兩相結合效能更佳：將 Arm Mobile Studio 與 Unity 整合

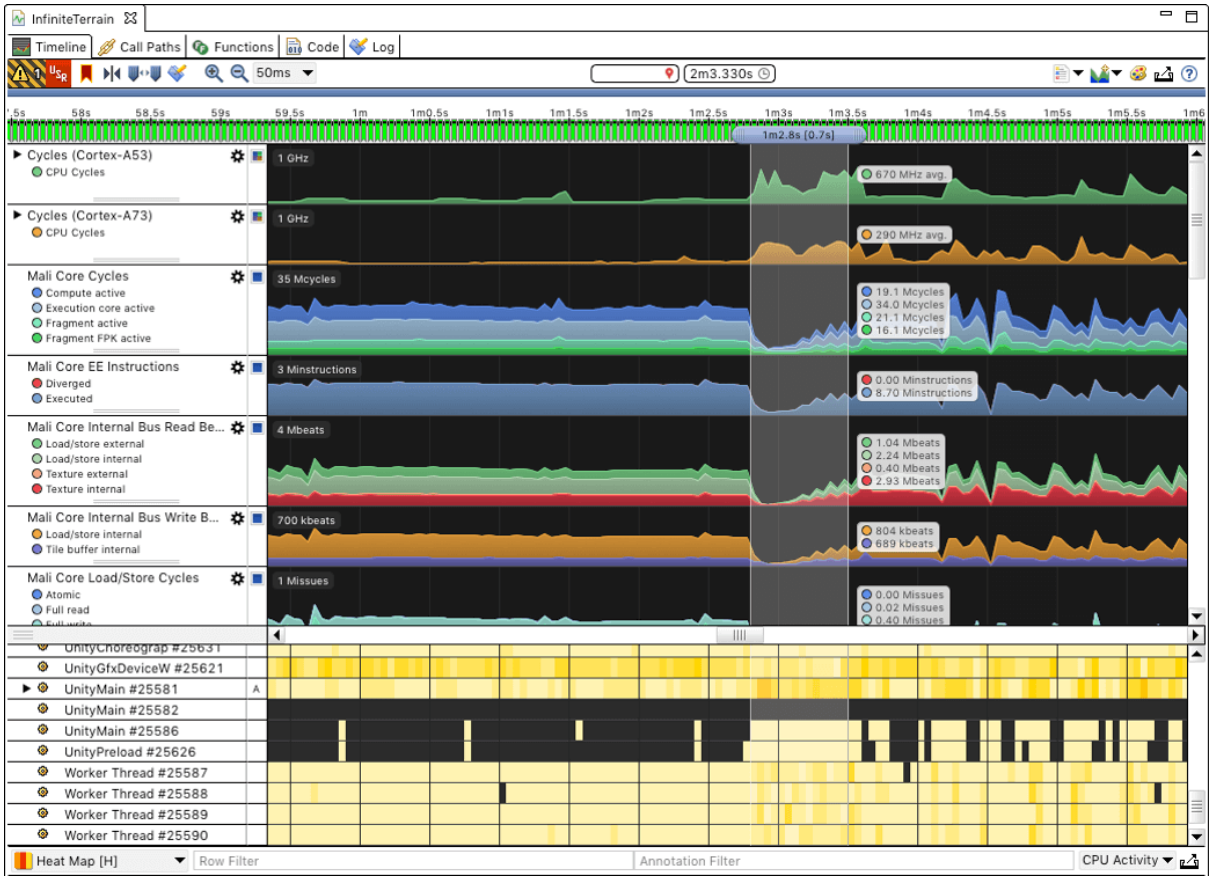
從最新高階頂級智慧型手機到大眾市場機型或較舊型裝置，行動裝置遊戲開發人員竭盡所能地確保其遊戲內容在這些廣泛多樣的裝置上，都能發揮良好效果。隨著行動裝置遊戲內容的複雜度日益增加，開發人員仰賴優質工具為他們提供所需的洞悉功能，以維持穩定的畫面播放速率以及低耗電率。

在這篇部落格文章中，我將說明新的 Arm Mobile Studio 工具集如何協助進行 Android 效能分析，以及如何與遊戲引擎搭配運作，進而產生令人更加愛不釋手的效能分析功能。Arm Mobile Studio 的 Starter Edition 為免費提供，而本文中使用的範例原始碼是取自 github。

更明確來說，我將著重於說明 Streamline，這是 Arm Mobile Studio 中用途範圍最廣泛且效能分析最詳細的元件。我將說明如何將 Streamline 與 Unity 整合，實際展現遊戲在行動裝置上運用 Arm Cortex-A CPU 或 Arm Mali GPU 資源的不同層面。

Streamline 簡介

Streamline 會從 Android 裝置上的多個來源收集以樣本與事件為依據的效能資料，並以數種不同檢視畫面顯示彙總結果，其中的「時間軸」檢視便是本文將著重說明的部分。下列畫面的上半部顯示已收集到的系統效能計數器，下半部可顯示同一時間軸上各種不同類型的資訊。此畫面下半部顯示「熱圖」，其中指出運算活動在目標應用程式中不同執行緒上的分佈方式：

