リアルタイム C++ コーディングのための C++/AngelScript ハイブリッド開発・実行環境の提案

A Proposal for C++/AngelScript Hybrid Development and Execution Environment for Real-time C++ Coding

5114EE01-0　鈴木 遼　　指導教員　長 幾朗 教授
SUZUKI Ryo　　Prof. CHOH Ikuro

概要: 本研究では、スクリプト言語の特徴であるインタラクティブなコード編集と、C++ の実行時性能を同時に達成するために、単一のソースコードを AngelScript[1] と C++ の 2 通りに解釈して並列にコンパイルし、コンパイルの状況に応じて実行コードを動的に切り替える手法を提案、実装した。パフォーマンス評価では、ネイティブコードに匹敵する実行時性能が得られ、スクリプト言語と同様のリアルタイムなコード編集も可能であることが示された。本手法は C++ コンパイラによるコード最適化を有効にしても動作するほか、プラットフォームに非依存で、既存のプロジェクトに比較的組み込みやすい。また、C++ のコンパイル中にスクリプトのコードが実行されることで、開発フローにおける時間当たりの計算処理量を最大化できる利点がある。

キーワード：C++, AngelScript, スクリプト言語
Keywords: C++, AngelScript, Scripting language

1. 背景と目的

C++ はコードのビルド（コンパイルとリンク）に要する時間が特に大きく、開発者からイテレーションやテストの機会を奪い、生産性を低下させる原因となっている。ビデオゲーム産業のように、C++ が主要な開発言語となっている現場では様々な対策がとられているが、コード編集のリアルタイム性と実行時性能を両立させるものはない。

そこで、本研究ではスクリプト言語の柔軟性と、ネイティブコードによる最大限の実行時性能を両立する開発・実行環境の実装を行った。

2. 関連研究

C++ のビルド時間による生産性低下を防ぐために、様々な手法が研究、実用されている。

スクリプト言語は、コンパイル時間が短く、一般にアプリケーションの実行中でもインタラクティブにコードやパラメータの変更ができる。実行には VM (Virtual Machine) が必要で、ネイティブコードと比べて実行時性能は低い。

Edit and Continue は、アプリケーションの実行を一時中断し、コードや変数の値を編集できるという Microsoft 製 C++コンパイラの機能である。メモリレイアウトを変える変更や、最適化されたコードには使用できないなどの制約がある。

モジュールシステム[2]は、マクロなど、高速なコンパイルの障害を取り除くための C++ の言語仕様の提案で、実用レベルの実装は開発されていない。

ホットリロード[3] は、アプリケーションの実行中に、クラスや関数の実行コードを、新しくコンパイルしたコードで更新するテクニックであり、DLL (Dynamic Link Library) の再リンクなどにより実装される。C++ のコンパイルが必要であることは変わらず、リアルタイムなコード編集を可能にはしない。

3. 実装

本研究で実装したシステムの構成を **図 1** に示す。

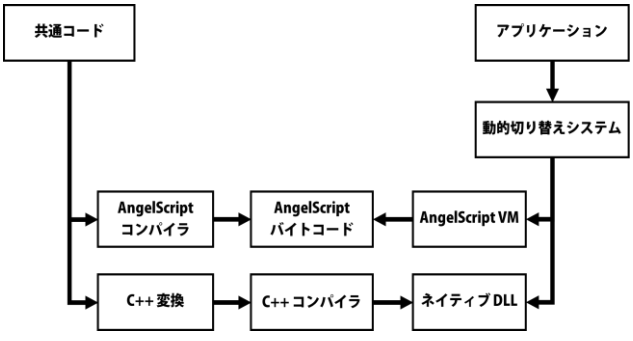

図 1　提案システムの構成

本手法では、ユーザは「共通コード」の仕様に準拠したコードを記述する。共通コードは AngelScript としてコンパイルできる構文をもち、一部制約はあるが C++ とほぼ同様のスタイルで記述できる。

コードが変更されると、システムは 2 つのスレッドを立ち上げる。1 つ目のスレッドでは、AngelScript コンパイラが共通コードからバイトコードを生成する。生成が完了すると、アプリケーションは実行対象を新しいバイトコードに切り替える。2 つ目のスレッドでは、共通コードを C++ コードに変換し、C++ コンパイラが DLL を生成する。生成が完了すると、アプリケーションは実行対象を AngelScript バイトコードから DLL のネイティブコードに切り替える。

