器进行的入站连接请求，收集每个帧的信息，并通过网络将其发送给分析器。分析器可充当一个 AMP 客户端，这意味着它可连接到特定的 AMP 服务器，处理来自服务器的每个帧的信息，并发送请求以控制所分析的应用程序。

AMP 可用于 Scaleform 3.2 和更高版本。

1.1 AMP 服务器

被分析的 **Scaleform** 应用程序 (**AMP** 服务器) 通过网络将一定数量的信息发送到分析器。

可以根据函数分析级别对每个 **ActionScript** 函数和一些重要的 **C++ Scaleform** 函数进行计时。如果启用 **ActionScript 2** 中的每行源代码分析, 就会对每次 **ActionScript** 字节码执行进行计时 (**ActionScript 3** 源代码分析不需要计时每一个字节码)。此外, 还记录渲染统计数据、详细内存堆和图像信息、**Flash** 文件信息以及 **ActionScript** 标记。请注意, 每个帧的这一额外工作都会减缓应用程序运行速度并增加其内存消耗 --- 在启用每行计时的情况下尤其如此。

为了不让 **AMP** 干扰所分析的应用程序, 将所有 **AMP** 服务器信息都保存在一个单独的内存堆上。如果此内存堆的大小超出预设极限, 就会暂停应用程序, 直到通过网络发出待发帧信息, 并随后将其删除。

Scaleform Shipping 构造 (**Builds**) 不充当 **AMP** 服务器, 而且可以根据需要针对一个应用程序内的任何构造, 容易地禁用 **AMP** 支持。

服务器上的性能统计数据是按 **Movie** 维护的。这样一来, 倘若有多个视图与特定 **SWF** 文件关联, 用户就可以容易地查明问题点。它还使 **AMP** 可以为 **ActionScript** 在单个线程上执行多个 **Movies** 产生有意义的调用图表。

除向客户端报告内存和统计数据外, **AMP** 客户端可以通过多种方式控制服务器以便于进行分析。例如, 分析器可以发送如下请求: 线框模式渲染请求、暂停、快进或重启 **Flash** 电影请求、切换防混叠 (**Anti-Aliasing**) 和笔画模式请求、更改本地化字体请求以及更改矢量图形曲线公差。此外, 客户端可以切换特殊 **AMP** 渲染模式, 以指出显示性能瓶颈, 例如, 遮罩、过滤器和像素无效渲染突出。

尽管在 **Scaleform Player** 中实现了上述所有功能, 但其中某些功能还需要针对应用程序加以实现, 而且可以作为 **Scaleform** 集成的组成部分进行执行。为了“优雅地”处理 **Scaleform** 应用程序中没有实现其中某些功能的情形, **AMP** 服务器向连接中的客户端报告那组支持的应用程序控制功能。它还向客户端报告其当前状态 (比如线框模式), 进行视觉反馈。

最后, **AMP** 服务器能够通过网络将 **Flash** 调试信息 (**ActionScript 2 only**) 发送到分析器, 这样一来, 假如客户端无法在本地访问此信息, 也可以显示源代码和每行计时。为了实现这种传输, 该 **SWD** 文件 (**ActionScript2**) 或 **AS** 文件 (**ActionScript3**) 应处在 **AMP** 可找到它的位置。

1.2 AMP 客户端

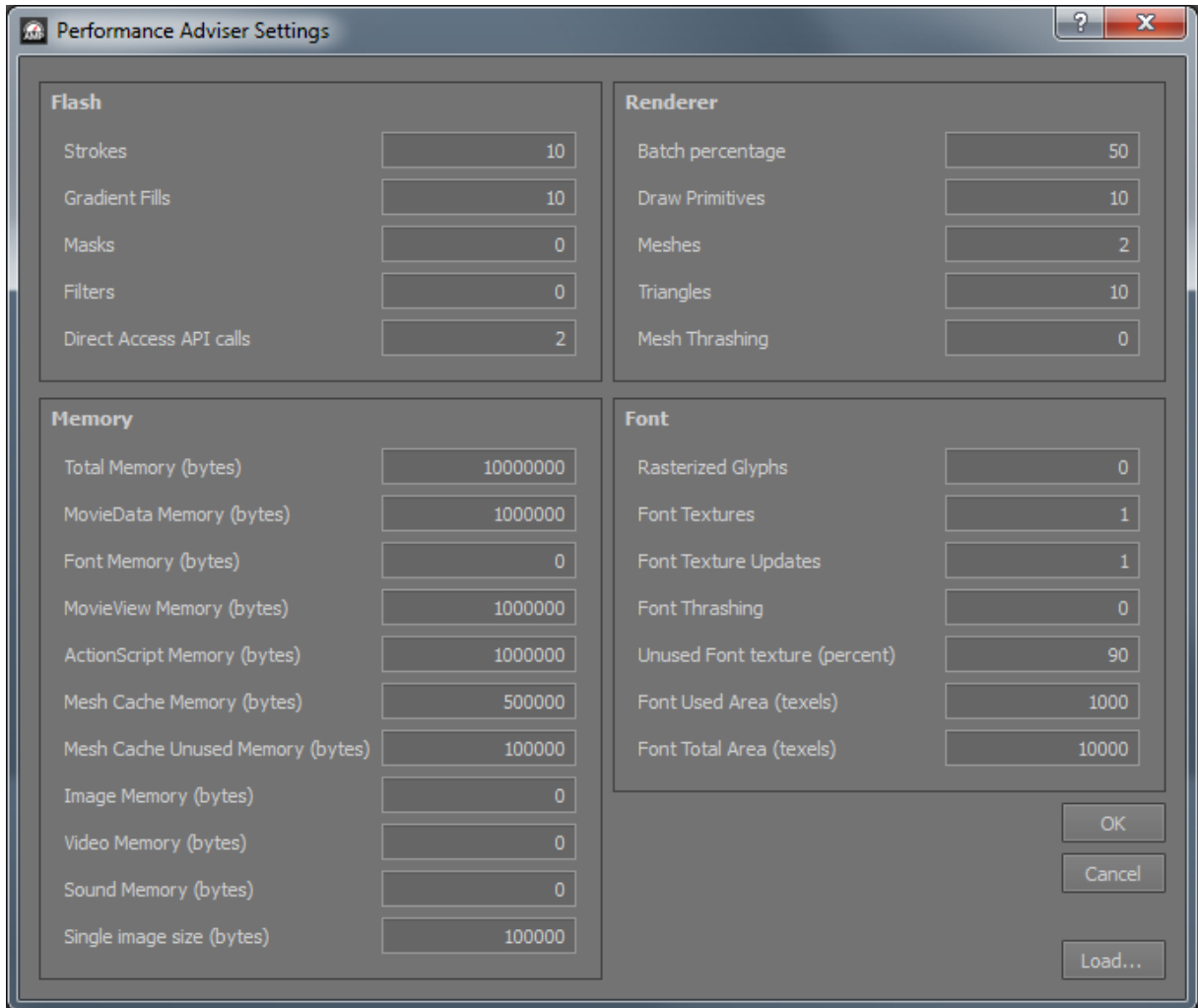
远程分析器 (**AMP** 客户端) 是一个连接某个 **AMP** 服务器的独立应用程序, 接收每个帧的概要 (**Profile**) 信息, 并以一种对用户友好的方式显示此信息, 以便于有效地分析性能和内存瓶颈。分析器本身就是一个 **Scaleform** 应用程序, 它是利用 **Scaleform CLIK** 实现的。

AMP 客户端拥有多种特性, 使开发人员可以查看其 **Flash** 资源的多个方面并查明具体问题所在:

- **CPU 图表：**显示在 **ActionScript (Advance)**、**Scaleform 渲染 (Display)**、**Direct Access API (User)** 和图形缓冲区交换 (**PresentFrame**) 中每个帧花费多少时间。此图表对于由于 **Scaleform** 引起的 CPU 使用峰值尤其有用。当选定一个带有峰值的帧时，就可以通过检查该帧的详细统计数据来查明原因。用户可以通过在此图表上左击并拖动鼠标直到全部选中所需的帧来选择多个帧。右击并拖动此图表可滚动图表窗口。通过在图例框中单击相应的切换按钮，可以显示或隐藏此图表的每个组件。
- **渲染图表：**[Scaleform 最佳做法指南](#) 文档标识多个与渲染相关的区域，这些区域可能导致性能影响，例如，绘制图元、三角形和线条数量太多，或者存在掩码、过滤器、笔触或渐变填充。渲染图表中显示上述每个区域的计数器，因此，开发者可以更加深入地了解 **Scaleform** 内部所发生的情况。
- **内存图表：**内存图表跟踪 **Scaleform** 的内存使用情况，这使开发者可以按类别（例如，渲染、图像、声音、字体、电影数据等）监测系统内存使用情况，并包括所有加载的文件。
- **图形内存图表：****Scaleform** 图形内存使用情况大部分也是通过 **AMP** 跟踪的，而 **AMP** 报告针对图像和网格缓存使用的 **GPU** 内存。
- **“日志”选项卡：**点击“日志”(Log) 选项卡，可显示被分析的应用程序在连接过程中产生的 **Scaleform** 日志。
- **Performance Adviser:** This tab provides guidance on how to improve the performance of Flash content. It is a list of warnings that fire when a user-defined threshold of some content-related property has been exceeded. The performance-related properties are:
 - *Text Fields optimized for readability:* If you use "Anti-alias for readability" for large text fields then the font cache contains separate glyph entries for each font size of each character. If a text field uses "Anti-alias for animation" then there will be only one entry, regardless of font size.
 - *Ratio of draw primitives to number of meshes:* This is a measure of batching efficiency. In the worst case of no batching, there will be one mesh for each draw primitive (ratio of 1.0). The ratio drops when some meshes are part of the same draw primitive, which is an indication of improved batching efficiency.
 - *Number of draw primitives:* A high number of draw primitives is an indication of a complex scene that takes longer to render.
 - *Number of meshes:* A high number of meshes is also an indication of a complex scene.
 - *Number of triangles:* A high number of triangles hurts both memory and performance.
 - *Mesh cache evictions:* This is the number of times per frame that an item is removed from the cache to make room for a new entry. Ideally, this number is zero, which means that everything needed by the renderer is found in the cache.
 - *Number of strokes:* As indicated in the “Best Practices” guide, strokes in flash content result in a performance hit.
 - *Number of gradient fills:* As indicated in the “Best Practices” guide, gradient fills also result in a performance hit.
 - *Number of masks:* As indicated in the “Best Practices” guide, the use of masks results in a performance hit.

- *Number of filters*: As indicated in the “Best Practices” guide, the use of filters results in a performance hit.
- *Number of DirectAccess API calls*: Too many DirectAccess API calls can result in a performance hit.
- *Number of rasterized glyphs*: Too many glyphs can hurt memory consumption.
- *Number of font textures*: Some platforms may have more than one font texture.
- *Number of font texture allocations*: The number of times per frame a new font texture is created. This can hurt performance.
- *Font cache evictions*: This is the number of times per frame a glyph is removed from the cache to make room for a new glyph. Ideally, this number is zero, which means that the font cache is large enough to hold all rasterizations.
- *Font misses*: This is number of times per frame glyphs are rasterized because they are not found in the font cache. This number can be high when text has not been encountered before, and this is normal. Sometimes, there may not be enough space left in the glyph cache to store additional glyphs when new text is encountered. When this happens, older glyphs have to be evicted from the cache to make space for new ones. This means that when the old text is encountered again, it will have to be re-rasterized and rendered to the cache, incurring a performance penalty. This type of font cache eviction is bad and should be optimized whenever possible.
- *Unused font texture percentage*: A low percentage indicates that there is space reserved for the font cache that is not being used.
- *Used font cache area*: Too many different font and styles can result in excessive memory consumption.
- *Total font cache area*: The size of all the font textures. A font cache improves performance at a memory consumption cost.
- *Total Memory*: The total Scaleform memory footprint.
- *MovieData Memory*: This is directly related to the size of the imported movie files.
- *Font Memory*: Font memory can be reduced by having text fields share fonts.
- *MovieView Memory*: The memory used for GfX::Movie, containing its timeline and instance data. MovieView memory is subject to internal garbage collection.
- *ActionScript Memory*: Too much ActionScript content can result in excessive memory consumption.
- *Mesh Cache Memory*: The mesh cache improves performance, but hurts memory consumption.
- *Unused Mesh Cache Memory*: Unused mesh cache hurts memory consumption without improving performance.
- *Image Memory*: Both system and graphics card memory used for images.
- *Video Memory*: This is memory used for Scaleform video, not GPU memory.
- *Sound Memory*: Memory used for sound.
- *Number of images exceeding a certain size*: The user can set the image threshold, and AMP reports the number of images that exceed that threshold.





To set values for all of the above thresholds, AMP reads an XML file, called “adviser.xml” in its working directory. If it finds no such file, all thresholds are set to zero. The user can modify any threshold by clicking the Settings button on the right of the Performance Advisor tab. This brings up the settings dialog:










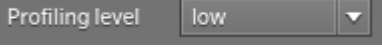
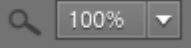


Clicking OK saves the settings to an XML file that AMP reads next time it is executed. The user may create different adviser settings for different content types, and select the one to be used by loading it from the settings dialog. Launching AMP with a command-line argument of “-adviser <AdviserFilePath>” makes the program read an adviser file other than the default one located in the working directory.

- **“SWF 信息”(SWF info) 选项卡：**此选项卡提供有关正在分析的所有 flash 文件的一些静态信息。这些信息包括文件名、flash 版本、电影尺寸以及电影帧速率。
- **“渲染器”(Renderer) 选项卡：**此选项卡显示与该图表相同的信息，但是采用数字形式。它还包括一些其它渲染统计数据，例如：网格和字体缓存抖动（在帧期间逐出的项目的数量）、字体填写率（所有字形的面积占字体缓存纹理总面积的百分比）、字体缓存失败率（帧期间有多少字形未能被分配，且被矢量形状所取代）、帧期间字体缓存纹理更新次数以及当前字体纹理数量。

- **“内存”(Memory) 选项卡:**此选项卡使用户可以详细确定 Scaleform 足迹、Scaleform 要求的内存有多少实际在用，以及如何按用途分配内存。当以完整内存报告模式（该模式通过 AMP 工具栏上的 “I” 按钮进行切换）分析时，内存报告变得更加详细，显示按堆、按文件和按用途分的明细数据。
- **“图像”(Images) 选项卡:** 显示每个图像及其尺寸和格式所占用的内存。点击列表中的每个项目可在预览窗格中显示该图像的一个缩略图。请注意，图像数据不是作为分析信息的组成部分发送，而是根据要求进行发送。因此，可能的情况是，假如 Scaleform 已经卸载该图像，就不会显示任何图像。AMP 目前并非支持预览所有图像格式，例如，DXT 压缩图像。
- **“字体”(Fonts) 选项卡:** 显示字体缓存占用的全部内存以及每个 SWF 的字体列表。当选中该选项卡中的每个项目时，就会显示代表当前字体缓存的一个图像，具体方式与“图像”选项卡中的图像预览方式相似。
- **CPU 选项卡:** 将“推进”(Advance)、“显示”(Display) 和“用户”(User) 时间细分为子类别，以便于更加详细地考察针对所选帧的时间花费在了什么地方。假如选中多个帧，则值就会是那些帧之间的平均值。
- **“函数”(Functions) 选项卡:** 显示一个树状控件，其中包含每个函数内花费的时间数量，这些函数也包含从该函数调用的函数。当选中多个帧时，调用时间和数量就是每个游戏帧的平均值。ActionScript 和 C++ 函数混合在同一个调用图表中，因为代码路径可以在 ActionScript VM 及该应用程序的其余部分之间移动。扩展每个函数就会显示从该函数调用的函数。
- **ActionScript 选项卡:** 此选项卡显示与“函数”(Functions) 选项卡相似的信息，但信息不是分层的，不包括 C++ 函数，而且为每个函数花费的时间不包括花费在列表中其它函数中的时间。此选项卡便于快速识别最昂贵的脚本函数。
- **“源”(Source) 窗格:** 此窗口是“函数”选项卡和 ActionScript 选项卡的组成部分。对于 AS3，始终为 AS 文件中驻留的代码启用按行计时。对于嵌入 SWF 中的 AS3 代码，不显示源代码。。对于 AS2，要查看每行计时情况（默认情况下，不显示），需要从工具栏上的下拉控件中选择“高”(high) 级函数分析。AS2 中的指令分析非常耗时，而且会减缓所分析的 Scaleform 应用程序的运行速度，最好在已经确定问题所在后再进行指令分析。倘若不止一个其他函数调用某个函数，紧接每行显示的计时数据不会加到该函数调用图表中报告的时间数量内。这是因为每行计时数据是花费在该函数中的总时间，而不管该函数是从哪里调用的。此外，行计时数据不包括函数调用内所花费的时间，而调用图表计时数据则包括被调用的函数中所花费的时间。要显示源代码和每行计时数据，AMP 服务器（SWF 位置）上找不到的任何 Flash 调试文件都需要位于 AMP 客户端可执行文件的工作目录之中，或者，如果是 AS3 文件，则位于已加载的 SWF 的相同相对路径中。
- **文件 (File) 菜单:** 此菜单包括下列选项：
 - **连接 (Connection):** 引出连接对话框，便于连接到一个不同的服务器。
 - **断开连接(Disconnect):** 终止当前连接。
 - **调试信息路径 (Debug Info Paths):** 调出一个对话框，列出文件系统中的所有路径，其中，AMP 搜索 SWD 文件（针对 AS2）或 AS 文件（针对 AS3）。可以分别使用“添加”(Add) 和“删除”(Remove) 按钮来添加或删除路径。