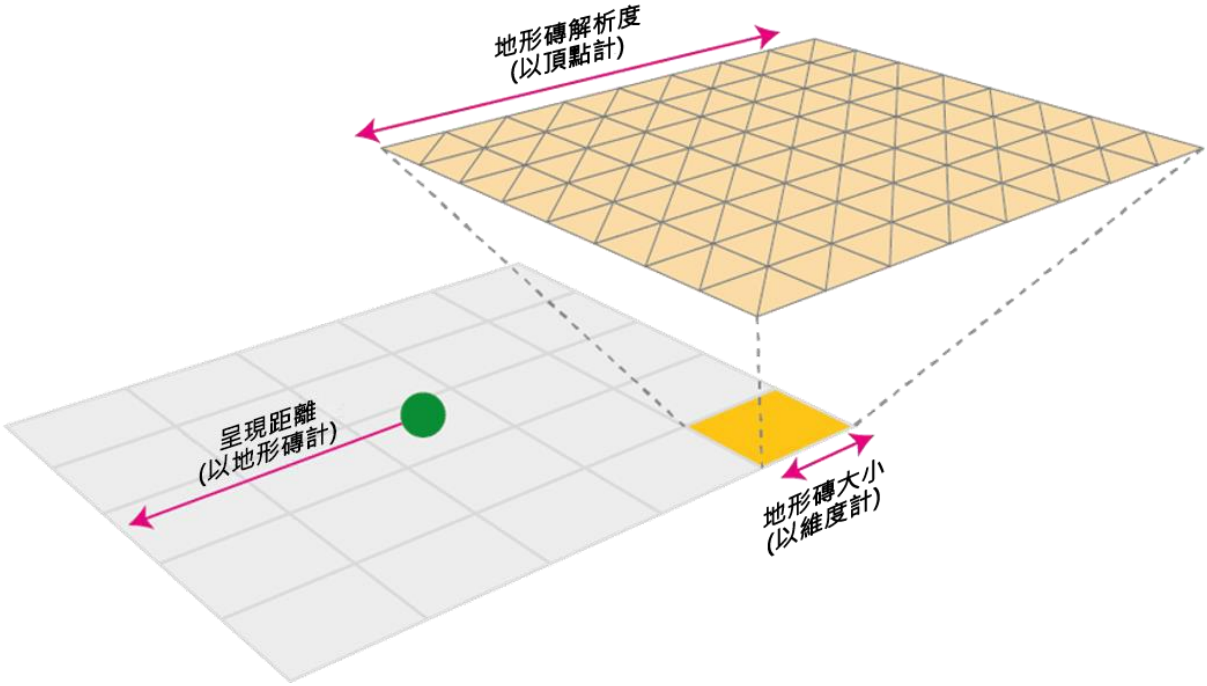


我們將要分析的內容

我們將分析一些非常簡單的 Unity 內容：飛越過程中動態產生的地形。鏡頭會搖動和轉向，在需要新的地形磚時，會動態產生地形磚。地形磚離鏡頭太遠時會被刪除，所以一旦場景填滿之後，隨著時間推移，此場景的複雜度依然會大致維持相同。有時鏡頭會緩慢移動，因此新地形產生速率就會變得非常緩慢，接著鏡頭會加速，這時就需要加快地形產生的速率。每塊磚的邊緣皆以較暗的顏色呈現，你可以輕易看出每塊磚的大小：

每個地形磚的產生都需要大量運算，而 Unity Job Scheduler 則讓我們能分派背景執行緒，以免耽擱主要的 Unity 執行緒。這會確保每當有新地形產生時，使用者可以體驗到穩定的畫面播放速率，而不是突然停頓的畫面。

此示範是設定為飛越四個不同場景，雖然四個場景看起來相同，但其中的地形磚卻是以不同方式產生。如以下圖表所示，地形是由許多地形區塊構成，而且每個區塊都是固定大小。每個區塊則由數種網格構成 (一種用於呈現綠色地形，一種用於呈現黃色地形，一種用於呈現水)，這些都有固定解析度。「呈現距離」會控制在遊戲角色周圍產生的地形磚數目。



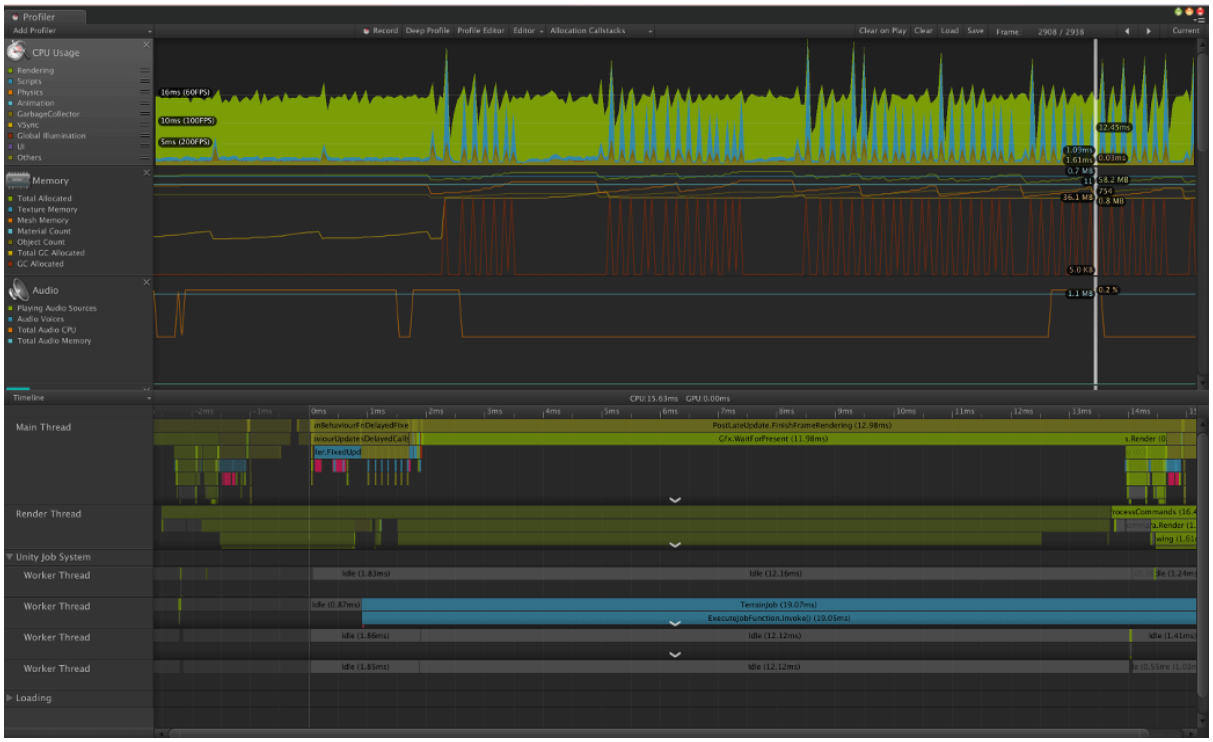
四個場景的設定如下表所示：

場景	呈現距離	地形磚大小	地形解析度	同步地形產生工作數目
1	3	20x20	32x32	8
2	3	20x20	32x32	1
3	6	10x10	16x16	8
4	6	10x10	16x16	1

我將在 2017 年發行的華為 P10 手機上進行剖析活動。此款手機內含一個 HiSilicon Kirin 960 晶片，其中包含四個高效能 Arm Cortex-A73 CPU 核心、四個高效率 Arm Cortex-A53 CPU 核心，以及 Arm Mali-G71 MP8 GPU。

Unity 內的剖析

Unity 本身包含一個剖析工具，而且在 Android 裝置上的運作效能絕佳：



Unity 的剖析工具在顯示作業排定時間上極為出色，但是不會顯示平台實體資源 (在本文中指的是 CPU 和 GPU) 的詳細資料，以及這些資源的使用情形。我們或許達到 60 FPS 的目標，但是否也為此而達到所有 CPU 核心的最大極限，並且耗盡電池電力？在這種情況下，Streamline 正好能派上用場。本文接下來的部分將展示 Streamline 會擷取和顯示哪些資料，以及我們能如何使用 Streamline 的*註解*功能，將一些高階情境從 Unity 遊戲傳遞到 Streamline，讓資料更容易解讀。

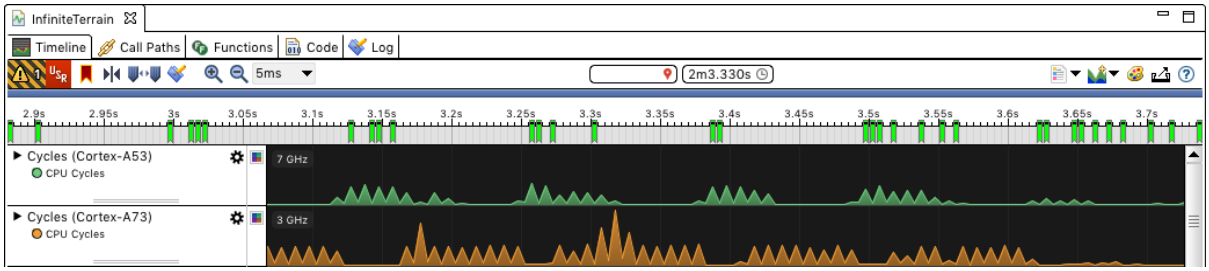
用 Streamline 剖析 Unity – 相關功能

在開始說明如何將註解插入 Unity 遊戲之前，先來看看當我們將遊戲修改為利用 Streamline 的三種註解功能時，範例內容的最終結果會是什麼樣子。

- **「標記」**是最簡單的註解形式，在單一時間點會有一個標籤出現在 Streamline 的「時間軸」檢視畫面的頂端。
- **「註解欄」**則提供多一點結構，會在每個執行緒旁提供另一行資訊列。註解可以置入註解欄內，與標記不同的是，註解欄內的每個註解都會用來標記一段時間範圍。
- **「自訂活動圖」**是最進階的註解形式，是一種用來顯示有複雜相依性的全域 (跨執行緒) 活動的機制。每個「自訂活動圖」會以自己的檢視畫面顯示在 Streamline 使用者介面下半部。

