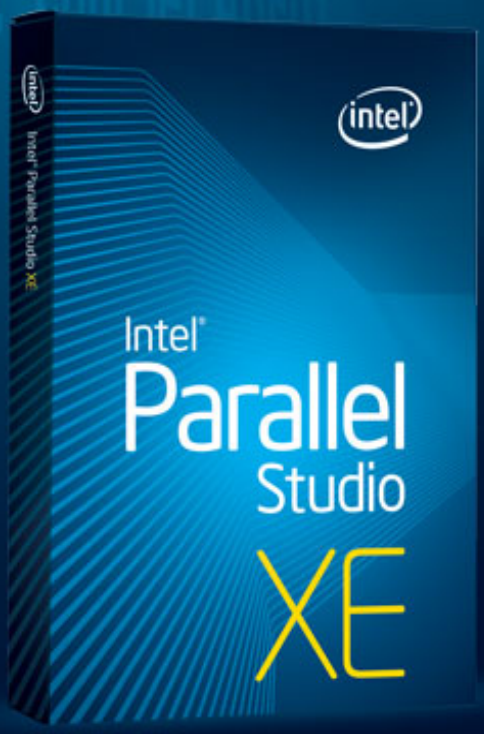
リソースリークを解決して安定性を向上



リソースリークとは、リソース消費の 1 つで、プログラムが割り当てたリソースを解放していない状態を指します。通常、一般的なリソース問題 (メモリーリークなど)やコードの不具合による結果として起こりますが、非常に限られた状況や、長い間アプリケーションを使用した場合に問題を引き起こします。

インテル® Inspector XE は、直列および並列アプリケーションをサポートし、メモリー/スレッドエラー検証機能の両方を備えた動的分析ツールです。1 回の発生でリソースリークを検出するため、アプリケーションで不具合が再現できない場合でも、エラーの場所を特定できます。インテル® Inspector XE を実行してアプリケーションをチェックすることで、リソースリークをピンポイント検出し修正して、問題を未然に防ぐことができます。

はじめに

スレッドエラーとメモリーエラーを検出し解決した後にも、プログラムで原因の分からない不具合が断続的に発生する場合は、リソースリークが発生している可能性があります。リソースリークのほとんどは大きな問題ではありません。リソースにハンドルを割り当ててプログラムが終了するまで保持するようにすると、プログラムの終了時にそのリソースは自動で解放されます。

通常はこの方法で問題ありませんが、ファイルやハッチブラシなど、使用するリソースの数が増えるに従って問題となります。GDI ブラシなど、リソースの種類によっては利用が制限されているものもありますが、すべてのリソースはある程度のシステムメモリーを消費します。その結果、システムのパフォーマンスが低下したり、予期しないシステム API 障害が発生することがあります。

インテル® Inspector XE は、26 種類のリソースを追跡し、コード内で割り当てられているリソースのうち解放されていないものを特定する、シングルおよびマルチスレッドのエラー検証ツールです。Linux* 版と Microsoft* Visual Studio* に統合可能な Windows* 版があります。C/C++、C#、.NET、および Fortran で開発されたアプリケーションをサポートします。この統合開発支援ツールは、エラーを正確に示し、アプリケーションの信頼性と品質を保証するためのガイダンスを提供します。

本ガイドは、インテル® Inspector XE を使用してプログラム中のリソースリークを検出、修正して、問題を未然に防ぐ方法について説明します。

手順:リソースエラーの特定、分析、解決

インテル® Inspector XE を使って、一連のステップを実行することで、直列または並列プログラムのリソースエラーを特定、分析、解決できます。このチュートリアルでは、「Colors」という名前のサンプルプログラムを使用して、順に説明します。

注: インテル® Inspector XE は Microsoft* Visual Studio* 2008、2010、2012 に統合されます。このチュートリアルでは、Microsoft* Visual* Studio* 2010 開発環境 (IDE) を使用しています。メニュー項目の異なる IDE を使用する場合はご注意ください。

ステップ 1: インテル® Parallel Studio XE のインストールと設定

推定所要時間: 15 ～ 30 分

1. インテル® Parallel Studio XE の評価版をダウンロードします。
2. **parallel_studio_xe_2013_setup.exe** をダブルクリックしてインテル® Parallel Studio XE をインストールします (システムにより異なりますが、約 15 ～ 30 分かかります)。

Colors サンプル・アプリケーションのインストール:

1. Colors_conf.zip サンプルファイルをローカルマシンにダウンロードします。このサンプルは、Microsoft* Visual Studio* 2010 を使用して作成された C++ GUI アプリケーションです。
2. **Colors_conf.zip** ファイルをシステムの書き込み可能なフォルダーに展開します。