- **加载概要帧 (Load Profile Frames):** 加载要重新检查的以前保存的概要运行。选择此菜单项可引出一个文件加载对话框，该对话框用来定位概要数据文件。
 - **保存概要帧 (Save Profile Frames):** 将当前概要运行保存到磁盘。选择此菜单项可引出一个文件保存对话框，该对话框用来选择包含概要数据的文件的路径和名称。
 - **转储内存报告 (Dump Memory Report):** 将当前配置文件帧保存到 AMP 的工作目录中的一个名为“AmpMemReport.txt”的文件中。
 - **退出 (Exit):** 关闭应用程序。
- **“视图”(View) 菜单:** 此菜单包括下列选择：
 - **图表工具提示 (Graph Tooltips):** 选择此菜单，当把鼠标放在 AMP 中的图表上时，可切换上下文敏感的弹出式帮助。
 - **时间单位 (Time Units):** 允许用户选择用于报告所有时间数量的单位，例如，“显示”时间和“推进”时间。选择可以是毫秒，也可以是微秒。
 - **内存单位 (Memory Units):** 允许用户选择“内存”选项卡和“内存”图表的单位。选择可以是字节，可以是千字节，也可以是兆字节。
 - **“窗口”(Window) 菜单:** 此菜单允许用户显示和隐藏上面所述的任何选项卡（日志、SWF 信息、渲染器、内存、图像、字体、CPU、函数、ActionScript）。它还以一项 Restore UI（恢复 UI）选择为特色，该选项可以取消用户以前对 AMP 应用程序所做的任何窗口自定义。
 - **“帮助”(Help) 菜单:** 此菜单目前有两个选项：“用户指南”(User Guide) -- 链接到 Scaleform.com 开发者中心中的此文档，以及“关于”(About) -- 调出一个包含 AMP 版本和积分的屏幕。
- **应用程序控制工具栏:** 此 UI 元素包含一系列按钮，它们用来控制所分析的应用程序。
 -  **线框 (Wireframe):** 向被分析的应用程序发出一个请求以便于以线框模式渲染内容。当应用程序已处于线框模式时，单击此按钮可使其退出此模式。
 -  **透支模式:** 向被分析的应用程序发出一个请求，以进入一种渲染模式，该模式突出像素渲染无效、遮罩和过滤器区域。AMP 服务器用绿色渲染像素渲染无效区域（颜色强度与渲染无效次数成比例），以红色渲染遮罩，以蓝色渲染过滤器。当应用程序已经处于透支模式时，单击此按钮可使其退出此模式。
 -  **批量分析模式:** 向所分析的应用程序发送一个请求以进入某种渲染模式，该渲染模式以不同颜色绘制每个渲染批次。当应用程序已经处于批量分析模式时，单击该按钮可使其退出此模式。
 -  **防混叠 (Anti-aliasing):** 向被分析应用程序发出一个请求，以循环通过防混叠模式。三个可用模式为 Scaleform 的边缘防混叠技术 (EdgeAA)、硬件全景防混叠 (HW FSAA) 或不防混叠 (None)。

-  **快进 (Fast-forward):** 向被分析的应用程序发出一个请求, 以便以快进方式播放 Flash 内容。这使 SWF 就像渲染帧一样快速前进, 而不管其文档 FPS 设置如何。当该应用程序已经处于快进模式时, 单击此按钮可使其退出此模式。
-  **重启 (Restart):** 向被分析的应用程序发出一个重启 Flash 内容的请求。
- **“帧控制”(Frame control) 工具栏:** 此工具栏包含下列按钮:
 -  **清除 (Clear):** 放弃当前概要运行。丢弃分析帧, 由此为新数据释放 AMP 客户端内存。
 -  **第一个帧 (First Frame):** 选定当前概要运行中的第一个帧, 由此更新底部面板选项卡上显示的信息。
 -  **前一个帧 (Previous Frame):** 从当前选定的一个帧选择前一个帧, 由此更新底部面板选项卡上显示的信息。
 -  **暂停 (Pause):** 向被分析的应用程序发出一个停止发送内存和性能数据的请求。此操作不会暂停应用程序本身。
 -  **下一个帧 (Next Frame):** 从当前选定的一个帧选择下一个帧, 由此更新底部面板选项卡上显示的信息。
 -  **下一个帧 (Last Frame):** 选择并锁定到当前概要运行中的下一个帧上。接收到新帧的信息时, 选择更改为始终是最后一个收到的帧, 并连续更新底部面板选项卡上显示的信息。
- **“分析”(Profiling) 工具栏:** 此元素包含用于进行分析的一系列按钮。
 -  **详细分析 (Detailed Profiling)** (详细的内存明细): 选中此选项时, AMP 收集每个帧的更加详细的内存统计数据, 这些统计数据显示在“内存”选项卡中。此选项还使 AMP 显示“图像”选项卡中的图像的完整列表, 而不是最多十个图像。
 -  **分析级别:** 此下拉控件允许有三个函数分析级别: “低”(Low)、 “中”(Medium) 和 “高”(High)。低级分析使 AMP 只对某些高级 C++ 函数计时。中级分析导致 ActionScript 函数计时以及上一级函数。高级分析也收集来自某些较低级 C++ 函数的数据。此外, 高级分析跟踪每行 ActionScript 2 代码中花费的时间, 该代码显示在选定的帧的“代码”(Code) 面板中。要显示源行计时情况, 必须有一个调试文件 (SWD)。如果是 ActionScript 3, 就始终显示这些计时情况。
 -  **“缩放”(Zoom) 工具栏:** 包含一个下拉式控件, 用来更改图表的水平缩放比例。最右边的下拉控件变更图表的水平标度。放大 (Zooming out) 功能允许查明该运行的问题领域, 而缩小 (Zooming In) 功能则允许更加精确地检查这些领域。也可以使用鼠标滑

轮进行缩小和放大，并提供通过下拉方式所达不到的较高分辨率。此工具栏还显示当前选定的帧数。

- **“状态”(Status) 栏：**分析器底部有一个“状态”栏，它显示当前连接以及通过网络接收信息的比特率。

AMP 允许用户自定义其外观。可以通过将工具栏和选项卡拖动到其新位置来固定它们的位置。可以从应用程序取消固定这些选项卡和工具栏。图表和选项卡的尺寸可以调节，而且图表可以部分或全部折叠。这些 UI 更改的状态在 AMP 分析会话过程中都是不变的，但用户可以从“窗口”菜单中的“重置 UI”(Reset UI) 选项中将几何图形重置回到其默认状态。

2 AMP 入门

要开始使用 AMP 分析器，请运行 Scaleform 主安装目录下 Bin/AMP 中的 *GFxAmpClient.exe* 可执行文件。应用程序启动，并出现连接对话框。

2.1 进行连接

客户端启动时会自动出现一个连接对话框。它提示用户要么从一个发现的 AMP 服务器列表中进行选择，要么手动键入要连接的 IP 地址和端口。如果相关 AMP 服务器尚未运行，就选第二种方法，就像需要分析启动中的应用程序时的情况一样。