在我們收集到一個剖析檔之後 (稍後將更詳細說明)，此檔案會在 Streamline 中開啟，我們就能開始探索檔案中的剖析結果。

在檢查內容時，首先映入眼簾的是「標記」(時間軸頂端的綠色標記)，在圖中標出每個畫面的開始時間點：



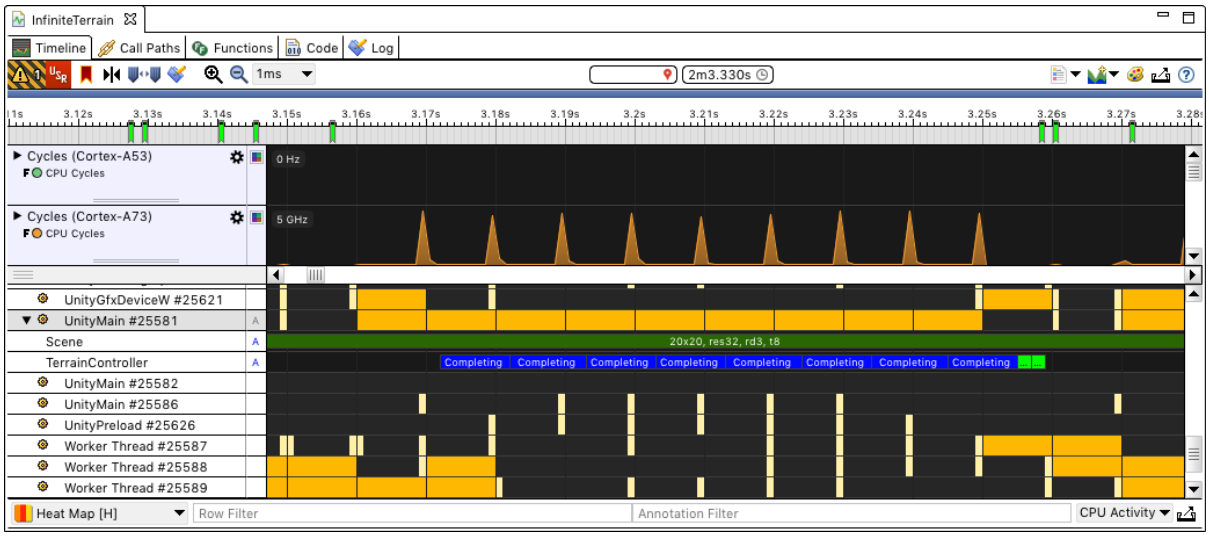
畫面播放速率並不如我們想像般的規律，而且在所有的 CPU 核心中皆有相當多的暴增活動。畫面播放速率一開始很緩慢，接著似乎加快，並且偶有停頓。這與我們對於地形產生時的預期情況相當一致，我們是否可以更深入探究呢？

Streamline 中的「時間軸」檢視畫面分成兩個部分 – 上半部顯示指標圖，下半部能顯示各種不同項目，包括熱圖。熱圖會顯示工作在整個系統中的分布方式，並讓我們能篩選最上方的時間軸，只顯示歸屬於特定流程或執行緒的工作。藉由查看熱圖，首先點選「UnityMain」執行緒，然後點選所有的「Worker Thread」，就能看到 CPU 活動如何劃分到主要 Unity 執行緒與作業排程器內的執行緒：

「UnityMain」執行緒相關的 CPU 剖析檔顯示 Cortex-A73 CPU 上的活動大量暴增。

所有「Worker Thread」相關的 CPU 剖析檔顯示 Cortex-A73 與 Cortex-A53 CPU 上的活動較小量暴增。

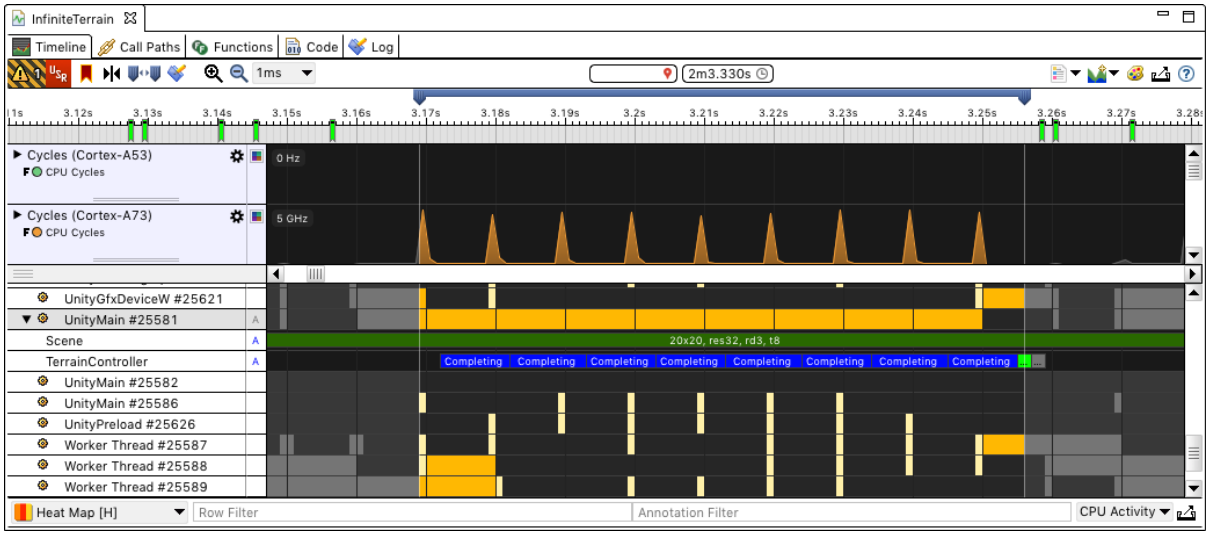
接著一起來看看主要執行緒。如果仔細檢視左手邊的螢幕擷取畫面，會看到目前正在檢查的「UnityMain」執行緒旁有一個「A」標記。這表示存在著 Streamline 註解欄。稍微拉近點查看時間軸，並展開「UnityMain」執行緒，看看是什麼情形：