サンプル・アプリケーションの実行

1. Microsoft* Visual Studio* でサンプルを開きます。**[ファイル] > [開く] > [プロジェクト/ソリューション]** を選択して、**colors_conf\vc10\colors.sln ソリューション・ファイル**を開きます (図 1 を参照)。

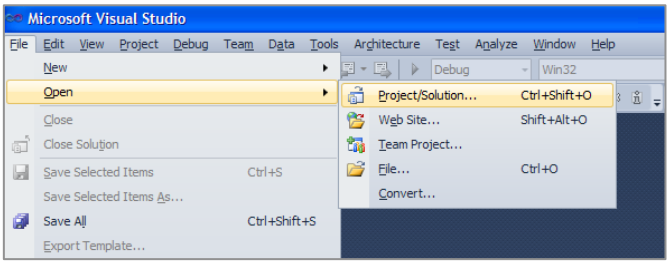


図 1

[ソリューション エクスプローラ] に Colors ソリューションが表示されます (図 2 を参照)。

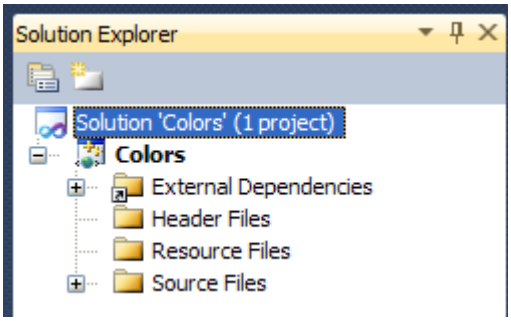


図 2

3. **[ビルド] > [ソリューションのビルド]** を選択して、アプリケーションをビルドします (図 3 を参照)。

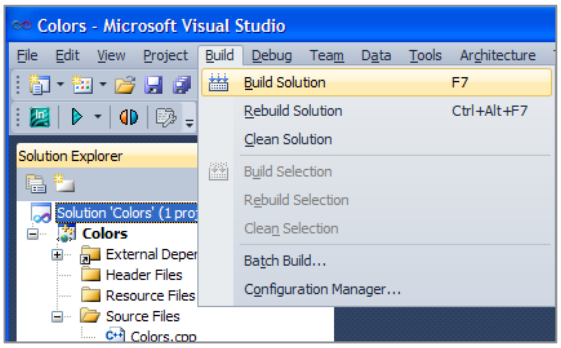


図 3

4. [デバッグ] > [デバッグなしで開始] でアプリケーションを実行します (図 4 を参照)。

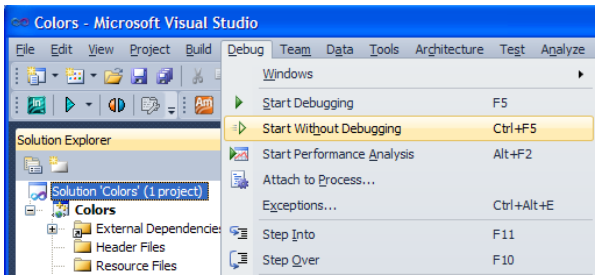


図 4

図 5 のようにアプリケーションが表示されます。

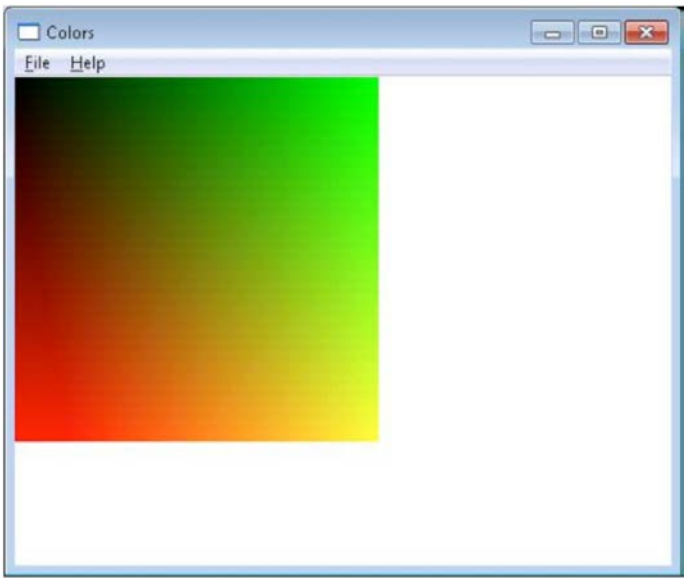


図 5