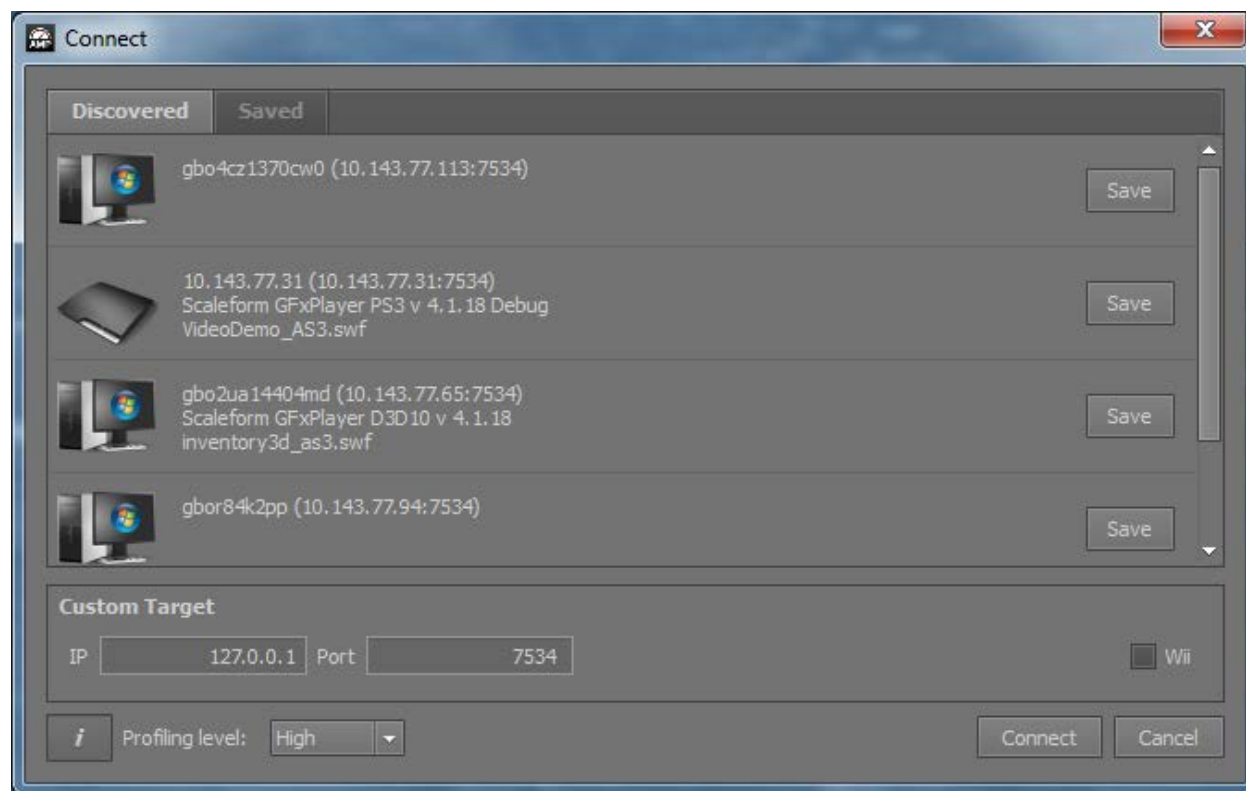


图 1：AMP 连接对话框

“文件” (File) 菜单下的“连接” (Connection) 项随时可用来引出连接对话框。发现的列表中的每个 AMP 服务器都有一个醒目的图标，用以识别其平台（PC、Xbox360、PS3 等）。紧接该图标的是应用程序的标题、服务器正在侦听的 IP 地址和端口以及目前播放的 Flash 内容。如果其中某些信息丢失，这是因为相应的 AMP 服务器不在发送这些信息。本文档后面部分将会提供一个如何向

任何 **Scaleform** 应用程序添加完全 **AMP** 支持，包括广播应用程序和内容信息。通过双击一个发现的服务器或选择一个服务器并单击“连接” (**Connect**) 按钮来建立连接。

从连接对话框，可以选择连接上的分析等级以及详细内存报告，这样就可以保证立即在连接上设置这些项目。当正在分析某个 **Scaleform** 应用程序的启动情况时，这尤其重要。

请注意，与 **Wii** 开发工具箱 (**NDEV**) 的连接不使用上面的服务器发现方法。相反，使用内置到 **NDEV** 中的 **USB** 接口 (**COM** 端口) 将 **Wii** 从物理上连接到 **PC**。通过在 **IP** 地址字段中指定“**wii**”来从连接对话框连接到 **Wii AMP** 服务器。

2.2 分析性能指标

一旦建立连接，分析信息就开始流入客户端，而图表就开始获得填充。最上面的图表显示每个帧在渲染 **Scaleform (Display)** 和执行 **ActionScript (Advance)** 以及在 **Direct Access API (User)** 中所花费的 **CPU** 时间数量。

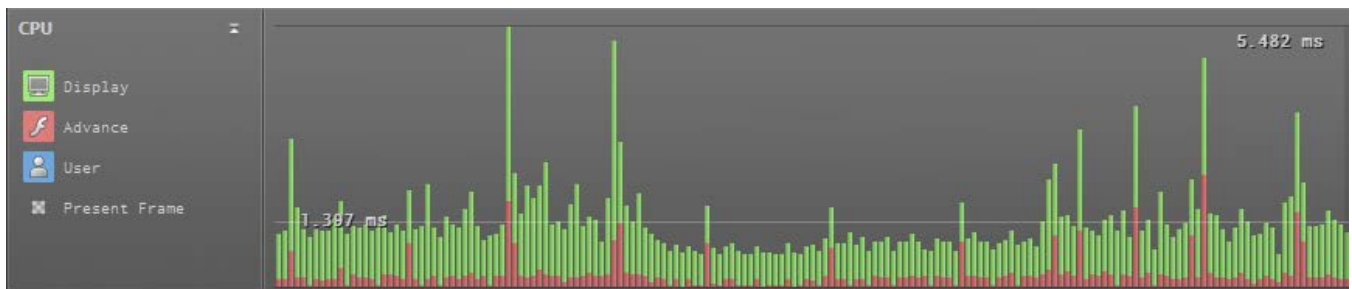


图 2: CPU 图表

如果在分析期间 **CPU** 图表在某个点没有任何意义，则可以将该图表最小化，或者重新调整大小，以便为其它 **UI** 元素留出更多空间。如果对某个图解数量不感兴趣，可以通过点击其图例旁边的图标来将其删除。

与 **CPU** 图表相关的是分析器底部的“**CPU 摘要**” (**CPU Summary**) 选项卡、“**函数**” (**Functions**) 选项卡以及 **ActionScript** 选项卡。“**CPU 摘要**”选项卡显示更详细的图表信息，“**函数**”选项卡显示选定帧的调用图表，而 **ActionScript** 选项卡显示选定帧期间调用的所有脚本函数的非分层视图。

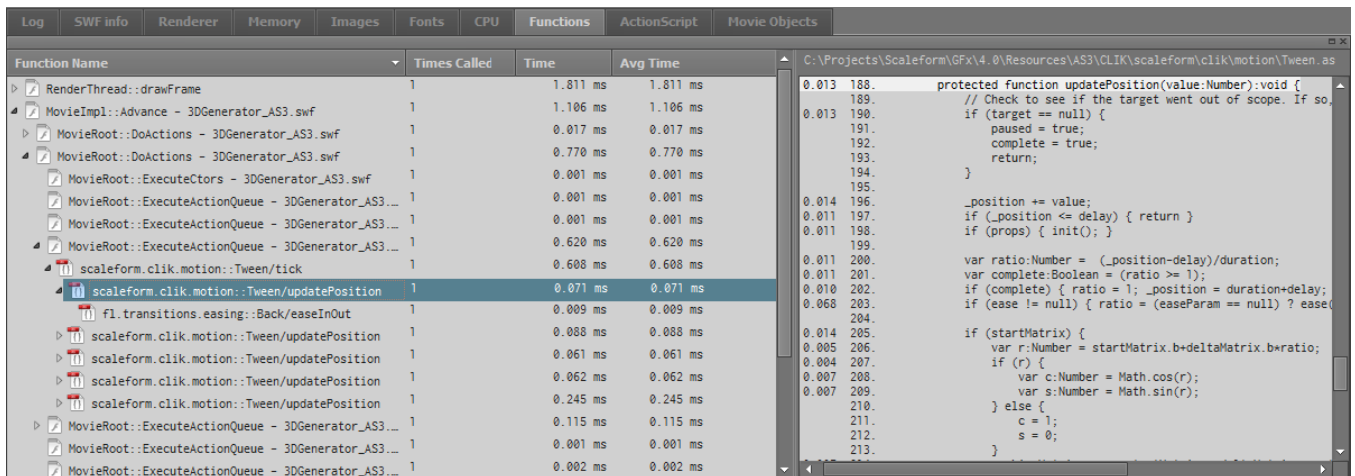


图 3: CPU 摘要

通过检查 CPU 详细信息选项卡在 CPU 图表中找到某个有峰值的帧，就可以确定脚本性能瓶颈的原因。另一方面，通过检查渲染图表，可以更容易地查明渲染性能瓶颈。

2.3 分析渲染指标

渲染图表显示绘画单元数目、三角形数目、线条数目、使用过的遮罩数目、笔画数目、梯度填充数目、网格缓存中的字形数目以及针对选定帧进行的渲染期间发生的更新次数。[最佳做法指南\(Best Practices Guide\)](#) 详细说明最大限度地减少上述数量对于性能的重要意义。