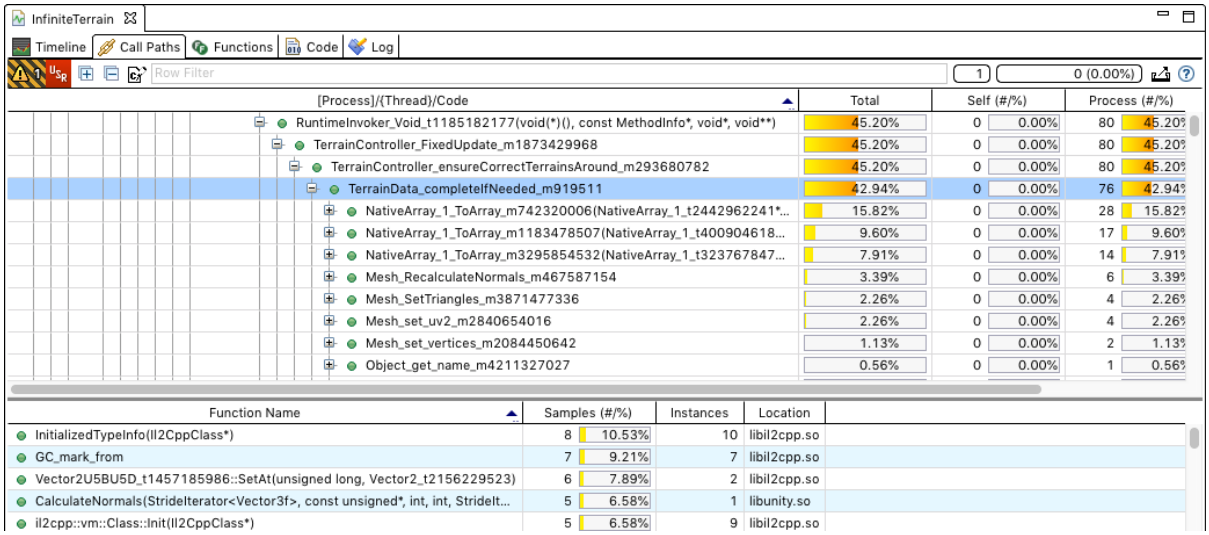
「Scene」與「TerrainController」列都是 Streamline *註解欄*，是由放入遊戲中的註解所產生。「Scene」註解欄顯示目前正在執行哪個場景 – 可以看到這個場景的地形磚大小為 20x20，地形解析度為 32x32，呈現距離為 3 且有 8 個同步地形產生工作。

「TerrainController」註解欄用於指出我們特別感興趣的程式碼片段在何時於主要 Unity 執行緒上執行。藍色區塊標出某一地形工作完成時所執行的程式碼。綠色區塊標出排定新地形產生的位置。在這裡可以看到，主要執行緒的所有活動基本上是起因於某一作業完成時需要進行的工作，而且需要產生最後網格，並將其插入此場景中。

除了關注特定執行緒，我們也可以將分析侷限在特定時間段。Streamline 的*游標*讓我們能標出要分析的特定時間範圍，在此圖中，已選出與地形完成相關活動激烈期的開始與結束時間 (游標設定在「時間軸」檢視畫面頂端)：



如果現在移到「呼叫路徑」檢視畫面，就能相當清楚地看到，在游標所選定的時間範圍內，時間花費在哪些地方。由於我們[針對 Unity 使用 IL2CPP 指令碼後端]，所以比起使用預設 Mono 執行階段，我們獲得的資訊多出許多。雖然在此不會深入探究此圖的詳細情形，但顯然有發生許多活動，值得更深入探究：

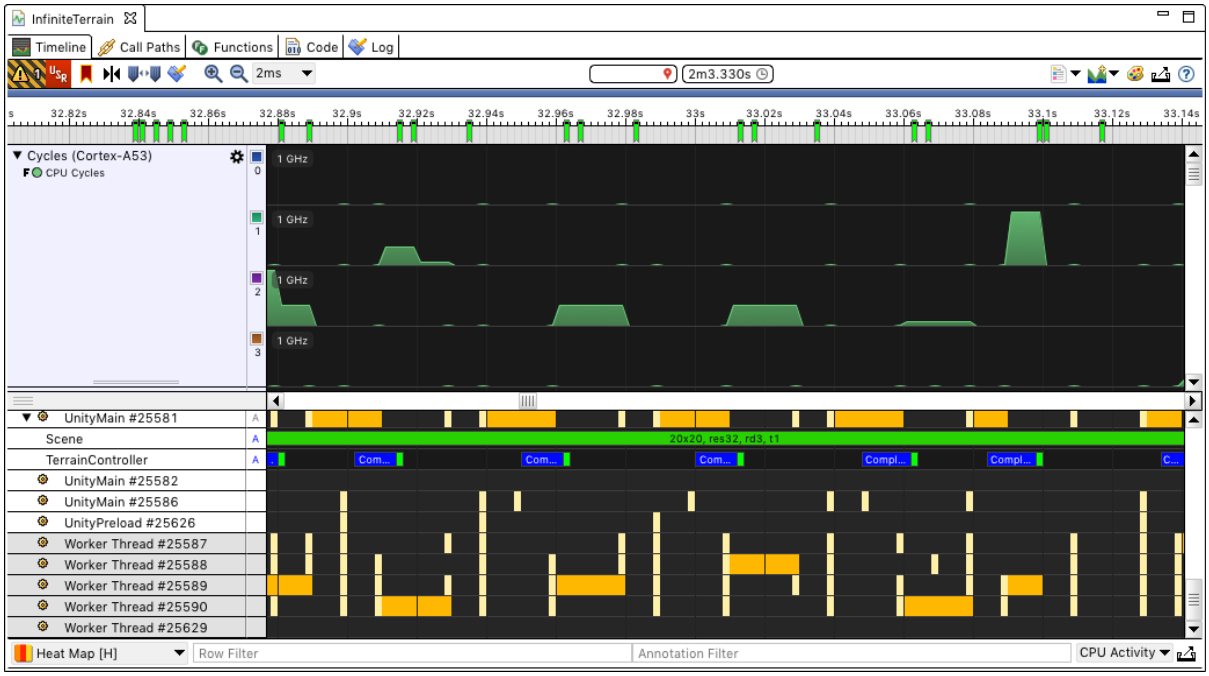


Worker Thread 的情形

當我們設定篩選條件只顯示「Worker Thread」時，在此顯示的結果並未出乎我們的意料，畢竟我們要求平行執行上限八項作業。在此螢幕擷取畫面中，我們展開了 Cortex-A53 叢集，因此可以看到各個核心的利用率。

「TerrainController」註解欄內的一些綠色區塊顯示目前正在排定新地形，接著看到所有核心中都有一些激烈活動，再接著「TerrainController」中的一些藍色活動會在新地形產生之後，立即在主要執行緒中處理這些地形 (由於我們未選取「UnityMain」執行緒，所以在圖表中看不到主要執行緒的活動)。

