

天に星、地に花、人にスクリプト

コンシューマ・ゲーム開発におけるスクリプト言語の利用

松田白朗/Hak Matsuda
CRI Middleware inc.

自己紹介

松田白朗(Hak Matsuda)

Lead Developer, CRI Middleware inc.

CriScript開発、オープンソースとして公開

マルチプラットフォーム向けミドルウェア開発

Xbox360、Xbox、DC、Saturn等ゲーム機のライブラリ
開発、技術サポート

San Francisco在住



本日のお題

- * 対象:これからスクリプトエンジンを組み込み/実装/高速化しようとする方への入門編
- * コンシューマゲーム開発へのスクリプト言語採用
 - * なぜコンシューマゲームにスクリプト言語が必要か
- * CriScriptの概要、実装について
 - * CriScriptとは何者か
 - * 言語系を実装、カスタマイズするにあたって

え、コンシューマゲームに

スクリプト??

スクリプト言語のメリット

- * 大規模開発では
スクリプト言語はすでにスタンダード
- * 生産性の向上 → クオリティの向上
 - * コンパイル、リンク時間が不要、その場で結果を確認
 - * 試行錯誤の回数が増える
→ クオリティに貢献
 - * 言語仕様の簡素化によるバグの低減
 - * C++比
- * !簡単に開発できる
 - * 主要な目的では無い

スクリプト言語採用例

- * LUA: Far Cry, World of Warcraft etc
- * UnrealScript: UnrealEngineにて使用
- * Squirrel: 小さな王様と約束の国(2008)
- * Python: GDC 2002: Game Scripting in Python (PC, 2002)
- * Lisp: Jack & Dexter(PS2, 2003)
- * 他沢山

今後の潮流

- * 五年後、ゲームコードの殆どはスクリプト言語で記述される
 - * Nativeコードと遜色無いパフォーマンス
 - * 又は富豪的アプローチ(遅くても十分速い)
 - * 生産性の向上
 - * 言語レベルでの並列性の確保
 - * 関数型言語
- * より高速、安定、生産性の高いスクリプトエンジンの組み込み
 - * ゲーム向けカスタマイズ
 - * 組み込み向け最適化
 - * プラットフォーム向け最適化

「純関数型プログラミングも、メニーコア時代のマルチスレッド・プログラミングには有効だ」(TIM SWEENEY氏)

CriScript概要

CriScriptとは

- * オープンソースのECMA262(ECMAScript/JScript)実装
 - * www.criscript.com にて公開
- * Win32/Xbox360/OSX/iPhone(移植中)で動作
- * ゲーム機向けにカスタマイズされた
軽量、高速なスクリプト言語実行環境
 - * をめざしてβ公開中

CriScriptライセンス

- * BSD系(よりも緩い)ライセンス

<http://criscript.com/trac/wiki/CRI%20Script%20Software%20License%201.0>

- * 改変、再配布自由
- * 商利用制限無し
- * その他表示義務等無し



言語仕様

- * ECMA262(JavaScript) + 一部ECMA4相当の仕様取り入れ
 - * 利点
 - * プログラマ人口が最も多い
 - * C++/Javaに近い文法の手続き型言語
 - * 欠点
 - * 仕様が煩雑
 - * 文字列の連結が不適切
 - * “;”自動挿入が曖昧
 - * evalの範囲が奇妙
 - * 文字コードがUTF-16
 - * 他沢山
 - * ECMA4仕様が破綻
 - * 全て連想配列
 - * パフォーマンス上の負荷が高い

