0 A.D. のフレームレートのボトルネックを特定

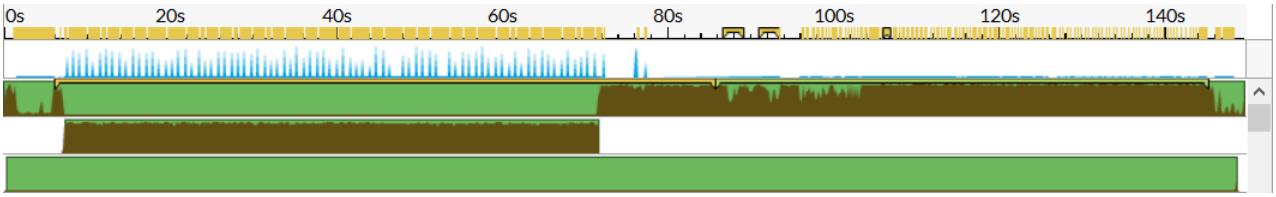
この記事は、インテル® デベロッパー・ゾーンに公開されている「Identifying the Frame Rate Bottleneck in 0 A.D.*」の日本語参考訳です。

0 A.D. (英語) は、2003 年以来 Wildfire Games により開発が継続されている無料のオープンソースのリアルタイム戦略ゲームで、1997 年にリリースされた Age of Empires* に似ています。ゲームとそのカスタムメイドの Pyrogenesis エンジンは、C++ と Javascript* を使用して記述されています。ゼロから構築されている 0 A.D. は、既存のゲームエンジンに関連する通常のライセンス要件はありませんが、最適化が行われていないため、パフォーマンスの問題があります。長いロード時間に加えて、数百単位のマップにはフレームレートのボトルネックがあり、1 秒あたり 4 フレームまで低下します。インテルは 0 A.D. 開発コミュニティーからの支援を受けて、パフォーマンス・プロファイリング・ツールであるインテル® VTune™ Amplifier を使用してこの問題の原因の特定に取り組みました。

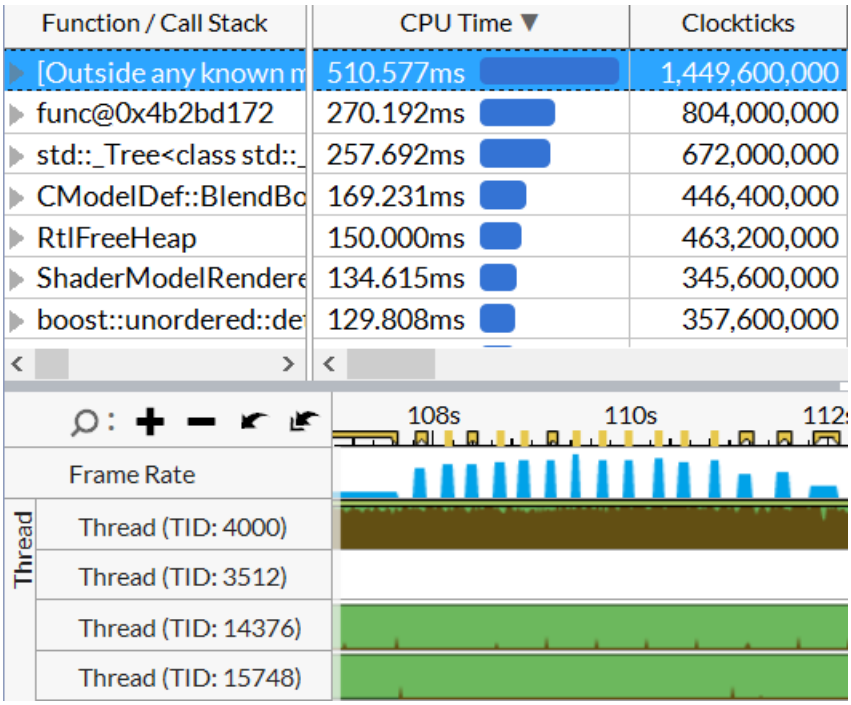
インテルの分析

最初のステップは、インテル® VTune™ Amplifier のタイムラインにフレーム、画面のロードの開始と終了を示すイベント領域、実際のゲーム領域を示すマーカーが適切に表示されるようにすることでした。ポーズと再開アノテーションを使用して対象領域外でデータが収集されないようにすることも可能ですが、ここではその方法を採用しませんでした。フレームとイベントのアノテーションを含めてゲームをコンパイルした後、マイクロアーキテクチャー特性解析に着手しました (開始点としてホットスポット解析も同様に適切です)。