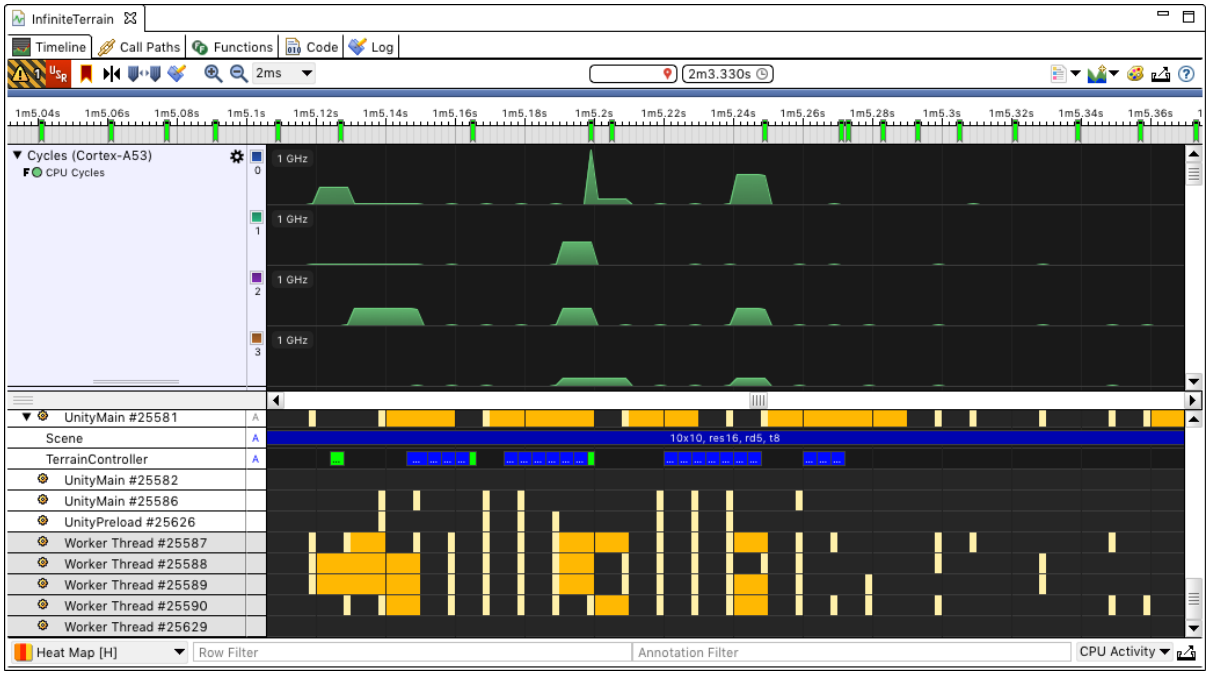
將此場景與第二個場景做比較，會看到有趣的差異，在第二個場景中地形磚的複雜度與第一個場景相同，但我們只允許一個同步地形產生工作：



此處有兩點值得留意：

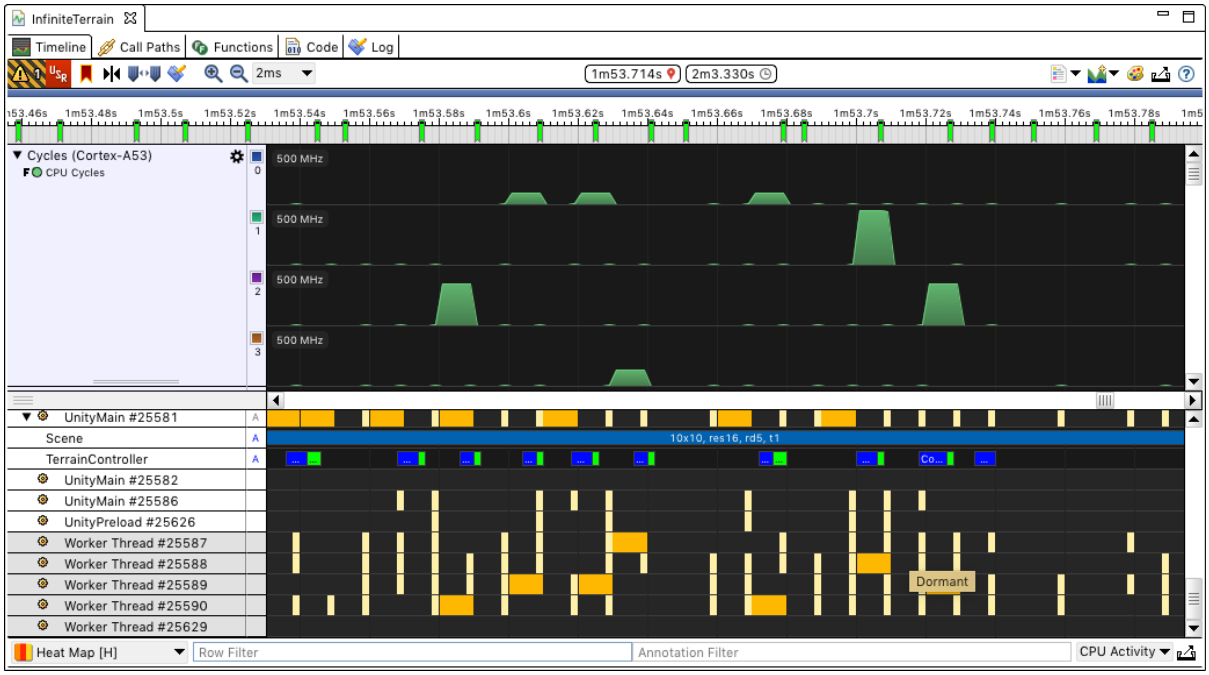
1. CPU 活動的激烈程度降低許多 – 大多數的核心在大部分時間都處於閒置或接近閒置狀態。
2. 畫面播放速率雖然並不完美，但順暢許多。由於在任何一個時間點都只有一個畫面完成，所以會減少在主要執行緒上造成耽擱的大量活動暴增。

也可以將剖析檔與第三個場景比較，第三個場景使用較小的地形磚：

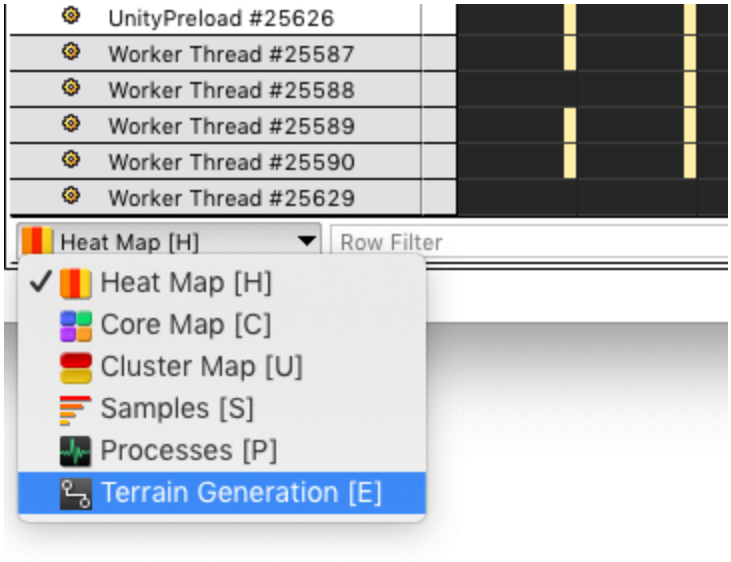


我們可以看到 CPU 活動的激烈程度降低許多，而且在主要執行緒中，已完成工作的藍色區塊也短了許多，因此畫面播放速率會比第一個場景順暢 (但是當然整體上會有更多作業，因此我們必須確保地形產生速率仍然能跟上鏡頭飛越地形的速率)。

