光線追踪的一致性收集：硬體光線追踪的好處

雖然理論上，建置一顆新款 GPU 的方法有無限多種，但要讓晶片真正有效地運作，還是有許多考量的。製造先進高效能半導體的挑戰，以及如何解決可編程光柵化(rasterization)的眼前難題，是目前建置 GPU 硬體的主要產業趨勢。

舉例來說，SIMD 處理和固定功能紋理硬體是現代 GPU 中不可或缺的，甚至，我們可以說，如果不利用它們來實現 GPU，幾乎是意味著，此 GPU 是不會有商業市場的。即使是過去二十年來任何一款 GPU 的最瘋狂的願景，也都沒有放棄過這些核心原則。

過去 15 年來，即時光線追踪加速是 GPU 潛規則中的最大困擾。關於光線追蹤在 GPU 上如何工作的主要規範，即 Microsoft 的 DXR，要求的執行模型不能真正與 GPU 的工作方式融合在一起，因此，已為想要支援它的 GPU 設計人員帶來了嚴重的潛在麻煩。若過去十多年來，他們一直沒有仔細考慮即時光線追蹤，那麼情況更是如此。但在 Imagination，我們從未停止光線追蹤技術的開發。

光線追踪的關鍵挑戰

如果您透過 DXR 規範，並考慮需要在 GPU 中建置什麼才能提供有用的加速，那麼您將快速選出一些高階設計方案，而這些是任何最終設計都需要處理的。

首先，您需要一種方法來生成和處理一組包含該幾何繪圖的數據結構，以使您可以高效地針對該幾何繪圖來追蹤光線。其次，在追踪光線時，當 GPU 測試光線是否與物件相交時，可能會用到一些使用者定義的可編程特性。 第三，被追踪的光線會發射出新的光線！雖然 DXR 建置時，還需要考慮其他的事情，但是，從更大的格局來看，這三項考慮才是最重要的。

PowerVR 光線追踪的混合渲染功能

生成並使用加速結構以有效地表示需要被測試是否與光線相交的幾何形狀，意味著 GPU 可能需完成一個全新的執行階段，然後我們需要執行一種全新類型的工作負載來處理這些加速

結構，測試它們是否擊中物件，然後再根據結果，由程式人員來控制是否需執行哪些運算。GPU 採用平行處理，那麼一起處理一堆光線意味著什麼？ 這樣做是否發現了新的挑戰，而這些挑戰與傳統的幾何和畫素平行處理所帶來的挑戰，是否有明顯的不同呢？

最後一個問題的答案是肯定的，對於如何把光線追蹤映射到當代的 GPU 執行模型上，這些差異會帶來深遠的影響。由於 GPU 的運算和記憶體資源不平衡，使記憶體存取成為一種寶貴的資源，而浪費這些，是造成效率和效能降低的最主要原因。

傳統 GPU 執行光線追蹤的限制

GPU 的設計方式是要充份利用與其連接 DRAM 的存取，並利用記憶體存取的空間或時間局部性來實現此目的。值得慶幸的是，最常見和現代的光柵化渲染都具有很好的特性，亦即在著色(尤其是畫素著色，這通常是的主要工作負載)期間，三角形和畫素會與其相鄰畫素共享數據。因此，如果您存取任何的快取數據，而此數據是某一組畫素所需要的，那麼很可能下一個相鄰的畫素組會需要使用您已經從 DRAM 和快取中取得的部分或全部記憶體。對現今大多數的光柵化渲染工作負載來說，這是正確的，因此我們能確定地以此特性為基礎，來設計 GPU。

在我們談到光線追踪之前，這一切都很棒。但光線追踪不會用到空間局部性的特性，以下我們將說明原因。

表面問題

最簡單的思考方法是環顧四周，觀察一下，當你坐下和閱讀時，環境中的光線是如何運作的。由於光線追蹤模型是根據它在所有表面傳播的特性進行建模的，因此它必須處理光線擊中場景中任何表面發生的情況。也許我們只在乎光線撞擊到什麼，以及那是什麼。可能是表面以均勻方向把光線散射出去，但也有可能幾乎是完全隨機的；也許，表面吸收了所有的光線，便不再反射出去；也許，表面上有一種材料可以部分吸收所有光線，然後隨機散射出少部分它無法捕獲的光。