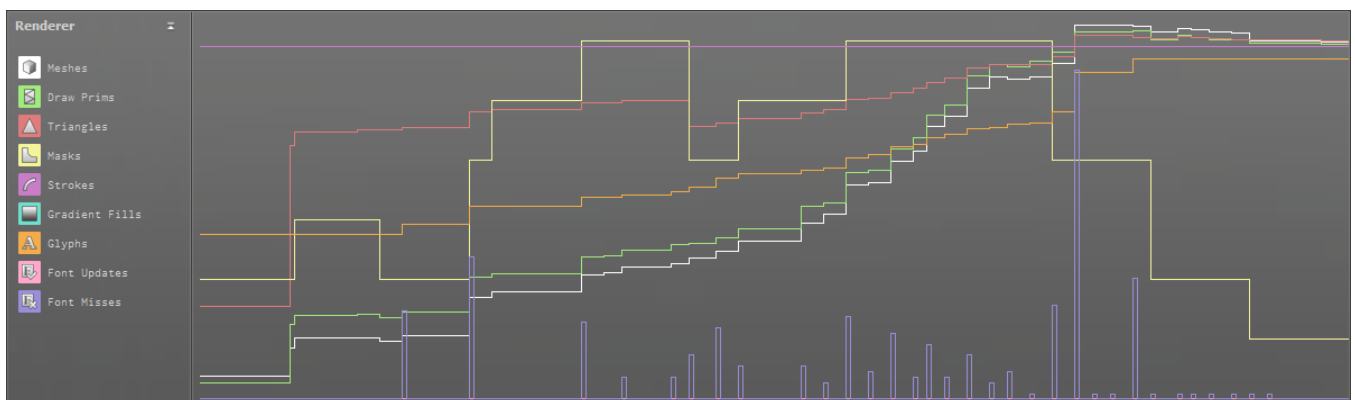


图 4: 渲染图表

与渲染图表相关的是应用程序底部的“渲染摘要” (Render Summary) 选项卡。

Log	SWF info	Renderer	Memory
Meshes:		182	
Draw Primitives:		183	
Triangles:		5124	
Masks:		6	
Filters:		0	
Strokes:		21	
Gradient Fills:		0	
Rasterized Glyphs:		412	
Font Cache Updates:		0	
Font Textures:		1	
Font Cache Thrashing:		0	
Font Cache Fill:		28%	
Font Cache:		1048576 pixels	
Font Cache Used:		298294 pixels	
Font Failures:		0	
Font Cache Misses:		5	
Mesh Thrashing:		0	

图 5：渲染摘要

2.4 分析内存消耗

从顶部数的第三和第四个图表分别显示 **Scaleform** 系统和图形内存使用情况：

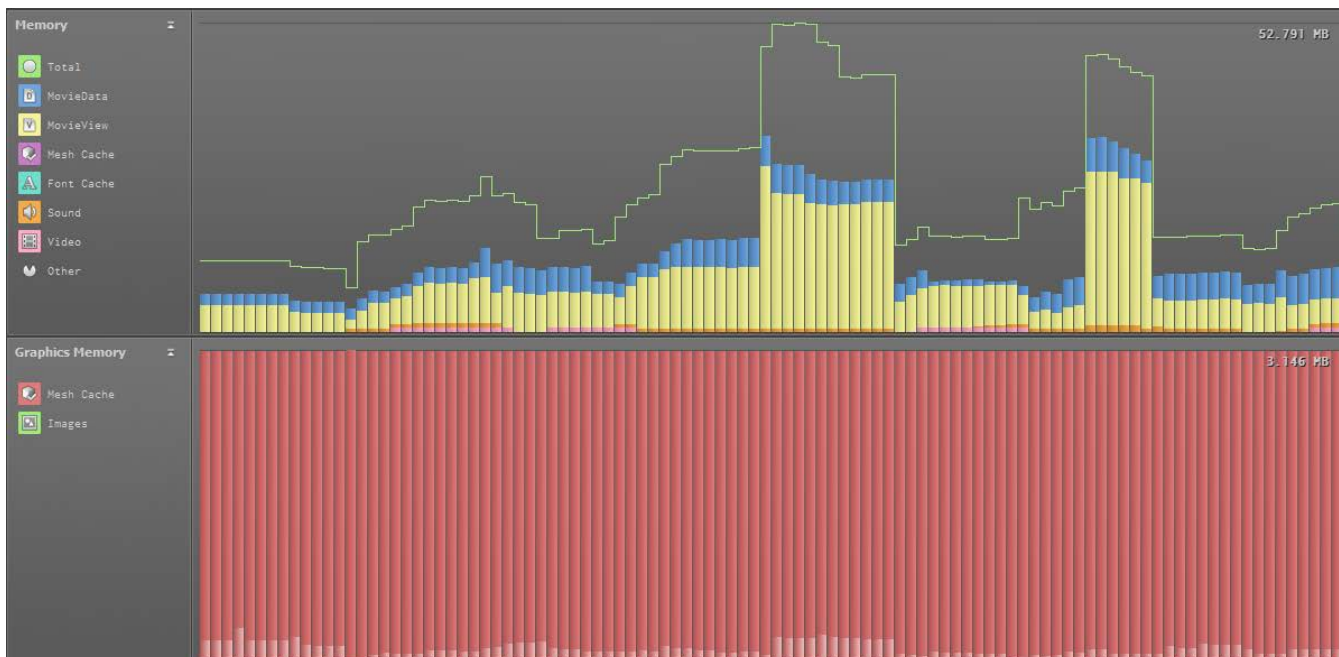


图 6：内存图表

- **总计 (Total):** Scaleform 要求的全部内存（足迹）。总内存画为一个线形图，包括下列全部类别，以及由于系统开销、被请求的分配的粒度以及碎片整理而未使用的内存。请注意，由于任何类别中都不显示未使用的内存，因而总内存大于使用的内存类别的总和。
- **电影数据 (MovieData):** 所有加载的 SWF 或 GFX 文件所占用的内存。复杂矢量图形对象、嵌入式字体以及时间线动画会影响这一数字。
- **电影视图 (Movie view):** GfX::Movie 实例占用的运行时内存，分配给时间线动画支持、屏幕对象和 ActionScript。
- **网格缓存 (MeshCache):** 为缓存的形状网格数据占用的内存，这些数据是由矢量拼嵌器 (Tesselator) 和边缘防混叠功能生成的。可以从系统或图形内存分配网格缓存内存，具体取决于平台。实际使用的内存以较暗的阴影显示，因此，用户可以容易地确定是否可以设置一个较小的网格缓存。
- **字体缓存 (Font Cache):** 字体缓存使用的内存。
- **声音 (Sound):** 用于嵌入式声音数据的内存。它受嵌入的声音示例的数量、长度或质量影响。
- **视频 (Video):** 用于视频内存回放缓存器的内存。这些缓存器是在视频启动时分配的，然后在其停止时释放。
- **其它 (Other):** 由 Scaleform 使用但并非上述类别组成部分的内存。有关内存类别的更加详细的细分信息，请切换下文介绍的详细内存报告。
- **图像 (Images):** 用于图像（纹理）的图形内存。

有关内存系统及其配置、优化和管理方式的更多信息，请参阅[内存系统概述 \(Memory System Overview\)](#) 文档。

“内存摘要”选项卡更详细地显示上述内存的类别。“内存” (Memory) 选项卡中的所有值的单位都相同（字节、千字节或兆字节），与在视图菜单下的内存单位选择中指定的相同。“内存”选项卡中的所有值（除非另有说明）均以字节计。请注意，“Used Space”（已用空间）是 GfX 有效使用的内存。“Unused Space”（未用空间）是 GfX 已经保留但主要由于碎片而尚未使用的内存。Debug Data（调试数据）表示 GfX 为跟踪内存和性能统计数据而在内部使用的内存。

当以完整内存报告模式（该模式通过 AMP 工具栏上的“i”按钮进行切换）分析时，内存报告变得更加详细，显示按堆、按文件和按用途分的明细数据。Scaleform 库的调试版本将堆分配分割成若干内存类别，用户可以由此更好地理解 Scaleform 的当前内存使用情况。下面介绍报告的内存类别：

- **Renderer（渲染器）：** Scaleform 渲染引擎分配的内存。此类别进一步细分为下列子类别：
 - **Buffers（缓冲区）：** 为网格暂存缓冲区 (Mesh Staging Buffer) 分配的内存，该缓冲区充当一个短期缓存，用来在将生成的网格实例化并复制到顶点和索引缓冲区之前存储这些网格。
 - **RenderBatch（渲染批次）：** 给实体（例如：网格、图元，或与渲染树中即将绘制在一起的节点关联的渲染命令）的集合分配的内存。
 - **Primitive（图元）：** 用来保存在一起绘制的顶点的集合的内存。
 - **Fill（填充）：** 用来存储和管理描述硬件填充的数据的内存。