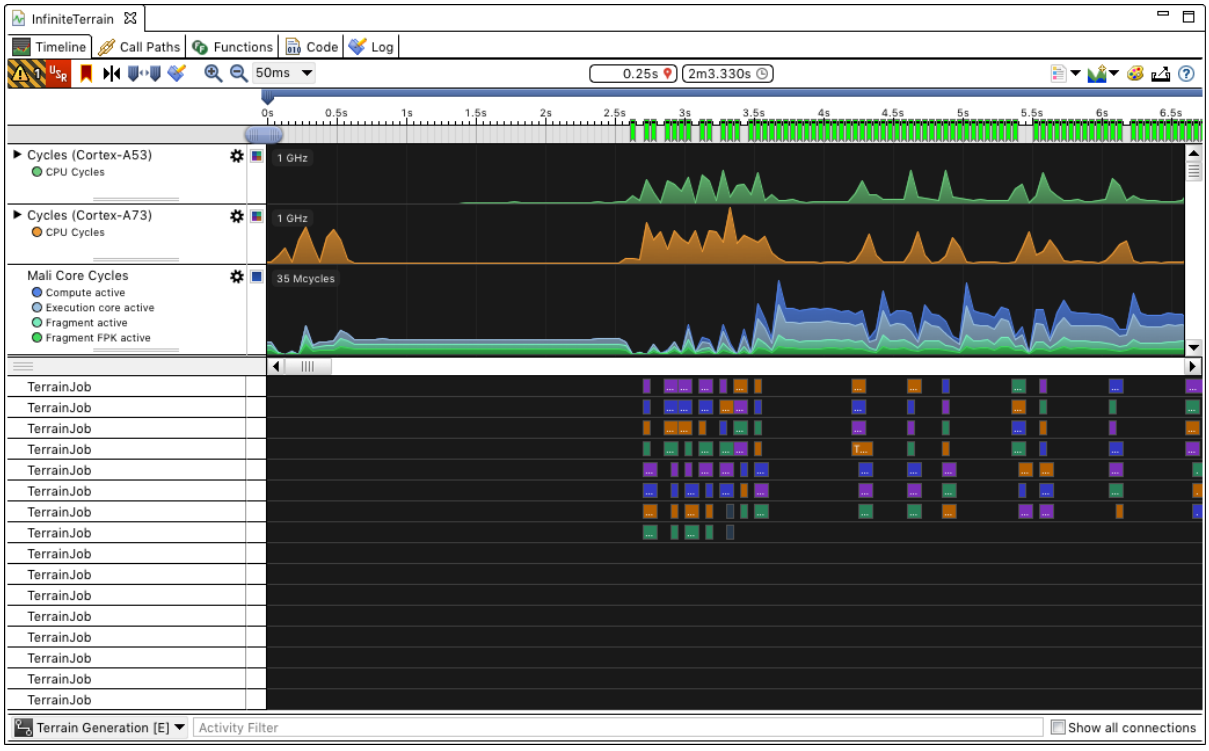
第四個場景使用小塊地形磚，而且同一時間只進行一次同步地形產生工作，這顯示出整體最順暢的畫面播放速率，但是我們必須非常謹慎，以確保地形產生速率夠快，足以跟上鏡頭的速率，而在原始影片中會看到，當鏡頭快速飛越第四個場景時，地形產生速率並非總是能跟上鏡頭的速率。



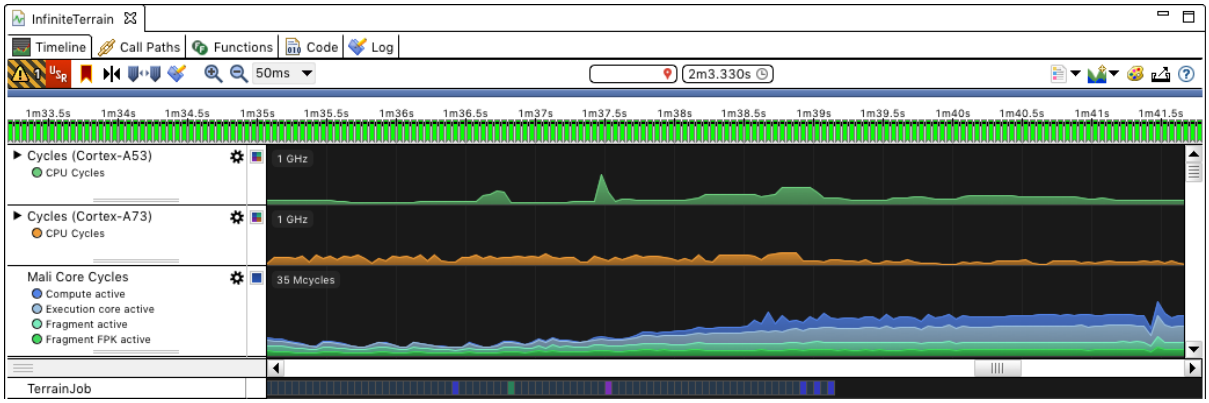
最後，我們可以使用「自訂活動圖」，更深入解析 Worker Thread 如何進行地形產生作業。每個「自訂活動圖」都會以選項形式出現在左下方功能表上，到目前為止我們一直使用此功能來顯示熱圖：



當我們選取「地形產生」檢視時，會看到每個地形產生活動都有一個色塊，上面顯示地形生成活動的開始與停止時間，如果將滑鼠游標移到色塊上方，就會顯示地形磚的座標、起始時間，以及完成所花時間。在此螢幕擷取畫面中，我們以圖表顯示在 Mali GPU 上進行的運算工作，正如我們的預期，隨著地形逐漸填滿，GPU 活動也穩定增加。此螢幕擷取畫面的擷取時間點是在第一個場景開始時，當時我們正在產生大塊地形磚，最多同時產生 8 塊。主要執行緒因準備所有新網格期間所發生的停頓，導致 GPU 長時間閒置：



接著移到第四個場景，在這裡我們依序產生較小塊地形磚，可以看到 GPU 活動上揚的幅度順暢許多，而且可以明顯看到，同一時間只有一項地形作業正在執行 (而且因為使用較小塊地形磚，所以每項作業都較短)：



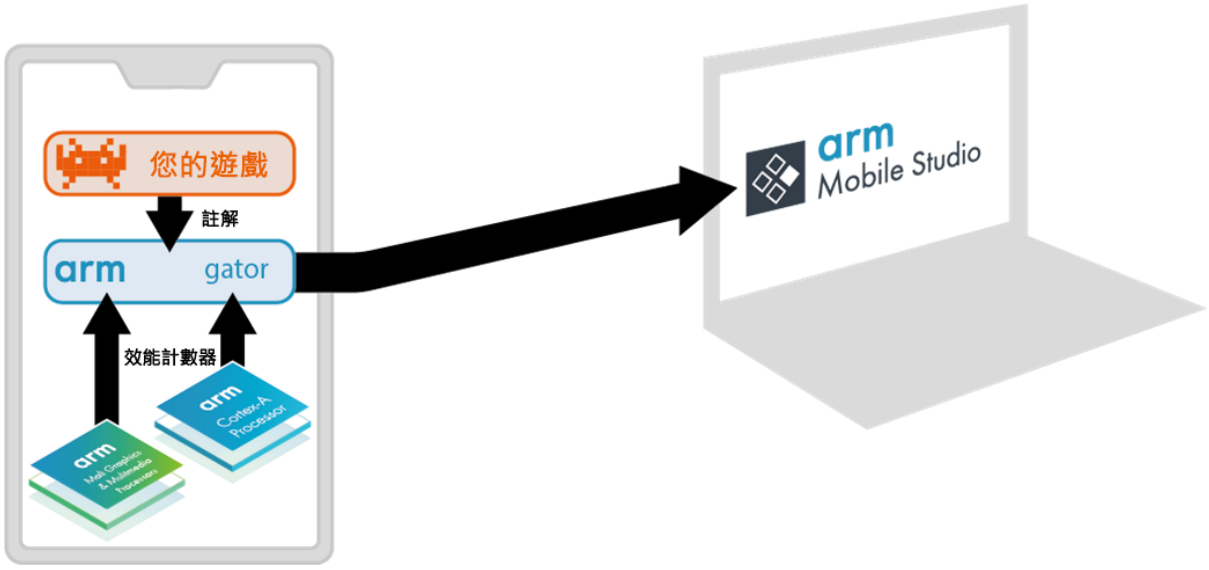
以上快速概要說明，如果使用遊戲本身的註解功能來提供一些更高階情境，就可以在 Streamline 中獲得更多的洞悉見解。我們使用了：