在這些情況中，只有第一種情況符合 GPU 利用記憶體局部性的運作方式。即使如此，也只有當所有擊中同類型三角形的光線是以平行方式處理時，才是符合的。

正是這種潛在的明顯分歧導致了問題。如果在一群平行處理的光線中，有任何一條光線的行為可以與其他的不同，例如，擊中加速結構中的不同部位、或發射出新的光線，那麼就破壞了 GPU 運作的基本模型，而且更嚴重的是，會比一般的幾何或畫素處理面臨更大的差異。

一致性收集

與目前市場上的其他硬體光線追蹤加速技術相比，PowerVR 的光線追蹤硬體加速是非常獨特的。它是利用硬體光線追蹤和分類，並對軟體是通透，以確保當硬體執行計算時，平行分配的光線確實有類似的基本特性。我們將此稱為一致性收集(coherency gathering)。業界的其他光線追蹤解決方案已在軟體方面邁出了關鍵的一步，但這將不可避免地變得更慢、效率更低。

利用硬體來維護由軟體發出的飛行光線資料庫，因此能夠根據光線的方向，按照它們在加速結構中前進的位置進行選擇和分組。 這意味著，在處理它們時，它們更能夠共享從記憶體中已存取的加速結構數據。此外，這樣做還有另一個好處，能夠最大化 GPU 之後處理的平行光線─幾何相交數量。

透過分析由硬體排序的飛行中光線，我們可確保，能以更有效的處理方式將它們分組，並且是符合 GPU 運算方式的。這是系統成功的關鍵，能夠不打破 GPU 產業精心建構的執行模型，並同時建立高效的光柵器。這樣就不需要專用於光線追蹤硬體的特殊記憶體系統，因此能夠更輕鬆地與其他 GPU 元件整合。

複雜的一致性光線機制

一致性收集的機制本身非常複雜，因為它需要快速追蹤、分類和發送系統中的所有飛行光線，而不會給饋入它的調度程式系統造成壓力，或耗盡測試硬體資源，此硬體是用來比對幾何加速結構，以處理已經分類的光線。

如果沒有適當的硬體系統來幫助 GPU 處理相似的光線，您就得依賴應用程式或遊戲開發人員以某種方式處理主機上的光線一致性，或是，爭取在 GPU 上利用運算程式來處理一部分的分類 ─ 如果您在硬體中處理光線的方式，是一開始就允許這樣做的話。這些選項都無法在即時系統中提高效能和效率，但 Imagination 是市場上唯一具有這種硬體光線追蹤系統的 GPU 供應商。

充分利用光線流

我們之所以是唯一一家提供硬體光線追蹤技術的業者，是因為我們長久以來一直致力於解決此問題，相較之下，其他業者還在採用較不成熟的方式。但現在，光線追蹤已成為主要繪圖 API 的重要功能了。

我們的一致性收集能與現今的光線追蹤視圖相容，(在此情況下，如果光線恰好會發出新的光線，那麼堆疊也會解開，也可能恰好會發出新的光線，依此類推)，能夠在每個發送步驟進行一致性收集，並確保盡可能接近硬體的可能光線流。

在新型的硬體光線追踪器中，最重要的是測量光線流(ray flow)。峰值平行測試速率，或空光線發射(empty ray launch)和失效率(miss rate)，是描述光線追蹤硬體效能的簡單方式，但並不是很有用。畢竟，開發人員不是只在乎較高的峰值平行測試速率，或較高的失效率。

我們的目標是在整個加速系統中充分使用光線流，使開發人員能在可用的光源預算之內，發揮最大的效益。我們的一致性收集系統使我們能夠提供這一點，因此與現今市場上的任何其他系統相比，它都具有獨特性。