ウィンドウのサイズを 2 回変更してみてください。1 回目のサイズ変更後は正しく表示されますが、2 回目のサイズ変更後は 図 6 のように正しく表示されません。

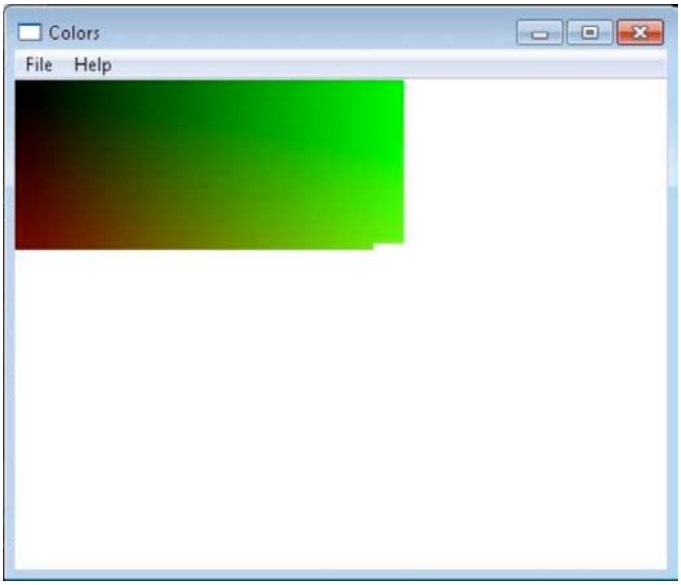


図 6

もう 1 度サイズを変更すると、カラーの部分が全く表示されず、ウィンドウのコントロール・ボタンも正しく描画されません。この問題は、プログラム中のリソースリークにより発生します。

分析の設定と実行

メモリーエラー分析の範囲と実行時間に合ったプリセット設定を選択します。

メモリーエラー分析を設定するには:

1. Microsoft* Visual Studio* メニューから **[ツール] > [Intel Inspector XE 2013 (インテル(R) Inspector XE 2013)] > [New Analysis (新しい分析)]** を選択して、**[Analysis type (分析タイプ)]** ウィンドウを表示します。
2. **[Detect Leaks (リークの検出)]** 分析タイプを選択して、次のようなウィンドウを表示します (図 7)。

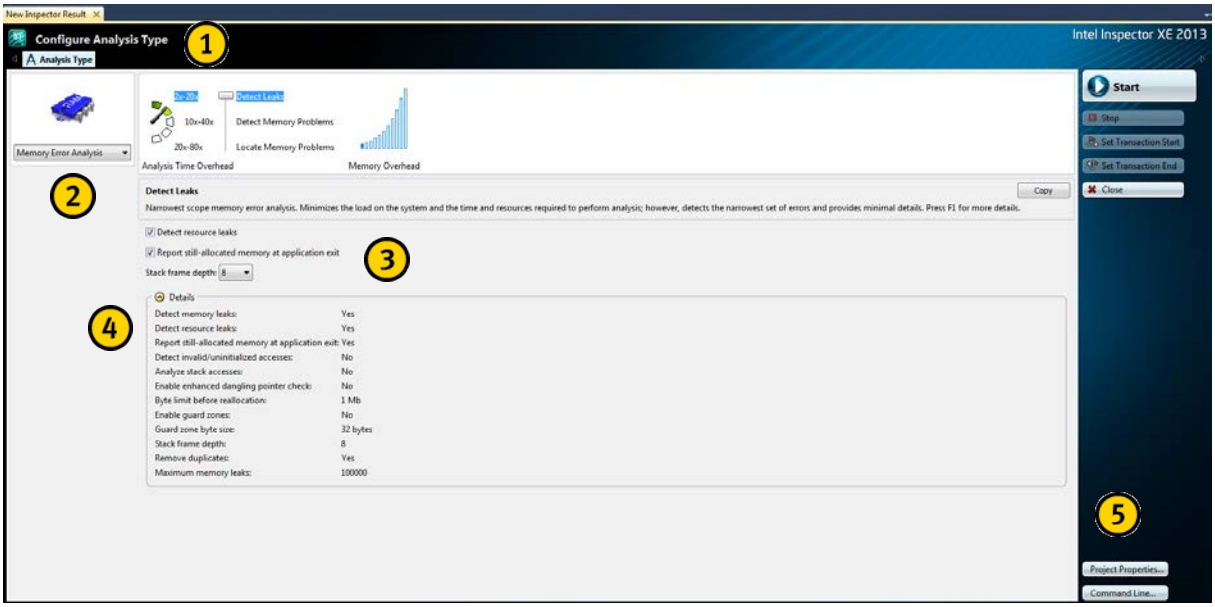


図 7

1. ナビゲーション・ツールバーで、インテル® Inspector XE のウィンドウを切り替えることができます。ツールバーに表示されるボタンは、表示されているウィンドウにより異なります。