標準仕様の採用

* 利点

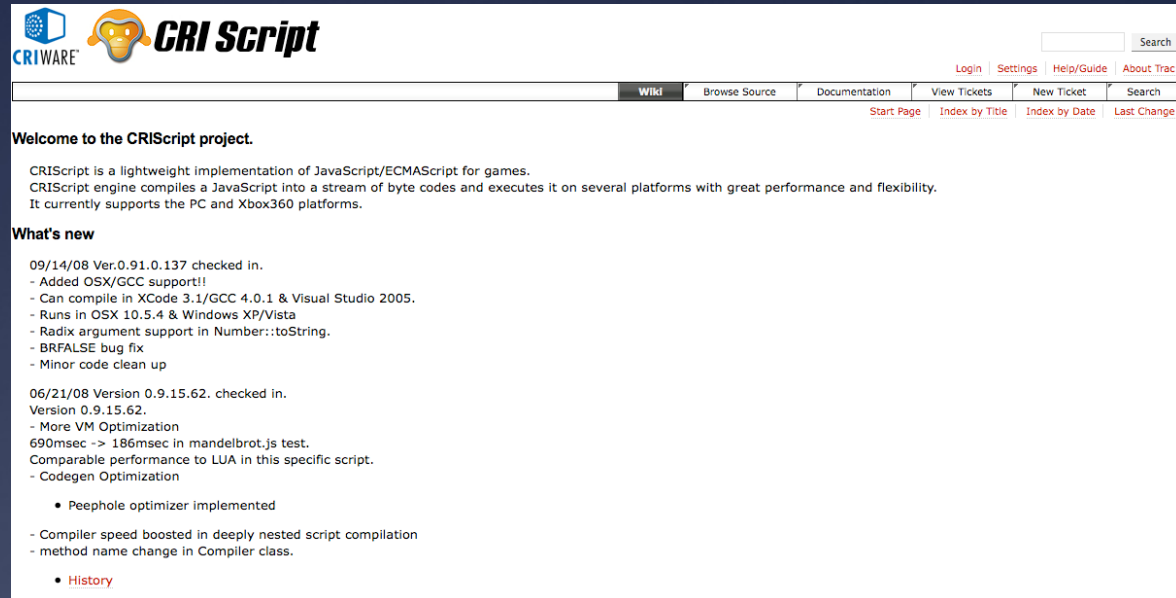
- * ある程度こなれた言語仕様
- * 既存のテストスイーツ、ライブラリコードが使える
- * 標準仕様である事自体の意味

* 欠点

- * 仕様規模が大きい
- * 大胆な自分的拡張は難しい
- * 組み込みに向いていない仕様もある

CriScript.com

- * tracによるサポートサイト
 - * SDKパッケージダウンロード
 - * SVNリポジトリ閲覧
 - * Bugzillaチケット管理
 - * ドキュメント
 - * メーリングリスト



The screenshot shows the Trac interface for the CriScript project. At the top left, there are logos for CRIWARE and CriScript. The main header includes navigation links: Login, Settings, Help/Guide, About Trac, and a search box. Below the header, there are tabs for Wiki, Browse Source, Documentation, View Tickets, New Ticket, and Search. The main content area starts with a welcome message: "Welcome to the CRIScript project." followed by a brief description: "CRIScript is a lightweight implementation of JavaScript/ECMAScript for games. CRIScript engine compiles a JavaScript into a stream of byte codes and executes it on several platforms with great performance and flexibility. It currently supports the PC and Xbox360 platforms." Below this is a "What's new" section with two entries. The first entry is dated 09/14/08 and describes version 0.91.0.137, listing features like OS support, compilation options, and bug fixes. The second entry is dated 06/21/08 and describes version 0.9.15.62, listing performance improvements and optimizations. At the bottom of the page, there are links for "Start Page", "Index by Title", "Index by Date", and "Last Change".

Welcome to the CRIScript project.

CRIScript is a lightweight implementation of JavaScript/ECMAScript for games. CRIScript engine compiles a JavaScript into a stream of byte codes and executes it on several platforms with great performance and flexibility. It currently supports the PC and Xbox360 platforms.

What's new

09/14/08 Ver:0.91.0.137 checked in.

- Added OSX/GCC support!!
- Can compile in XCode 3.1/GCC 4.0.1 & Visual Studio 2005.
- Runs in OSX 10.5.4 & Windows XP/Vista
- Radix argument support in Number::toString.
- BRFALSE bug fix
- Minor code clean up

06/21/08 Version 0.9.15.62. checked in.
Version 0.9.15.62.

- More VM Optimization
690msec -> 186msec in mandelbrot.js test.
Comparable performance to LUA in this specific script.
- Codegen Optimization
 - Peephole optimizer implemented
- Compiler speed boosted in deeply nested script compilation
- method name change in Compiler class.
 - History