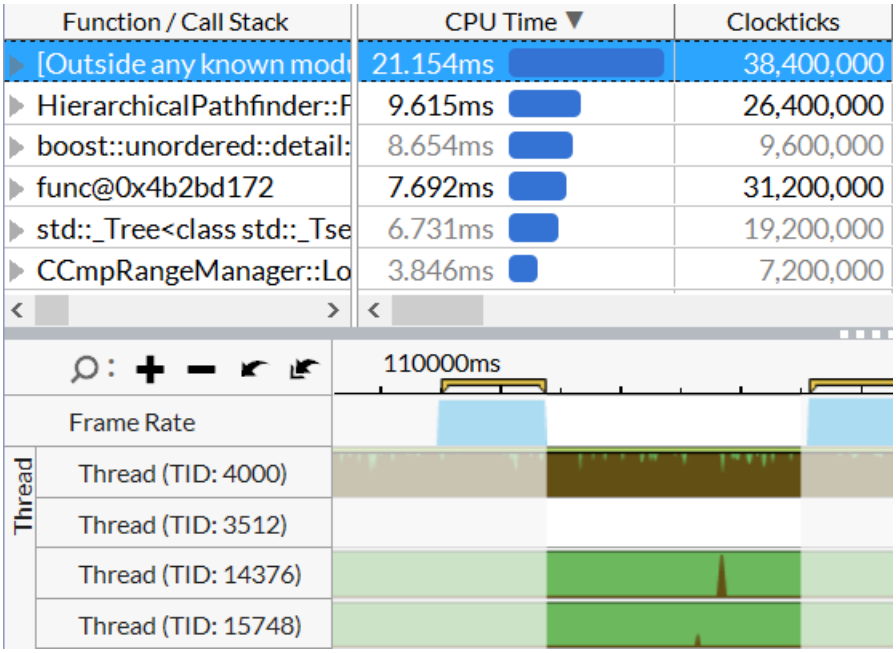
アノテーションは、インテル® VTune™ Amplifier の結果のタイムラインにマークを示します。最上部の黄色のブロックはフレームで、その下の青色のグラフはフレームレートです。その下の各バーはスレッドに関する情報で、実行中のスレッドは緑色で、CPU 時間は茶色で示されています。1 番上のスレッドバーには、バーの上部に 2 つの長いブラケットがあります。これらはイベント領域です。左の黄色のブラケットは画面のロードで、右の緑色のブラケットはゲームプレイです。



このゲームの使用パターンの特徴は、ロード時間が長く、ゲーム開始時にいくつかの非常に遅いフレームがあり、ゲームプレイの残りの部分は比較的安定したフレームパターンを示しています。ゲームプレイの途中の代表的なセクションを拡大してフィルター処理すると、ギャップの大きな短いフレームのパターンが顕著になるだけでなく、非常に多くのゲーム時間が [Outside any known module] (既知のモジュール外) で費やされていることが分かります。

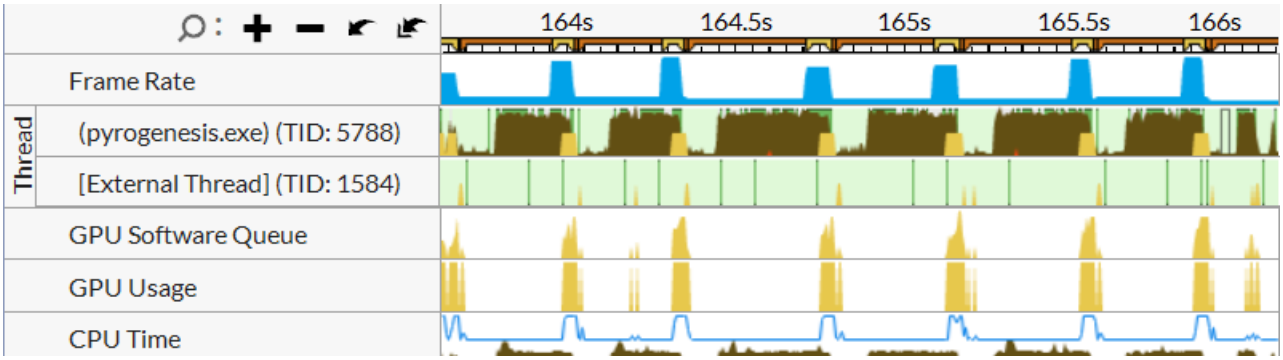


フレーム自体は非常に短いため、ギャップの 1 つにフィルターインして、この遅延に関連するデータのみを表示します。多くのゲームプレイ関連の関数が除外され、不明なモジュールデータと下位レベルの関数に多くの時間が費やされていることが分かります。