2. 分析タイプツリーには、利用可能なプリセットの分析タイプが表示されます。
このチュートリアルでは、メモリーエラー分析を説明します。メモリーエラー分析は、GDI リソースリーク、不正な memcpy 呼び出し、無効な割り当て解除、カーネル・リソース・リーク、不正なメモリーアクセス、不正な部分メモリーアクセス、メモリーリーク、割り当てと解除の不一致、不明な割り当て、初期化されていないメモリーアクセス、初期化されていない部分メモリーアクセスの検出に使用できます。
スレッドエラー分析では、次のようなエラーを検出することができます:
データ競合、デッドロック、ロック階層違反、クロススレッド・スタック・アクセスの検出。
また、**[New (新規)]** ボタンをクリックして既存の分析タイプからカスタム分析タイプを作成することもできます。

3. チェックボックスとドロップダウン・リストを使用して、一部の分析タイプ設定を調整できます。より細かい設定が必要な場合は、別のプリセット分析タイプを選択するか、カスタム分析タイプを作成します。

4. [Details (詳細)] 領域には、現在の分析タイプのすべての設定が表示されます。異なる分析タイプを選択したり、チェックボックス/ドロップダウン・リストの値を変更して、[Details (詳細)] 領域の表示内容がどのように変化するのか試してみてください。

5. コマンドツールバーを使用して、分析の実行を制御したり、ほかの機能を実行することができます。例えば、[Project Properties (プロジェクト・プロパティー)] ボタンをクリックして [Project Properties (プロジェクト・プロパティー)] ダイアログボックスを表示し、デフォルトの結果ディレクトリーの場所を変更したり、分析を高速に行うためのパラメーターを設定したり、その他のプロジェクト設定を行うことができます。

3. 最後に、[Start (開始)] ボタンをクリックして、アプリケーションの分析を開始します。図 8 と同じ結果が表示された後、先程と同様に、ウィンドウのサイズを何回か変更して正しく表示されない問題を発生させます。その後アプリケーションを閉じてください。インテル® Inspector XE によって最終分析が行われます。この処理にはしばらく時間がかかります。分析中は、Microsoft* Visual Studio* IDE に 図 8 のようなウィンドウが表示されます。