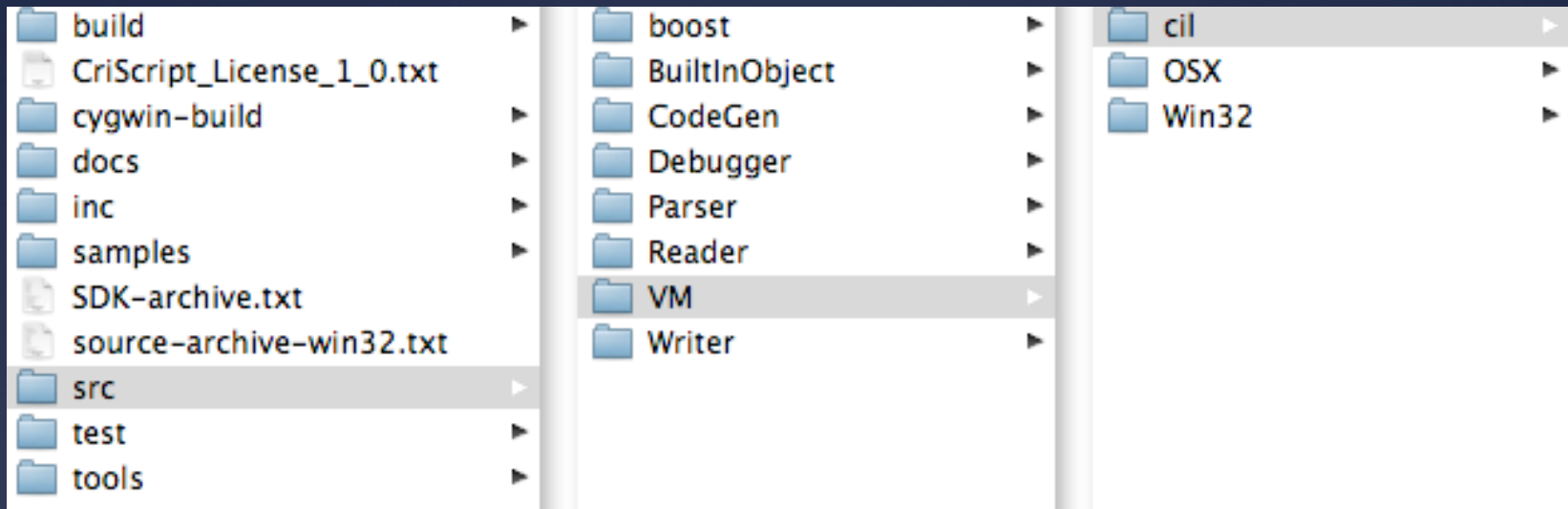
実装編

コード・ガイド

SVNリポジトリ: `svn://criscript.com`

username: "guest"

password: ""



コンポーネント

- * コンパイラ
 - * 字句解析
 - * 構文解析
 - * バイトコード生成
- * リンカ
 - * アドレス解決
 - * シンボル解決
- * VM(Virtual Machine)
 - * VMエンジン
 - * ECMA262標準ライブラリ
 - * C++バインディング

字句解析 (lexer)

- * スクリプト文字列 (UTF-16/BE-BOM) をトークンに分割
 - src/parser/lexer.cpp
 - * flexは未使用
 - * 実行速度
 - * コード規模
 - * 定数、コメント等のハンドリング
 - * bisonコードとのインタフェース
 - * yylex()

構文解析 (parser)

- * Bison使用

src/parser/criscript.y

- * BNF定義を元にパーサを生成

- * LALR(1) パーサ

- * セミコロン処理がトリッキー

- * 構文ノードツリーを作成、コード生成は別パス

- * ライセンス:GNUの特例で問題無し

- * BNFでの構文管理

- * メリット

- * メンテナンスが楽

- * デメリット

- * コンパイラコードのサイズ

- * パフォーマンス

- * 実機上ではコンパイルしない前提



コード生成 (CodeGen)、リンカ

- * 構文解析パスで生成した構文ノードツリー解析

src/codegen/cil/cilCodegen.cpp

- * バイトコードを作成
- * 定数テーブルの生成
- * 構文解析パスではバイトコード生成をしない
 - * ノードツリー単位での最適化を想定

- * リンカ

src/codegen/cil/cilLinkage.cpp

- * ジャンプ命令の解決
- * 現状1オブジェクトファイルのみ入力のため
 - * シンボル解決は行っていない

バイトコード

- * CIL、JavaByteCode、ActionscriptByteCode等を検討
- * CIL仕様を元にした独自バイトコード
 - * .Netで使用されている中間コード
 - * 評価スタックベースのVM
 - * 公開仕様
<http://msdn.microsoft.com/en-us/netframework/aa569283.aspx>
 - * 一部使用を独自拡張
 - * 合成命令
 - * Inc/dec
 - * 変数型の明示的指定
- * これから選ぶならLLVMがお勧め

VM

* スタックマシン

- * オペレーション結果は全てスタック経由で受け渡し
 - * 実装が簡易
 - * ⇄レジスタマシン

* ディスパッチ・ループ

vm/cil/cilVM.cpp

- * 単純なswitch文
- * ダイレクトスレッド、コンテキストスレッド等の手法
vm/cil/cilVMThreadedDispatcher.cpp
 - * 現状それほど効果無し