- **Mesh（网格）**：给网格缓存分配的内存。网格缓存用来馈给包含顶点和索引缓冲区数据的渲染器。
- **MeshBatch（网格批次）**：给“网格缓存”的顶点和索引缓冲区中缓存的各个数据块分配的内存。
- **Context（上下文）**：渲染支持系统占用的内存，该系统用来管理渲染树的寿命。此上下文维护渲染树的多个逻辑上独立的快照，因而线程可作用于渲染数据，而不会彼此干扰。
- **NodeData（节点数据）**：用于帧状态信息（如应用到所渲染的对象的当前矩阵或效果）的内存。
- **TreeCache（树缓存）**：给渲染器的场景图表（所渲染的对象的一个树状结构）分配的内存。
- **TextureManager（纹理管理器）**：给管理纹理图像分配的内存。实际图像数据通常不存储在系统内存中，因此，不是此内存类别的组成部分。
- **MatrixPool（矩阵池）**：渲染器所维护的矩阵池占用的内存。
- **MatrixPoolHandles（矩阵池句柄）**：矩阵句柄分配所占用的内存。
- **Text（文本）**：给文本管理和渲染（如格式化信息、样式和颜色信息、html 解析、富文本数据和行缓冲区）分配的内存。
- **Font（字体）**：与字体渲染（如字形光栅化和字体缓存存储）相关的内存。
- **MovieDef（电影定义）**：给代表电影元素模板的不可变数据分配的内存。它包含下列子类别：
 - **CharDefs（字符定义）**：用于字符定义（如按钮和文本）的内存。
 - **ShapeData（形状数据）**：用于形状定义的内存。
 - **Tags（标签）**：用于 ActionScript 标签的内存
 - **Fonts（字体）**：用于字体资源的内存。
 - **Sounds（声音）**：用于声音数据的内存。
 - **ASBinaryData（ActionScript 二进制数据）**：ActionScript 缓冲区数据占用的内存。
 - **MD_Other（其它电影定义数据）**：上述类别中不包含的其它 MovieDef 数据。
- **MovieView（电影视图）**：给时间线动画支持、屏幕上对象以及 ActionScript 分配的内存。该类别进一步细分为下列子类别：
 - **MovieClip（电影剪辑）**：用于屏幕上动画对象的内存。
 - **ActionScript**：ActionScript 代码和数据占用的内存。
 - **ASString（ActionScript 字符串）**：ActionScript 字符串占用的内存。
 - **Text（文本）**：用于与文本相关的显示对象占用的内存。
 - **XML**：用于 XML 对象的内存。
 - **MV_Other（其它电影视图）**：不符合上述类别的其它电影视图内存。
 - **VM（虚拟机）**：分配给 ActionScript 虚拟机的内存。此类别进一步细分如下：
 - **AS3 VM（AS3 虚拟机）**：ActionScript 3 虚拟机占用的但不属于下列子类别组组成部分的内存。
 - **CallFrame（调用帧）**：执行 ActionScript 期间给调用栈中每个实体分配的内存。

- **Vtable（虚拟表）**：ActionScript 虚拟表占用的内存。
- **SlotInfo（插槽信息）**：用于描述 ActionScript 类成员的结构内存。每个 ClassTraits 和 InstanceTraits 都包含 SlotInfo 结构，以便于描述“类”(Class) 或“实例”(Instance) 内存布局。有关 ClassTraits 和 InstanceTraits 的更多信息，请参阅下文。
 - **SlotInfoHash（插槽信息哈希表）**：加快 ActionScript 的属性查找的哈希表 (Hash Table) 使用的内存。
- **ClassTraits（类特征）**：给保存 ActionScript 类的内存布局分配的内存。
- **Class（类）**：给保存 ActionScript 对象的静态成员分配的内存。
- **InstanceTraits（实例特征）**：给保存 ActionScript 对象的非静态内存布局分配的内存。
- **Instance（实例）**：给保存 ActionScript 对象的实例变量和方法分配的内存。
- **AbcFile（ABC 文件）**：给代表字节代码及其相应数据的数据结构分配的内存。
 - **AbcConstPool（ABC 常数池）**：用于存储常数（如数字、字符串、命名空间和 Multiname）的内存。这只是字节代码，因此它是上面的 AbcFile 的一部分。
- **VMAbcFile（虚拟机 ABC 文件）**：保存内部数据结构的内存，而内部数据结构存储从 AbcFile 生成的优化字节代码、依存关系以及其它信息。
- **Tracer（跟踪器）**：给代码优化分配的内存。
- **IME（输入法编辑器）**：用于支持输入法编辑器 (IME) 的内存。

Log	SWF info	Renderer	Memory	Image
		Memory		
		Memory	Size	
▲ Total Footprint			4849.402 KB	
▲ Used Space			4633.715 KB	
▷ Video Heaps			0.000 KB	
▷ Other Heaps			325.621 KB	
▲ Movie View Heaps			1226.094 KB	
▲ MovieView "3dgenerator_as3.swf"			1226.094 KB	
String			0.828 KB	
▷ Renderer			12.672 KB	
▷ MovieView			1176.781 KB	
General			35.812 KB	
▷ Movie Data Heaps			261.066 KB	
▲ Global Heap			2820.934 KB	
Video			264.176 KB	
String			20.633 KB	
Sound			0.141 KB	
▷ Renderer			2310.914 KB	
▷ MovieView			8.039 KB	
▷ MovieDef			116.676 KB	
Image			0.141 KB	
IME			9.914 KB	
General			90.301 KB	
▷ Debug Memory			8.000 KB	
▷ Unused Space			38.820 KB	
Debug Data			176.867 KB	

图 7：内存摘要

展开每个类别会显示负责该类别直接分配的子类别或内存堆（如果有的话）。以这种方式深入探讨，就会更好地了解任何特定帧中的 **Scaleform** 内存使用情况。

“图像” (Images) 选项卡还与内存分析相关，因为它显示在选定的帧内所加载的图像，以及其大小。

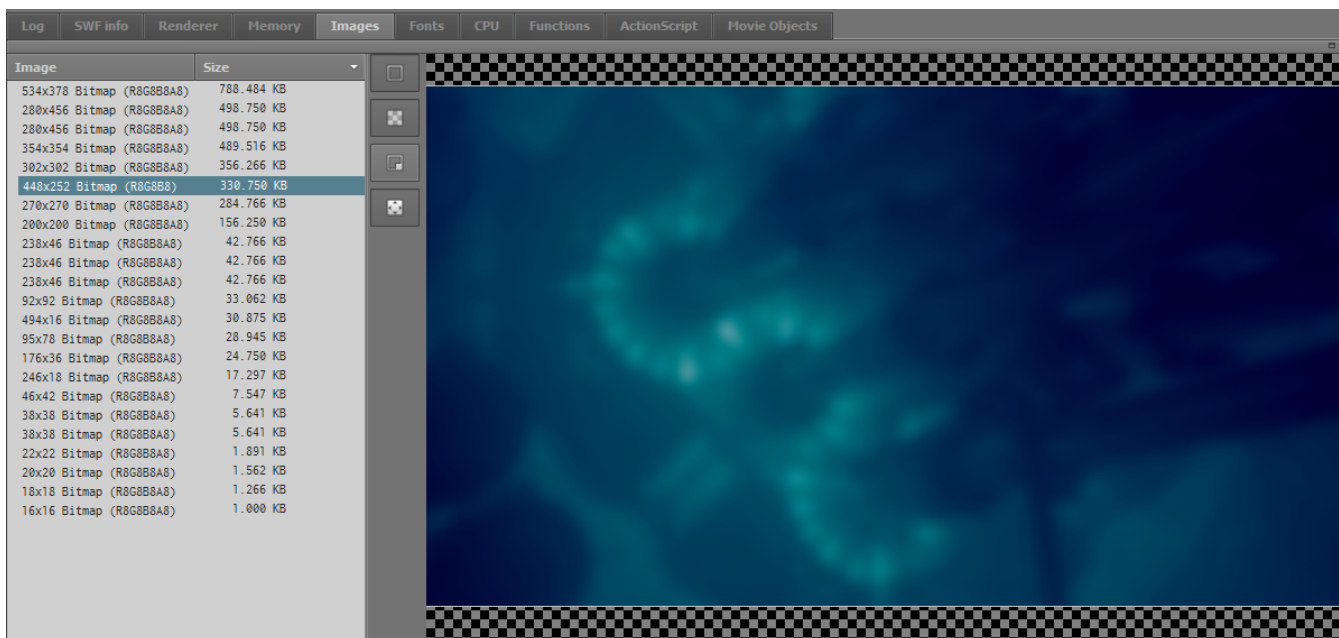


图 9：“图像内存”选项卡

使用此选项卡可快速检查是否存在增加目标 **Scaleform** 应用程序内存消耗的大图像。在图像列表中单击一个项目，可在预览窗格中显示该图像的缩略图。请注意，不是作为分析信息的组成部分来发送图像数据，而是在请求时发送。因此，如果连接的 **Scaleform** 应用程序已经卸载了该图像，就可能不会显示任何图像。