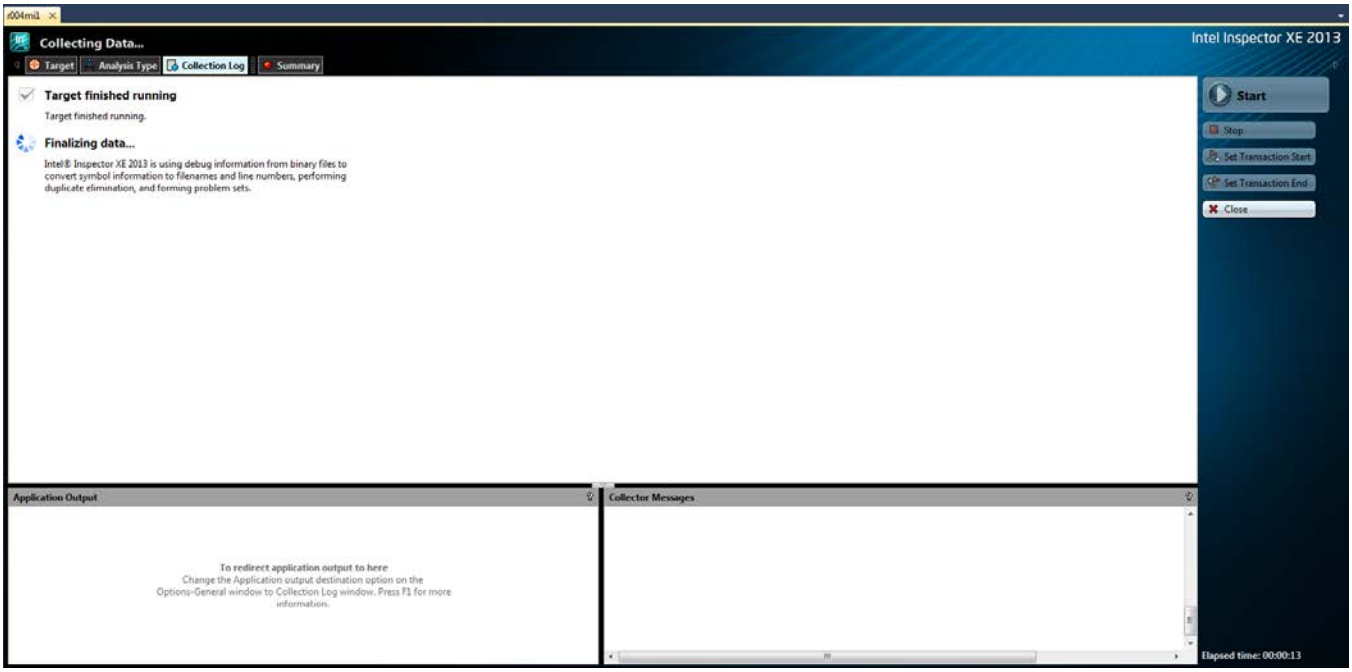


図 8

インテル® Inspector XE の分析が終了すると、図 9 のようなウィンドウが表示されます。

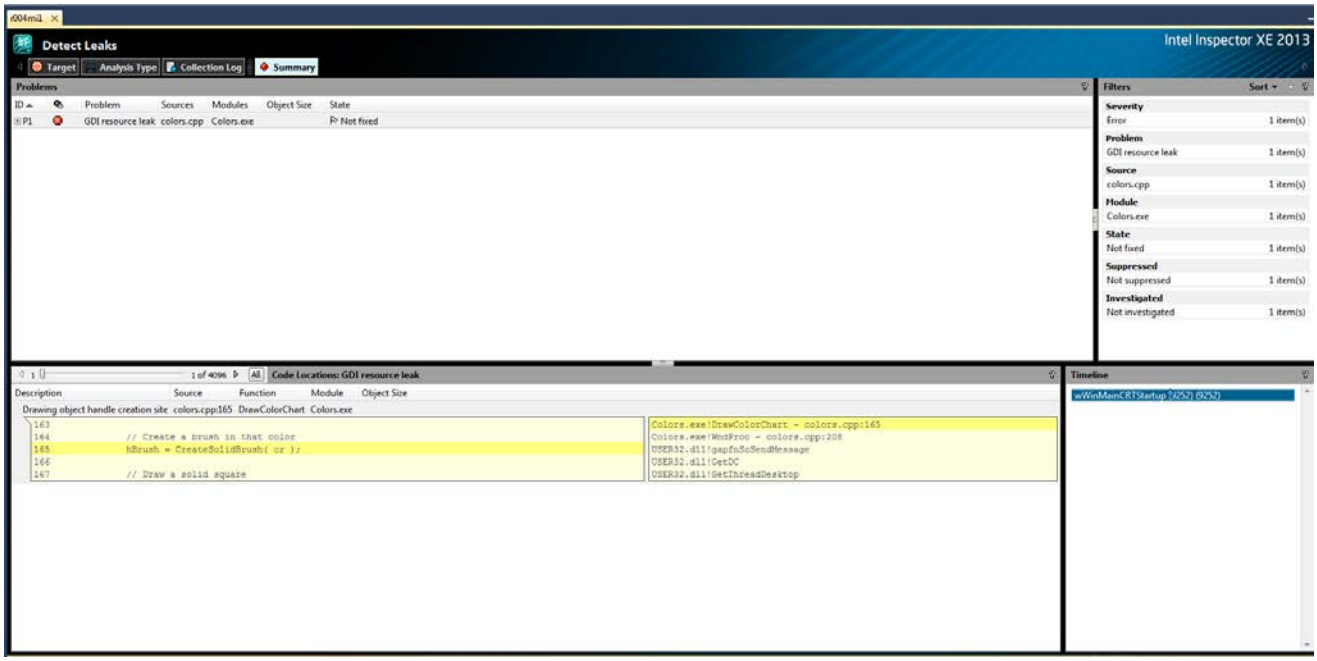


図 9

結果の解釈

図 9 のように、GDI リソースリークが 1 件表示されます。下のペインから、これは colors.cpp ファイルの 156 行目にある DrawColorChart 関数の描画オブジェクト・ハンドルのリークであることが分かります。[Problems (問題)] リストでこの問題をダブルクリックすると、[Sources (ソース)] ビューが表示されます。