VM

- * ゲームエンジン・バインディング
 - * Jscript->C++
 - * C++->Jscript
 - * 関数呼び出し
 - * 変数参照
- * 標準API
 - BuiltinObject/cil/*.cpp
 - * 標準オブジェクトのプロパティメソッド
 - * ECMA仕様で規定
 - * Boost::Regexを使用
 - * Date等はゲーム機向けには不要論

VM

* オブジェクト管理

Vm/cil/cilVmObject.cpp他

- * リファレンスカウンタ

- * GC

 - * Mark&Sweep

 - * 他にも最適化の余地は多い

 - * 仕組みは入れているが、現状では自動発動なし

ネイティブコード生成、JIT

- * ゲーム機としてはチャレンジング
 - * セキュリティモデル上の問題
 - * 非署名コードの実行
 - * C++ソースファイルを生成？
 - 技術的には可能
 - * JITパフォーマンスの問題
 - * ランタイム上での不定なタイミングでの負荷
- * CriScriptでは基本的にバイトコードを静的コンパイル
 - * 静的なevalのみ許可
 - * 動的に文字列を与えるとエラー

パフォーマンス

- * ネイティブコード比ではどうしても遅い
 - * バイトコード実行
 - * ネイティブコード比1/10程度が現実的&理想的
 - * LUAのパフォーマンスをリファレンスに
- * 最適化
 - * VM最適化
 - * バイトコード・コード生成最適化
 - * 高速化に適した言語仕様の追加

VM最適化

- * VMの典型的なボトルネック
 - * ディスパッチ・ループ
 - * バイトコードを読み込むメイン処理
 - * ディスパッチ方式
 - * 評価スタック、評価レジスタへのアクセス
 - * オペレーション毎に評価スタックにアクセスする
 - * Variable構造体へのアクセス
 - * グローバル変数、ローカル変数
 - * レジスタ、メモリアクセスと同質
 - * API呼び出し
 - * 疑似スタックフレームのセットアップ

最適化の指針

- * IA32, PPC, ARMプラットフォーム
 - * それぞれの特徴を意識
 - * IA32: 命令数を減らす、正統的な最適化が有効
 - * PPC: パイプライン・ストールを意識
 - * ARM: バスへのRead/Writeを意識、相対的に低速
- * プロファイリング環境を整える
- * コード生成を見る
 - * リスティングファイル

バイトコード生成最適化

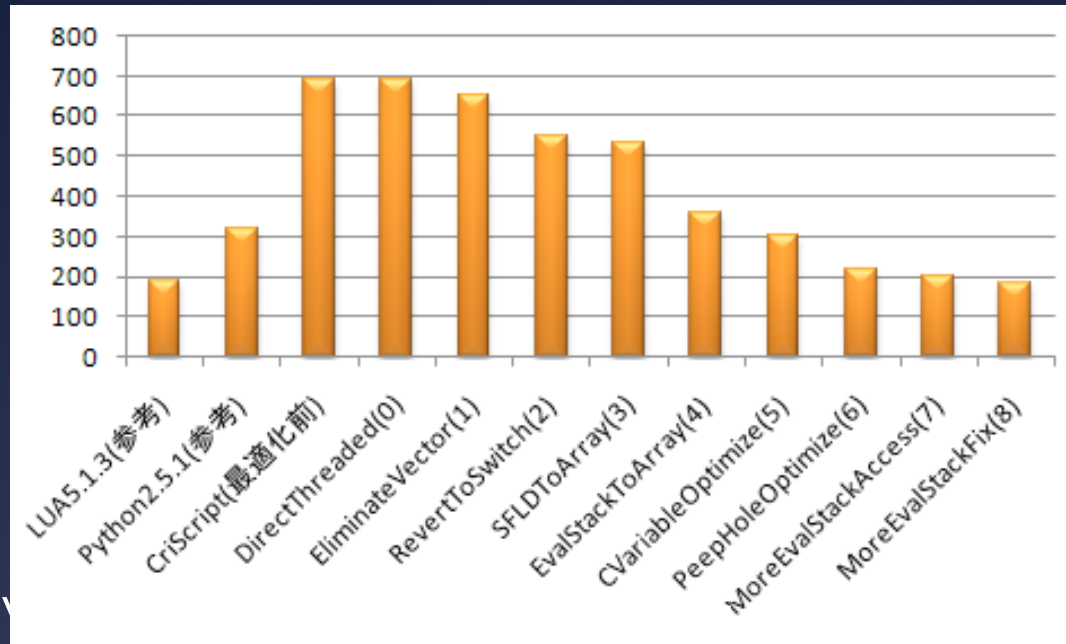
- * バイトコードの効率を上げて実行速度向上
 - * ピープホール最適化
 - * パターンマッチングによるコード置き換え
 - * 演算強度の軽減
 - * 簡単な演算に畳み込み
 - * ノードツリー解析時に個別コードで処理
 - * ループ最適化(plan)
 - * ループアンローリング
 - * ループ外への演算移動
 - * Dead Code Elimination(plan)
 - * 不要コードの除去