- 「標記」，以顯示新畫面的開始時間
- 「註解欄」，以顯示哪個場景正在執行，以及主要執行緒正在進行哪些活動。
- 「自訂活動圖」，以顯示在非同步排定地形產生作業的行為。

這些全都非常實用，那麼究竟是如何運作的呢？

Streamline 註解簡介

接下來讓我們更深入探究 Streamline 的運作方式。在分析某一 Android 應用程式時，另一個名為「gator」的流程 (執行時的使用者與應用程式相同) 會在裝置上執行，從不同的硬體來源 (例如 Mali GPU 與 Arm Cortex-A CPU) 收集剖析資訊，並將指標彙總資料流回傳到您的電腦。Streamline 註解是一種機制，讓應用程式本身能將其自己的標記與指標插入到上述資料流內。



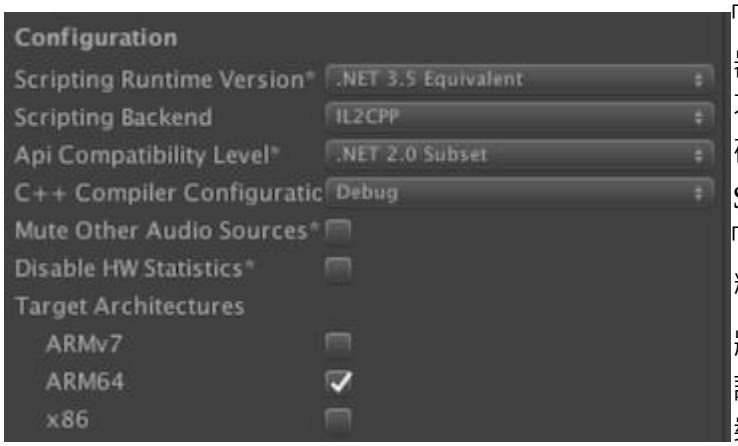
如何在 Unity 中使用 Streamline 註解

Streamline 註解使用特定的通訊協定，而開放式原始碼 C 語言執行功能則是隨附於 Arm Mobile Studio。為能輕易地從 Unity 內容中產生 Streamline 註解，您需要使用一些

C 語言實作相關 C# 包裝函式。本文逐步解說中所使用的包裝函式，以及必要的 C 語言實作功能以 Unity Asset Package 形式提供。下載此套件，並匯入到您的專案中，可作為自訂資產套件。此套件會將新方法加到 `Arm` 命名空間內，這些新方法讓您能在自己的專案中輕易地使用 Streamline 註解。API 說明文件可在此套件中的 README.md 檔案內找到。

設定 Unity 專案以獲得最佳使用經驗

如果您想以最快速且最容易的方式分析出自 Unity 的 Android 組建成果，應進行一些特定的 Android Player 設定：



請確定您是使用「IL2CPP」作為「指令碼後端」，並將「C++ 編譯器組態」設定為「偵錯」。此步驟不僅會將指令碼編譯為原生程式碼，進而實現更好的效能，更代表 Streamline 能看到偵錯資訊，並在「呼叫路徑」檢視畫面中將效能資料反向對應到您使用的函式。

將「目標架構」設定為 ARM64 (預設值為 ARMv7)。目前大多數行動裝置都是 64 位元，因此您將會獲得更高品質的程式碼產生體驗。

加上標記

標記是最容易使用的註解。提供的方法是使用一個字串和一個選用顏色。例如，用綠色為各個畫面做標記，在其中一個 GameObject 中使用下列程式碼 (如果您不熟悉 Unity 架構，在此說明 Update() 方法是每個畫面會被自動呼叫一次)。

全螢幕

加上註解欄

使用註解欄的方法也差不多一樣簡單。首先，必須建立一列註解欄，並指定其名稱。接下來，例如可以使用「註解欄」物件上的方法，將註解登錄到註解欄內：

全螢幕

切記，註解欄內的註解將會跨越一段時間。如果希望在開始下一個註解之前結束目前的註解，可以使用 `end()` 方法。例如，將「TerrainController」在主要執行緒中進行地形完成的部分包裝成以下函式：

全螢幕

使用自訂活動圖

「自訂活動圖」(CAM) 可以想成是註解欄頂端的另一層。您必須先為 CAM 命名之後，才能在其中建立軌跡。接著就能將註解加到這些軌跡上，方式將如同將註解加到註解欄一樣。

在此範例中，地形產生 CAM 是依照以下方式建立：

全螢幕

但是，此使用情境有一個枝節問題：當作業正在 Unity Job System 中執行時，根本無法與其餘遊戲物件互動 (此限制有助於維持物件執行緒安全)。我們在此作業中唯一能做的就是記住作業的開始與停止時間，然後當主要執行緒清除此作業時，將此作業的活動登錄到 CAM 中。

C# 包裝函式提供能從作業中安全叫出的函式，並會以 Streamline 註解需要的形式回傳目前時間。

全螢幕

一旦回到主要執行緒，就能挑選一條軌跡使用 (我們在一個集區內管理這些軌跡，以確保沒有重疊，這在視覺檢視上極有幫助)，並將作業登錄到此軌跡上。這裡顯示的 `job.timings` 是一個二項陣列，由各個作業填入開始與停止時間。

全螢幕

這部分的說明到此為止！我們將會隨著時間改良此 Unity Package，以增加更多功能；歡迎隨時給予指教！

在 Streamline 中收集基本剖析檔

您需要先完成幾個步驟，才能在 Streamline 中收集第一個剖析檔，但是一旦設定完成，後續操作就相當簡單。

首先，您需要下載並安裝適用於 Windows、Mac 或 Linux 的免費版本 Arm Mobile Studio Starter Edition。

稍早在說明 Streamline 的架構時曾提過，您需要先進行一些動作：

- 您需要在行動裝置上執行 gator，讓 gator 能存取您的應用程式。
- 您需要一個方法，將資料從裝置取出並傳到工具中。

雖然 Streamline 提供幾種達成此舉的方法，但是我們發現在各式各樣的裝置上都能穩健實施最簡單的方法，是先確保您知道一些關鍵資訊：

- 應用程式是 32 還是 64 位元。 如果您依照以上說明，並使用 ARM64 選項建置 Unity 遊戲，那麼您的應用程式將會是 64 位元。

- 應用程式的「套件名稱」(依照您在 Unity 的 Android Player 設定中指定的名稱)。本文範例應用程式的「套件名稱」是 `com.Arm.InfiniteTerrain`。
- 如何取得 gator 二進位檔：在您的 Arm Mobile Studio 安裝版本的 `streamline/bin/arm` (32 位元) 或 `streamline/bin/arm64` (64 位元) 資料夾中可找到這些二進位檔。