このビューでは、問題のハンドルが作成されたソースコードの場所と、エラーを引き起こしたコールスタックを確認できます (図 10 を参照)。

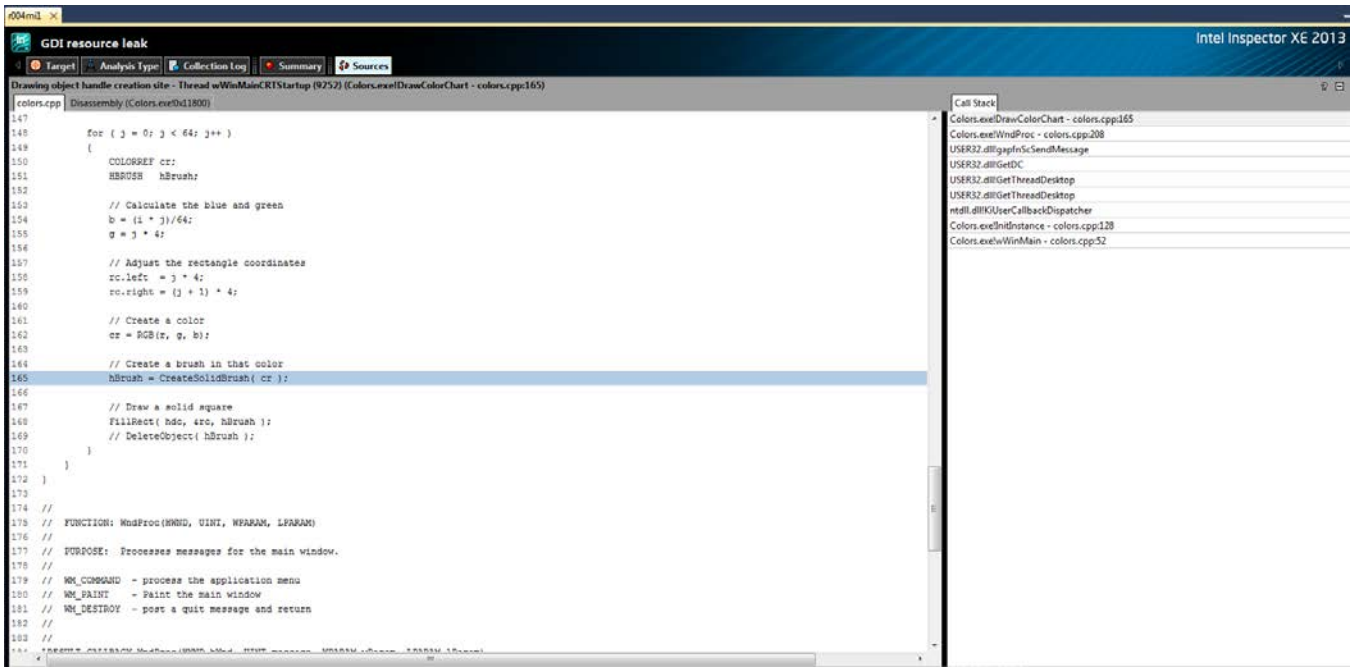


図 10

ソースコードから、GDI ブラシが 165 行目で作成され、このブラシが解放されていないことが分かります。

リソースリークのレポートでは、プログラムでリソースのハンドルが保持され解放されていない場合と、プログラムがすべてのハンドルを失った場合が区別されていません。問題を修正する前に、ソースコードを確認して、割り当てが行われているリソースの存続期間を決定する必要があります。

この例では、問題のハンドルは作成されたスコープの範囲外では使用されないため、使用後直ちに削除することができます。リソースを解放するための適切な関数が分からない場合は、左下のペインでコードを右クリックして **[Explain Problem (問題の説明)]** を選択すると、そのリソースを解放するための関数を含むリソースリークの説明が表示されます (図 11 を参照)。

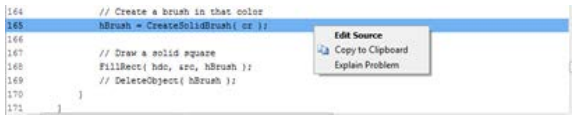


図 11

問題を修正する場合は、[Sources (ソース)] ビューで修正する行をダブルクリックすると、ソースファイルが開いて選択した行を編集できます。サンプル・アプリケーションで見つかった問題を解決するには、`FillRect()` の呼び出しの下に次のコードを追加します。

```
DeleteObject( hBrush );
```

そして、ソリューションをリビルドして再度実行してみます。ウィンドウのサイズを繰り返し変更しても、問題が発生しないことを確認できるでしょう。

結果

この例では、簡単に再現できる明確なリソースリークのエラーについて説明しました。しかし、リソースリークは常にこのように明確であるとは限りません。非常に特殊な状況下でのみ発生したり、長期間にわたってアプリケーションを使用した後に見つかることはよくあることです。

インテル® Inspector XE は、1 回の発生でリソースリークを検出します。そのため、アプリケーションで不具合が再現できない場合でも、エラーの場所を特定します。インテル® Inspector XE を定期的に実行してアプリケーションをチェックすることで、開発サイクルの早期にリソースリークのエラーを発見し、修正することが可能です。

大規模なアプリケーション/複雑なアプリケーションの場合のヒント

重要な概念: 小さく代表的なデータセットを選択する

分析を実行するとき、インテル® Inspector XE はデータセットに応じてターゲットを実行します。データセットのサイズはターゲットの実行時間と分析速度に直接影響します。

例えば、1000x1000 ピクセルのイメージのほうが、100x100 ピクセルのイメージよりも処理は長くなります。大きなイメージではループで 1...1000 の反復空間が必要になるのに対して、小さなイメージでは 1...100 でかまわないことも理由の 1 つです。完全に同じコードパスを両方のケースで実行します。違いは、これらのコードパスを繰り返す回数だけです。