言語仕様の拡張

- * 高速なコード生成が出来るような言語拡張
 - * class定義(plan)
 - * プロパティのインデクスアクセス(≠連想配列アクセス)が可能に
 - * 明示的な型指定
 - * 専用バイトコードと組み合わせてオペレータ処理を軽くできる
 - * `var foo:int;`
- * ゲーム機にフィットする拡張
 - * yield等のフロー制御
 - * vector等のSIMD処理
 - * 個別ゲーム用拡張

最適化例

* Mandelbrot集合演算



- * 今後ボトルネックにあわせての最適化が必要

メモリフットプリント

- * コンパイラ(.text + .data)
 - * 411KB
 - * Bison生成コード 92KB
- * VM(.text + .data)
 - * 216KB
 - * Regex 49KB
 - * boost::regex
 - * +バッファ
 - * バイトコード、グローバル変数テーブル
 - * 動的オブジェクト
- * 不必要なコンポーネント/ライブラリの切り離し等(plan)

テスト、ドキュメント

- * 2000ファイル、1.5M行程度のTest Suite
 - * 一部テストケースについては未対応
- * Nightly Build時に自動テスト
 - * テスト通過後に自動Deploy
- * API reference
 - * Doxygenを使用してヘッダから生成
<http://www.stack.nl/~dimitri/doxygen/>
- * その他、書き起こしドキュメント
 - * www.criscript.com
- * 現状、英語ドキュメント

今後の予定

- * 未実装仕様
 - * closure
 - * 標準ライブラリ実装
- * ゲーム向け機能追加
 - * yield
 - * vector型
 - * プラットフォーム・サポート
 - * バイナリ出力
- * その他拡張
- * コントリビュータ大募集中

振り返って

- * うまく行った点
 - * ECMA仕様の採用
 - * 仕様がクリア
 - * テストプログラム
 - * Bisonの使用
 - * 立ち上げが楽
- * 改善が必要な点
 - * バイトコードフォーマット
 - * LLVM
 - * 変数構造体の設計
 - * cVariableはもう一工夫
 - * 標準APIの実装範囲

ついでにIXMF

* IXMFとは

- * IASIG提唱のインタラクティブ・オーディオ用バイトコード仕様

- * インタラクティブオーディオに必要な仕様を網羅

- * コンテナフォーマットとしてXMFを利用

- * 拡張可能なバイトコード仕様

- * 言語仕様は規定しない

- * ツールGUIからのバイトコード生成を想定

* ドラフト仕様書

- * <http://www.iasig.org/wg/ixwg/index.shtml>

- * SCEE、CRI Middleware inc. 😊が支持

まとめ

- * コンシューマゲーム開発へのスクリプト言語採用は必要
- * スクリプト言語の開発、実装、メンテナンス、最適化について
- * CriScriptの正体
- * IXMFに期待

リファレンス

* 参考書籍

Compilers: Principles, Techniques, and Tools (2nd Ed.) ISBN: 978-0321486813

プログラミング言語処理系 (岩波講座 ソフトウェア科学) ISBN: 978-4000103459

Expert .NET 2.0 IL Assembler ISBN: 978-1590596463

* ゲームエンジンへのスクリプト採用

* Postmortem: Naughty Dog's Jak and Daxter: the Precursor Legacy

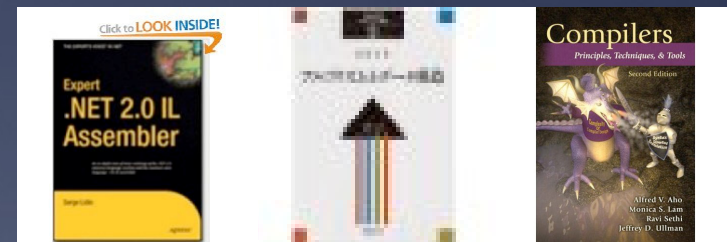
http://www.gamasutra.com/features/20020710/white_02.htm

* GDC 2002: Game Scripting in Python

http://www.gamasutra.com/features/20020821/dawson_01.htm

* VM最適化

* <http://hak.wablog.com/79.html>



ご質問？

* hak@cri-mw.com