与内存分析相关的最后一个选项卡是“字体” (Fonts) 选项卡，该选项卡显示选定的帧的字体信息。



图 10：“字体内存”选项卡

使用此选项卡检查每个字体类型的嵌入式字形的数量（按每个 **SWF** 文件显示）。

嵌入式字体可消耗相当大的内存，这会在 **AMP** 中按照 **MovieData** 堆和 **CharDefs** 类别的增加的大小显示出来。这些可分别在“堆”选项卡和“内存摘要”选项卡中进行检查。可通过在多个 **SWF** 文件上共享字体来减小字体内存。

2.5 使用透支分析

AMP 可用来远程触发一种特殊的渲染模式，这种模式可突出 (Highlight) 被分析的应用程序中的潜在问题领域。要切换渲染模式，请单击应用程序控制工具栏中的适当按钮。

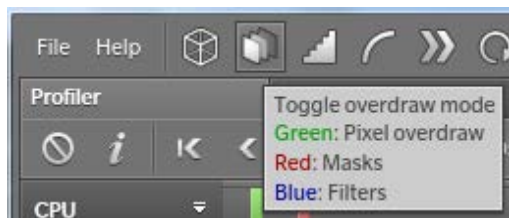


图 11：切换按钮

一旦切换此模式，就会以绿色渲染所有对象。绿色表示无效渲染（16 层达到全亮），遮罩为红色，过滤器为蓝色（8 层遮罩或过滤器达到全亮）。这些颜色混合在一起，因而一个明亮的黄色区域由于带有嵌入的遮罩而严重渲染无效。

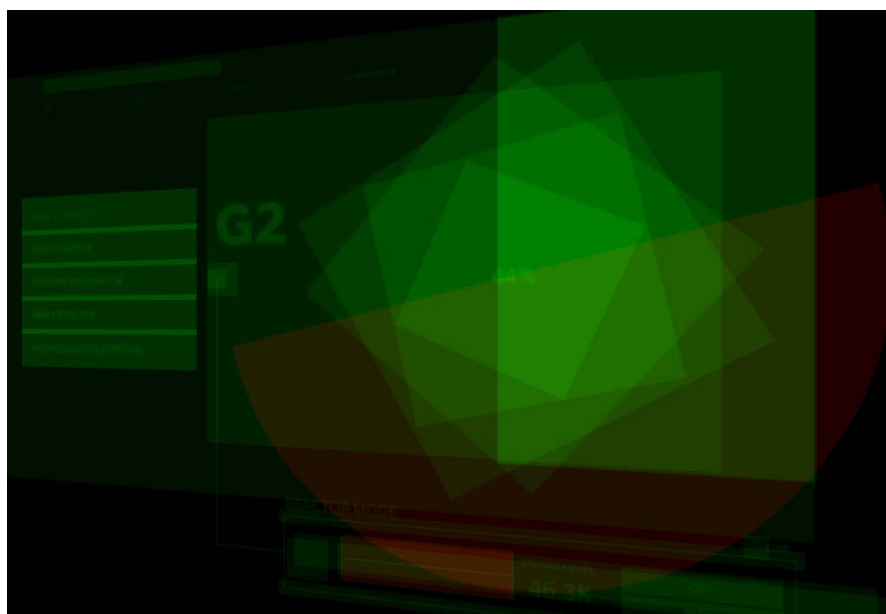
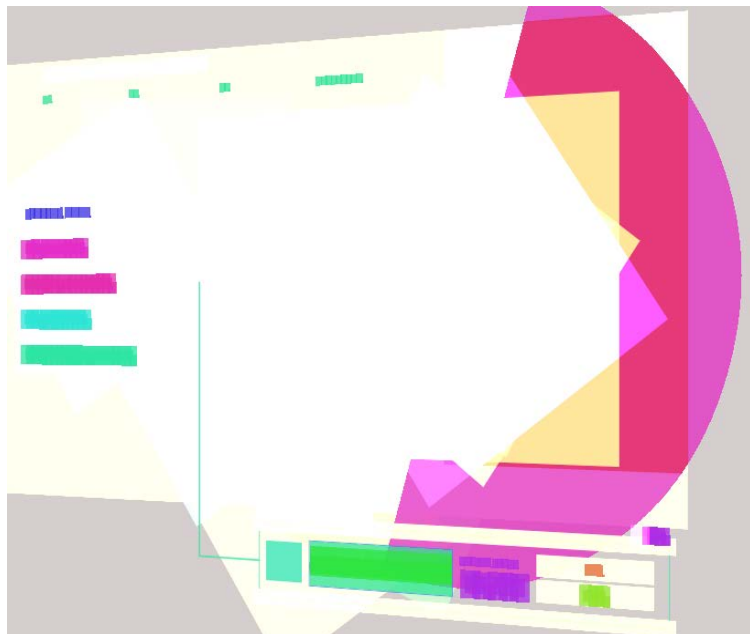


图 12：用于性能突出的渲染模式

无效渲染、遮罩和过滤器都会影响性能，应尽可能减小这种影响。

2.6 使用批量分析

AMP 支持的另一个特殊渲染模式是批量分析。要切换此渲染模式，请单击位于应用程序控制工具栏上的适当按钮。



一旦切换此模式，就会以一种不同的颜色渲染每个渲染批次。如有可能，通过最大限度地减少批次来实现最佳性能。

3 平台和集成说明

1. AMP 通常是在 `GFx::System` 对象构造期间或在 `GFx::System::Init` 函数调用期间初始化。或者，假如一个 `Saleform::System` 用来初始化 `Scaleform`，程序员就需要通过分别调用 `GFx::AMP::Server::Init()` 和 `GFx::AMP::Server::Uninit()` 来明确初始化和关闭 AMP。
2. AMP 服务器目前支持 Windows、Xbox360、PS3、Linux、Mac, Wii, Android 和 iPhone/iPad, 3DS, PS Vita, WiiU。使用 AMP 客户端的自动发现功能，连接非常简单。不过，不经自动发现的连接必需知道 AMP 服务器的 IP 地址，这会导致混乱，因为某些平台拥有多个 IP 地址。
 - 对于 Xbox360，请务必指定标题 IP 地址，而非调试 IP 地址。标题 IP 地址可通过查看 Xbox 360 包含的开发或测试工具箱的属性来确定。
 - 对于 PS3，如果开发工具箱配置为使用两个 IP 地址，请务必指定应用程序 IP 地址，而非用于调试的 IP 地址。
 - 对于 Android，需要在应用程序级启用互联网许可。为此，需要把下面的代码添加到应用程序的 `AndroidManifest.xml`：

```
<uses-permission android:name="android.permission.INTERNET" />
```
 - 请通过主机 PC 桥（而不是 WiFi 连接）来连接到 WiiU。主机 PC 通过自动发现端口 6003 来检测游戏机。一旦进行连接请求，就将通信从 TCP/IP 插槽切换到主机桥。
 - Wii 不使用网络套接口与 AMP 进行通信，而是使用内置到 NDEV 中的 USB 接口（COM 端口）来把控制台物理上连接到一台 PC。用户可以通过在 IP 地址字段中指定“wii”来从连接对话框连接到 Wii AMP 服务器。重要提示：用户必须确定 `RVL_SDK/X86/bin` 在环境变量里。这是为确保 AMP 能运行 `hio2.dll`。
3. 某些游戏引擎（如 Unreal Engine 3）采用一种特定的方式打包资源，使得 AMP 服务器无法查找与所分析的 Flash 内容相对应的 AS2 SWD 文件。倘若如此，请务必将 SWD 文件放在 AMP 客户端工作目录中，而不是依靠 AMP 服务器在与 SWF 相同的目录中找到它们并通过网络将其发送出去。

4 构建注释

1. 默认情况下，AMP 是所有非装运 Scaleform 构建。要从 Scaleform 删除 AMP，请确保没有在 GfxConfig.h 中定义 SF_AMP_SERVER，并重新构建 Scaleform 库。如果 Scaleform 源访问不可用，可以通过调用

```
Ptr<Gfx::AMP::ServerState> serverState = *SF_NEW Gfx::AMP::ServerState();  
serverState->StateFlags &= Gfx::AMP::Amp_Disabled;  
AmpServer::GetInstance().UpdateState(serverState);在运行时禁用 AMP。
```