生成した DLL はコードとともにバージョン管理される(**図 2**)。トラブルや変更のキャンセルが生じた際には、過去の DLL を再利用することで、アプリケーションの実行に中断や遅延を起こさずロールバックできる仕組みを実現した。

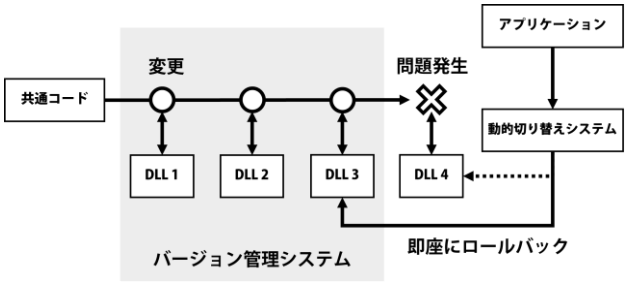

図 2　DLL のバージョン管理システム

共通コードには、C++ 変換時に解釈される特別な構文のコメントを記述できる。これにより、C++ コンパイル時にのみ有効になる文や、ユーザ独自の変換処理を可能にするなどの拡張性を持たせた。

4. パフォーマンス評価

C++ のホットリロード、AngelScript, 本研究のハイブリッドシステム、これら 3 手法についてパフォーマンス評価を行った。テスト環境のコンピュータは 4 コア 2.4GHz CPU と 8GB のメモリを搭載する。

ビルド時間の評価のために、3 つのサンプルプログラムを用い、コードの変更がアプリケーションに反映されるまでの時間を計測した。**表 1** の結果のとおり、本手法はスクリプト言語と同等の速度でコードの編集を反映できることが示された。

表 1　コードの編集が反映されるまでの時間

	Test1	Test2	Test3
C++	252ms	336ms	8210ms
AngelScript	3ms	3ms	3ms
ハイブリッド	2ms	4ms	4ms

実行時性能の評価のために、画像処理のサンプルプログラムを用いて、ユーザがコードを編集してから、更新されたコードにより処理された画素数を記録した。**図 3** の結果の通り、本手法は C++ のコンパイル中にスクリプトのコードを実行することで、時間当たりの処理量を最大化できることが示された。

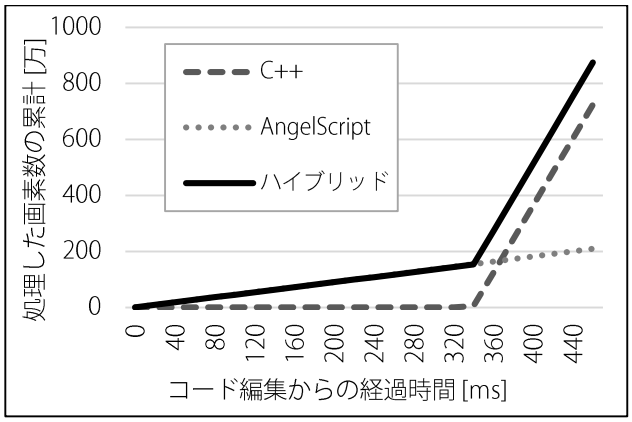

図 3　実行時性能（複合）

5. 結論

本研究では、単一のソースコードを AngelScript と C++ の 2 通りに解釈して並列にコンパイルし、コンパイルの状況に応じて実行対象を動的に切り替えるハイブリッド手法を実装し、C++ 開発におけるリアルタイムなコード編集とネイティブコードの実行時性能を両立する開発・実行環境を実現した。

本研究はビデオゲームやインタラクティブメディアの開発など、ラピッド・プロトタイピングの需要が高い C++ プロジェクトで活用可能であり、C++ 開発の生産性向上が期待される。

参考文献

[1] A. Jönsson, “AngelScript,” [online]. Available: http://www.angelcode.com/angelscript/

[2] M. H. G. N. Gabriel Dos Reis, “A Module System for C++” C++ Standards Committee's, 2015.

[3] S. Ishibashi, “Live Coding in C++,” CEDEC Digital Library, 2014.