ターゲットから冗長な処理を省くことで、完全性を損なうことなく、分析時間を制御できます。大きな繰り返し型のデータセットの代わりに、小さく典型的なデータセットを選択してください。数秒で実行できるデータセットが理想的です。追加のデータセットは簡単に作成することができます。

inspxe-cl のヘルプを表示するには、-help コマンドライン・オプションを使用します。

スレッドエラーの管理

インテル® Inspector XE は、並列プログラムの隠れたデータ競合やデッドロックのようなスレッドエラーも特定、分析、解決します。発見、再現、修正が非常に困難な、再現性がなく、異なる結果となるようなエラーも検出できます。

コマンドラインを使用したテストの自動化

インテル® Inspector XE は、エラーを検出するためにコードパスを実行する必要があるため、異なるコードパスや異なるワークロードを考慮して、コードを複数回実行することになります。このため、コード検査ツールが十分な時間をかけてテストできるように、これらのテストを一晩中、あるいはリグレッション・テストの一部として実行し、コンピューターに作業させるほうがより効率的です。翌朝に複数のテストの結果を確認するだけで済みます。

インテル Inspector XE のコマンドライン・バージョン (inspxe-cl) は、コマンドウィンドウから使用します ([スタート] > [ファイル名を指定して実行...] を選択し、"cmd" と入力して [OK] をクリックします。そして、インテル® Inspector XE のインストール・ディレクトリーに移動します)。(図 12)