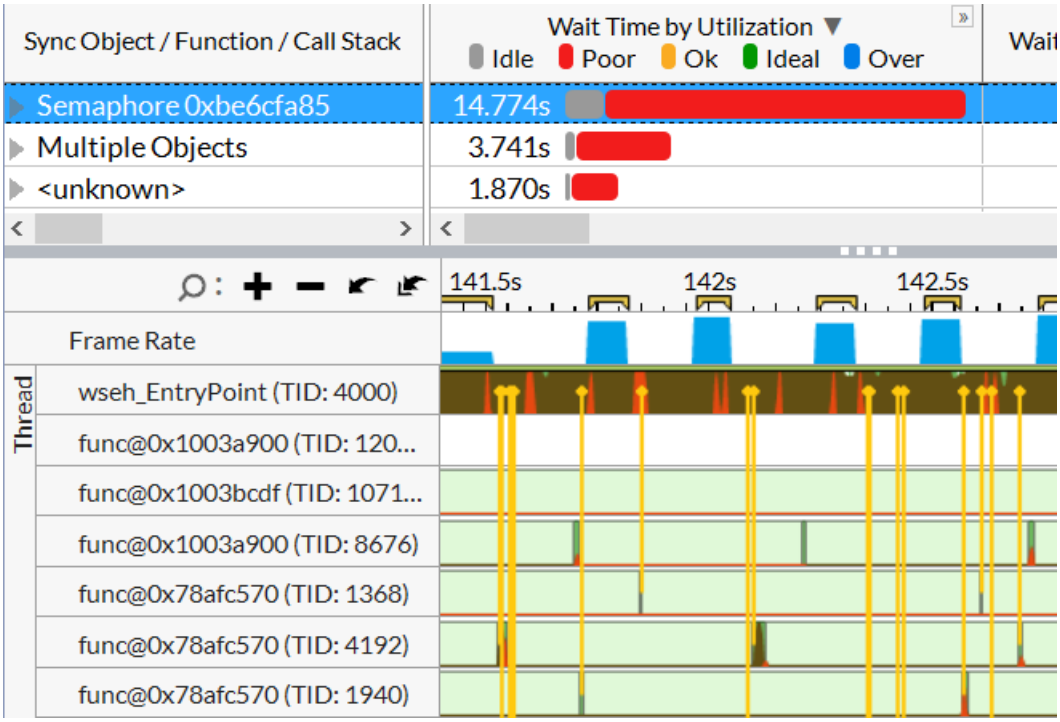


さらに深く掘り下げる

フレーム間の遅延のアクティビティーの大分部は Pyrogenesis コード外のものであるため、CPU/GPU 並行性解析を実行して、これらの期間中に GPU がアクティブであったかどうか確認しました。しかし、GPU も同じ断続的なパターンを示しており、ゲームは GPU を待機していませんでした。



次に、アクティビティーが非常に少ないスレッドが多くあるため、スレッド解析を実行しました。案の定、フレーム間で多くのスレッド遷移が見つかりました (図では黄色で示されています)。費やされた時間の大部分は、2 つの同期オブジェクトで占められています。



コールスタックを追跡することで、最終的に BaseThreadInitThunk にたどり着きました。サンクは、C++ と Javascript* など、2 種類のコード間のインターフェイスです。

まとめ

フレームギャップのサンクに関連する多数のギャップは、結果全体をとおして出現する さまざまな "js" エントリーと相まって、ゲームのボトルネックが Javascript*であることを強く示唆しています。問題の本質は未確認のままです。過剰なガベージコレクションや頻繁なインターフェイスである可能性もありますが、おそらくゲームロジックの大部分を実行する Javascript*が原因であり、C++ のほうがより効率良く記述できる可能性が高いでしょう。

インテルは Wildfire Games に調査結果を報告しましたが、残念ながら、Javascript* の更新は膨大な作業であるため、この記事の執筆時点では、このボトルネックは未対応のままです。

コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。