2. AMP 使用网络套接口 (Network Socket) 与分析器进行通信。因此，请确保应用程序与适当针对平台的网络库相链接 (Ws2_32.lib 适用于 Windows, Xnet.lib 适用于 Xbox360, libnetctl 适用于 PS3, 套接口适用于 Linux, 等等)。。
3. 如果 AMP 只是使用网络套接口的系统，它就会初始化套接口库，并在完成后发布它。在某些系统上，如果初始化的次数大于发布的次数，则发布是无效的。即使出现这种情况，也没有大碍。不过，在其他平台上，发布网络库立即生效 --- 即使该库初始化次数大于发布次数，而且 AMP 可能会干扰应用程序其他部分中使用的套接口连接。倘若如此，可以强制 AMP 跳过套接口初始化，并通过调用 AMP::Server::GetInstance().SetInitSocketLib(false) 来使用以前初始化的套接口库。此调用需要在初始化 AMP 后执行。
4. 可以针对 AMP 内存堆设置一个极限。当超过此极限时，就会暂停 Scaleform，直到将任何等候的消息发送到分析器。此极限是通过调用 AMP::Server::GetInstance().SetHeapLimit() 来设置的。
5. 默认情况下，AMP 服务器将在端口 7534 上创建一个侦听网络套接口。可以通过调用 AMP::Server::GetInstance().SetListeningPort() 来设置一个不同的端口。
6. AMP 可以配置为暂停执行，直到与分析器客户端建立一个连接。这对于在启动时获得统计数据非常有用，而且是在启动时通过进行以下调用来完成的：
AMP::Server::GetInstance().SetConnectionWaitTime。

5 添加 AMP 支持

按照本节所述的步骤，任何 Scaleform 应用程序都可以配置为通过 AMP 来支持远程分析。另外，还可以检查 Scaleform Player 源来查找到一个配置齐全的 AMP 服务器实例。

5.1 Debug 文件生成

AMP 能够使用 ActionScript 2 Flash 调试信息（SWD 文件）来显示源代码和每行计时数据。通过在 Flash CS3 或 CS4 内运行调试程序 (Ctrl+Shift+Enter) 来生成 SWD 文件。请将 SWD 文件要么放在与相应的 SWF 相同的位置，要么放在 AMP 客户端的工作目录之中。

5.2 应用程序控制

AMP 能够远程控制被分析的应用程序中的设置，例如，线框模式或批量分析模式。许多支持的设置都是针对应用程序的，而且可以通过本应用程序加以实现。不实现的任何此类功能都只是不能通过 AMP 客户端来使用。

请按照下述步骤处理发自 AMP 的应用程序控制消息：

- 实现一个从 AMP::AppControlInterface 派生的自定义应用程序控制回调类，并改写 HandleAmpRequest 方法来处理来自 AMP 的 AMP::MessageAppControl 消息。
- 通过在启动时调用 AMP::Server::GetInstance().SetAppControlCallback 来安装自定义处理程序。
- 通过调用 AMP::Server::GetInstance().SetAppControlCaps 来告知 AMP 支持的功能。此方法的参数是一个 AMP::MessageAppControl 消息，将支持的功能设置为 True（真）。例如：

```
AMP::MessageAppControl caps;  
caps.SetCurveToleranceDown(true);  
caps.SetCurveToleranceUp(true);  
caps.SetNextFont(true);  
caps.SetRestartMovie(true);  
caps.SetToggleAaMode(true);  
caps.SetToggleAmpRecording(true);  
caps.SetToggleFastForward(true);  
caps.SetToggleInstructionProfile(true);  
caps.SetToggleOverdraw(true);
```

```
caps.SetToggleBatch(true);
caps.SetToggleStrokeType(true);
caps.SetTogglePause(true);
caps.SetToggleWireframe(true);
AMP::Server::GetInstance().SetAppControlCaps(&caps);
```

5.3 连接状态

AMP 客户端能够在状态栏中显示连接的应用程序的名称。此信息是通过调用

`AMP::Server::GetInstance().SetConnectedApp` 来传递的。

另外，AMP 客户端还可以显示当前应用程序状态，但是为此，无论何时应用程序状态发生变化，都需要告知 AMP 客户端。使用下列方法来更新状态：

- `AMP::Server::GetInstance().SetState`，将第一个参数作为下列 `ServerStateType` 枚举类型之一：
 - `Amp_RenderOverdraw`
 - `Amp_App_Wireframe`
 - `Amp_App_Paused`
 - `Amp_App_FastForward`
- `AMP::Server::GetInstance().SetAaMode`
- `AMP::Server::GetInstance().SetStrokeType`
- `AMP::Server::GetInstance().SetCurrentLocale`
- `AMP::Server::GetInstance().SetCurveTolerance`
- 或者，可用上面的全部信息填充一个 `AMP::ServerState` 对象，并将该对象传递到 `AMP::Server::GetInstance().UpdateState`

5.4 标记

AMP 可以在 CPU 图表上显示与在被分析的应用程序内进行的 C++ 或 ActionScript 调用相对应的特殊指标、标记。按如下所示方式在 ActionScript 中添加标记：

```
Amp.addMarker(1)
```

注意：编译包含 `Amp.addMarker()` 代码的 .as 文件可能会产生某种编译错误。要避免这一点，务必使用 `Resources/AS2/CLIK` 目录下提供的固有 `Amp` 类。

按如下所示方式在 C++ 中添加标记：

```
GFx::MovieImpl::AdvanceStats->AddMarker(1)
```

AMP 就会在进行过调用的帧上显示一个标记。将来将使用整型参数显示一种以上类型的标记。

6 与 AMP 相关的常见问题解答

本节包含开发者在使用 AMP 工具时可能会遇到的其中几个问题。

1. 游戏端使 AMP 工作所需的最低限度是什么？

您需要一个启用 AMP 的内部版本（即定义了 `GFX_AMP_SERVER`）。默认情况下，GFX 的所有非发售版本均启用 AMP。另外，如果您的游戏不调用 `Platform::RenderThread::drawFrame`，您就需要确保每个帧都更新了 AMP，这是通过调用 `AmpServer::GetInstance().AdvanceFrame` 实现的。

2. 使用 `AmpClient` 连接我们的游戏时，我们的游戏的帧速率变得非常缓慢。我们发现 `AmpServer::AdvanceFrame()` 花费大量时间。这正常吗？

就多数分析器而言，连接 AMP 时出现性能损失属于正常情况。要最大限度地减小此性能影响，请确保以较低分析等级启动，这可通过从 AMP 客户端工具栏上的适当下拉控件中进行选择来设定。然后，一旦查找到瓶颈，您就可以在瓶颈处提升分析等级，以更加详细地确定根源。每个帧都请求完整内存报告也可能对性能产生重大影响，因此，请确保仅在需要查阅时才生成这些报告。您可以通过 AMP 客户端工具栏上的“i”（小写 I）来切换详细内存报告的开与关。

3. 我在 Windows/PS3/Xbox 上运行游戏，而且能够在发现窗口中找到 AMP 客户端后与其连接。不过，所有统计数据均为空白。

AMP 每个帧都需要收到信息，以便于显示内存和性能统计数据。如果您的游戏不调用 `Platform::RenderThread::drawFrame`，您就需要确保每个帧都更新了 AMP，这是通过调用 `AmpServer::GetInstance().AdvanceFrame` 实现的。

4. AMP 为何在更新时暂停一秒钟或更长时间（`AMP::Server::WaitForAmpConnection`）？

正常情况下，AMP 服务器在占用内存太大时会暂停一秒钟，以便于将积聚的数据发送到客户端。如果这种现象继续发生，那就可能是您的内容产生过多分析信息，以至于 AMP 如果不暂停 Scaleform 就没有时间将这些信息发送到客户端。

您可以通过关闭源代码行计时和详细内存报告（即 AMP 客户端上的 'i' 按钮）来减少所发送的分析信息的数量。AMP 客户端中也有一个‘分析等级’（profile level）下拉控件，也可以进行降低。最后，您可以升高触发此问题的内存阈值。默认数值设置为 1MB，但您可以通过调用 `GFX::AMP::Server::SetHeapLimit` 进行修改。

5.

默认情况下，AMP 堆限制为 1MB，尽管这实际上不是一个硬性限制，因此它是可以超过一点的。您可以通过调用 `AmpServer::SetHeapLimit` 在您的游戏中更改此限制。此限制越大，在将数据发送到 AMP 客户端时 GFx 必须暂停的可能性越小。

7 更多信息

有关 AMP 的更多信息，请查阅以下资源：

- [AMP 使用案例 \(AMP Use Cases\)](#) 文档。
- [Scaleform Reference 文档](#) – 查阅有关代码/类的详细信息。
- [AMP Forum](#) – 关于社区问题和支持。