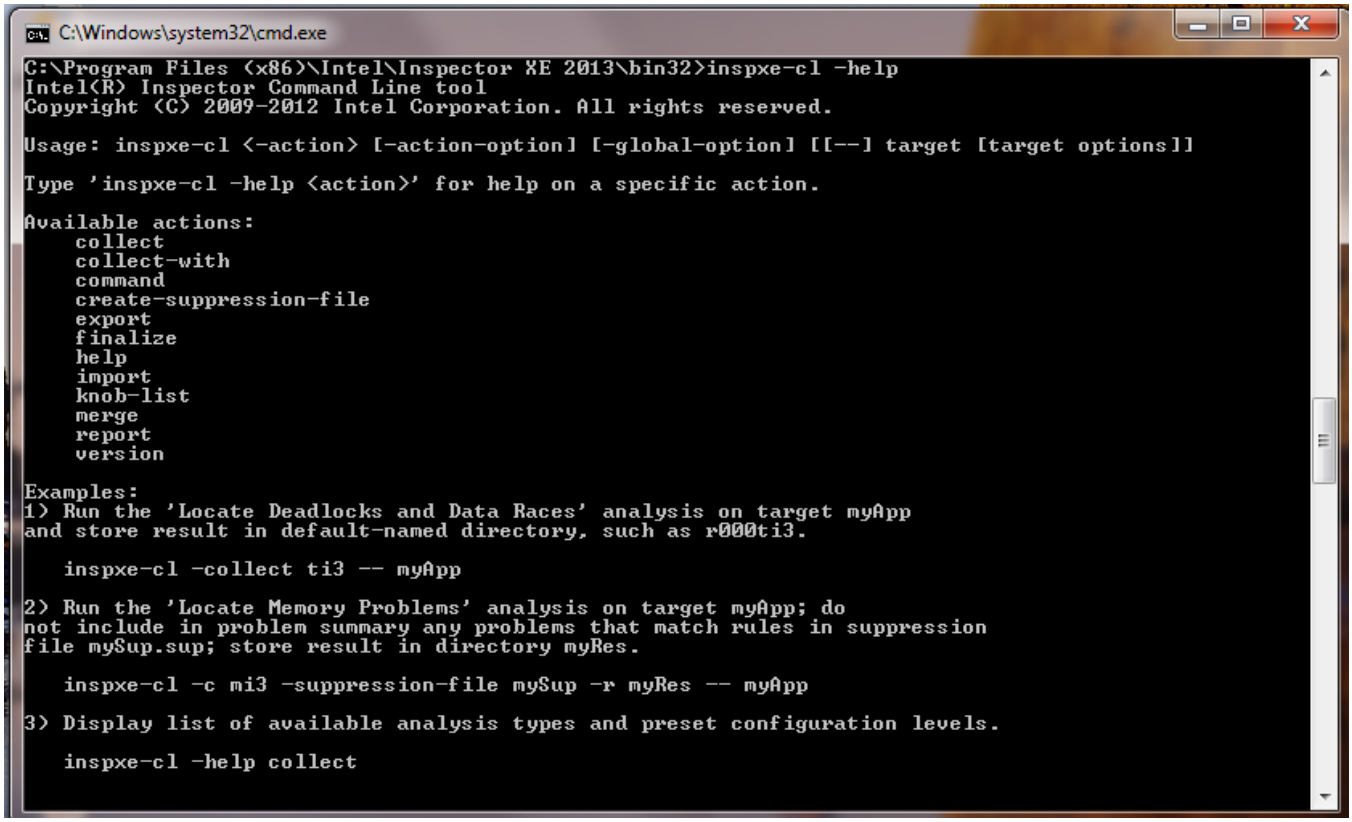


図 12

関連情報

ラーニング・ラボ – テクニカルビデオ、ホワイトペーパー、Webinar の再生など

インテル® Parallel Studio XE 製品ページ – HOW TO ビデオ、入門ガイド、ドキュメント、製品の詳細情報、サポートなど

評価ガイド – さまざまな機能の使用法を紹介する評価ガイド

インテル® ソフトウェア・ネットワーク・フォーラム – デベロッパー・コミュニティー

インテル® ソフトウェア製品ナレッジベース – インテル® ソフトウェア製品ナレッジベース

30 日間の評価版のダウンロード

購入方法: 言語別のスイート

アプリケーションをビルド、検証、チューニングする複数のツールが組み合わされた次のスイートがご利用になれます。本資料で説明している製品は青でハイライトされています。ライセンスは、シングルユーザー・ライセンス、フローティング・ライセンス、アカデミック・ライセンスが用意されています。

	スイート>>	インテル® Cluster Studio XE	インテル® Parallel Studio XE	インテル® C++ Studio XE	インテル® Fortran Studio XE	インテル® Composer XE	インテル® C++ Composer XE	インテル® Fortran Composer XE
コンポーネント	インテル® C/C++ コンパイラー	●	●	●		●	●	
	インテル® Fortran コンパイラー	●	●		●	●		●
	インテル® IPP[3]	●	●	●		●	●	
	インテル® MKL[3]	●	●	●	●	●	●	●
	インテル® Cilk™ Plus	●	●	●		●	●	
	インテル® TBB	●	●	●		●	●	
	インテル® Inspector XE	●	●	●	●			
	インテル® VTune™ Amplifier XE	●	●	●	●			
	インテル® Advisor XE	●	●	●	●			
	スタティック解析	●	●	●	●			
	インテル® MPI ライブラリー	●						
	インテル® Trace Analyzer & Collector	●						
	Rogue Wave IMSL* ライブラリー[2]							●
	オペレーティング・システム[1]	W、L	W、L	W、L	W、L	W、L	W、L、O	W、L、O

注:
[1] オペレーティング・システム: W = Windows*、L = Linux*、O = OS X*
[2] インテル® Visual Fortran Composer XE Windows* 版 IMSL* 同梱で利用可能
[3] OS X* 版は個別に提供されません。インテル® C++/Fortran Composer XE OS X* 版に含まれています。

インテル® Parallel Studio XE の詳細:

- 以下の Web サイトをご覧ください。 http://intel.ly/parallel-studio-xe
- あるいは、左の QR コードをスキャンしてください。

30 日間の評価版:

- http://intel.ly/sw-tools-eval の Web サイトで、「Product Suites」をクリックしてください。

注意事項

本資料に掲載されている情報は、インテル製品の概要説明を目的としたものです。本資料は、明示されているか否かにかかわらず、また禁反言によるとよらずにかかわらず、いかなる知的財産権のライセンスも許諾するものではありません。製品に付属の売買契約書『Intel's Terms and Conditions of Sale』に規定されている場合を除き、インテルはいかなる責任を負うものではなく、またインテル製品の販売や使用に関する明示または黙示の保証 (特定目的への適合性、商品適格性、あらゆる特許権、著作権、その他知的財産権の非侵害性への保証を含む) に関してもいかなる責任も負いません。

最適化に関する注意事項 改訂 #20110804

インテル® コンパイラーは、互換マイクロプロセッサー向けには、インテル製マイクロプロセッサー向けと同等レベルの最適化が行われない可能性があります。これには、インテル® ストリーミング SIMD 拡張命令 2 (インテル® SSE2)、インテル® ストリーミング SIMD 拡張命令 3 (インテル® SSE3)、ストリーミング SIMD 拡張命令 3 補足命令 (SSSE3) 命令セットに関連する最適化およびその他の最適化が含まれます。インテルでは、インテル製ではないマイクロプロセッサーに対して、最適化の提供、機能、効果を保証していません。本製品のマイクロプロセッサー固有の最適化は、インテル製マイクロプロセッサーでの使用を目的としています。インテル® マイクロアーキテクチャーに非固有の特定の最適化は、インテル製マイクロプロセッサー向けに予約されています。この注意事項で対象としている特定の命令セットの詳細は、該当する製品のユーザー・リファレンス・ガイドを参照してください。

JPN/1407/PDF/XL/SSG/KS Intel_Resolve_Resource_Leaks_Eval_Guide/Rev-082812/Rev-021712