在知道上述關鍵資訊之後，進行分析所需步驟如下：

- 確定應用程式已安裝在您的裝置上。
- 使用 Android `adb` 工具，將行動裝置的網路連接埠轉接到執行 Arm Mobile Studio 之系統的本機連接埠。
- 使用 `adb` 將 gator (32 或 64 位元版本，取決於您的應用程式) 推送到您的裝置，然後讓 gator 以與您的應用程式相同的「套件名稱」開始執行。
- 啟動 Streamline，並將其與本機連接埠連接。此連接埠是您使用 adb 轉送資料流的目的地。這時您可以選擇有興趣收集的效能計數器。
- 在行動裝置上啟動應用程式。Streamline 就會在資料進入時開始收集分析資料。

只要 gator 開始執行後，不必重新啟動 gator，就能安裝新版本的應用程式、開始和停止 Streamline，並執行更多分析。

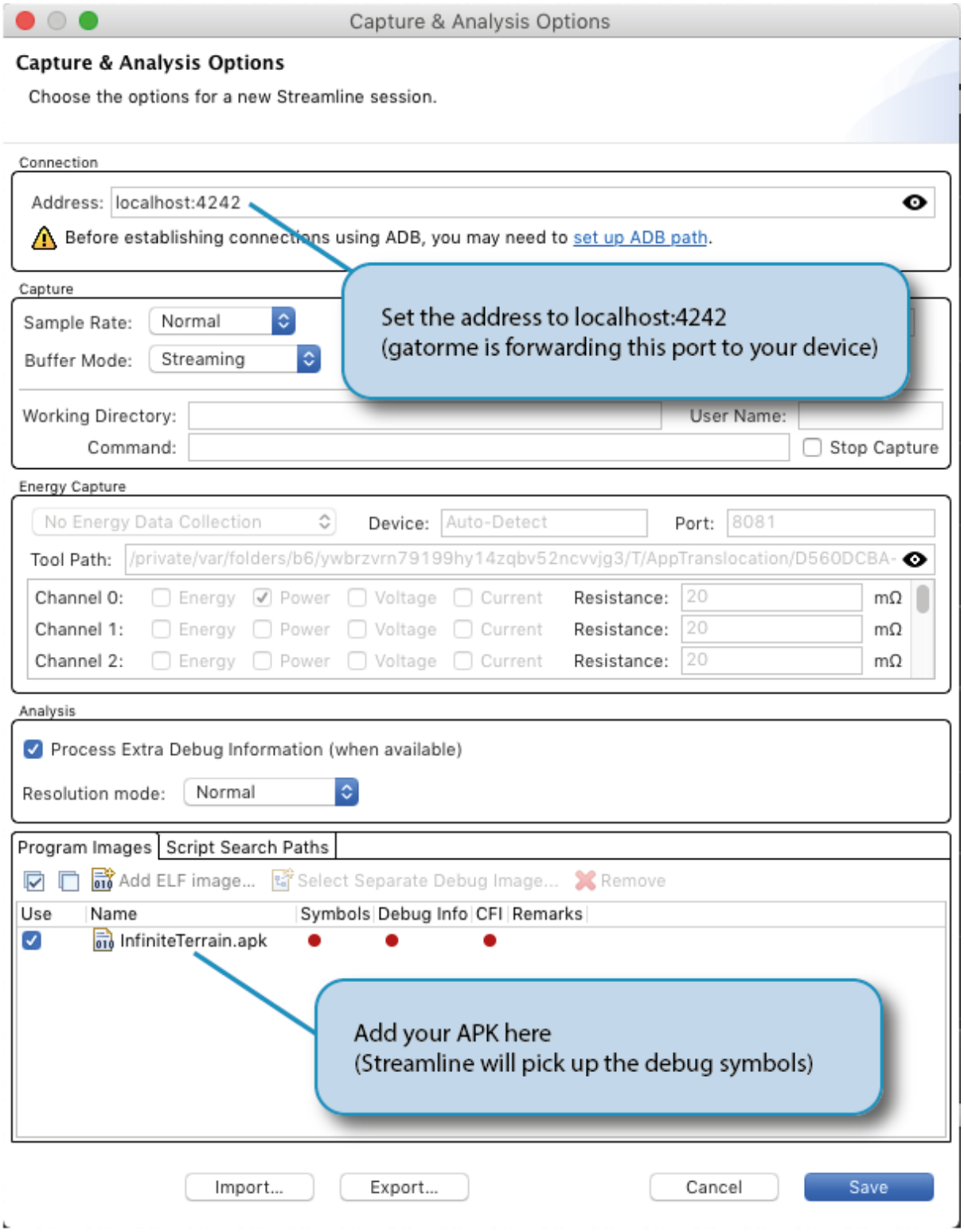
要讓上述流程更加容易，您可以首先下載 gatorme 批次檔，您可以用此批次檔為您設定並執行 `gator` 和 `adb`；您所需要提供的只有執行 gator 二進位檔所需要的路徑、應用程式的「套件名稱」，以及裝置中有何種 Mali GPU (如果 gator 在探查裝置後無法釐清是使用何種 GPU，Mali GPU 類型資訊能幫助釐清)。批次檔也會執行幾項其他步驟，以確保此方法在最廣泛多樣的行動裝置上都能順利執行，並確保在您完成剖析活動時，確實將 gator 關閉。(沒錯，我們在不久的將來會將 `gatorme` 功能直接納入 Streamline 內！)。

gatorme 說明文件提供詳細資訊，不過既然有現成的範例，以下將說明如何在 APK 安裝到裝置上之後，剖析 InfiniteTerrain 內容。

首先，從命令列執行 gatorme：

全螢幕

這時您就能啟動 Streamline 並準備好擷取資料。建議您進行兩項設定，以確定您的操作正確：



當您完成此設定之後，要重複進行部署/分析/修正步驟就很容易 – 在您關閉應用程式、直接從 Unity 建置並執行，以及擷取更多 Streamline 資訊時，可以讓 gatorme 繼續執行。

希望您覺得本文的說明既有趣又實用。InfiniteTerrain 範例所使用的所有原始碼都可以在 GitHub 上下載 (需要有 Apache 2.0 授權)。所有的原始碼和圖形資產也可下載，另外還有 ArmMobileStudio.unitypackage，您可以將此 Unity 自訂資產套件匯入到您自己的

專案中，以加入 Streamline 註解。還有預先建置的 InfiniteTerrain.apk，這個 64 位元的 Android 開發建置版本可立即部署到您的裝置，如果您只想快速開始分析某些內容，可以選用此版本。

最後，如果您對於 Arm Mobile Studio 或 Arm 在圖形與遊戲設計方面的一般應用有任何疑問，請加入我們的「圖形與多媒體論壇」(Graphics and Multimedia Forum)，或是在以下 Arm Mobile Studio 開發人員網站上閱讀更多的 Arm 工具相關說明！

Arm Mobile Studio 資源