Eötvös Lóránd Tudományegyetem
Programozási nyelvek és fordítóprogramok tanszék

Strukturált formában tárolt programszöveg

Diplomamunka

Témavezető
Králik Barnabás
(volt) Doktorandusz hallgató

Készítette
Thier Richárd István
Programtervező Informatikus MSc

2018. március 6.

Témabejelentő

Az elfogadott és aláírt témabejelentőt a dolgozat második oldalaként szükséges elhelyezni, amelyet az adminisztrációban lehet átvenni. Itt tehát már a tanszéki pecséttel ellátott és aláírt lapot kell belefűzni (ezen oldal *helyett*, ez az oldal csak útmutatás). Az elektronikusan feltöltött dolgozatba már nem szabad beleszerkeszteni ezt a feladatkiírást.

Tartalomjegyzék

Kivonat

A dolgozat témája egy strukturált formában tárolt programszöveg és feldolgozási mód vizsgálata, amely egy fordítóprogram-motor alapját képezheti - a 3D futtató-motor rendszerekkel analóg módon. Az említett 3D-motor technológia már láthatóan uralja a 3D játékfejlesztés és megjelenítés területét, de akárcsak a fordítóprogramok esetében, korábban a 3D technológiát is ad-hoc, egyedi fejlesztések jellemezték. Idővel az eljárások szoftverkönyvtárakká, keretrendszerekké és legvégül az iparban is használt teljeskörű futtatómotorrá fejlődtek. A szoftverkönyvtárak lényegében a hasznos függvények gyűjteményét jelentik, a keretrendszerek már egy kiterjeszthető rendszert adnak, de a fordítóprogram és a programozási nyelvek technológiájában a következő lépés, a teljeskörű futtatómotor még nem létezik.

A dolgozatban egy új strukturált reprezentációt, plugin-kezelési és scriptelési módszert definiálunk, elemzünk, vizsgálunk, ami egy C++14 implementációval megvalósítva, továbbfejleszthető alapot szolgáltat egy ilyen futtatómotor kifejlesztéséhez. A készülő kezdeti rendszer, a „turul” (Tree-Using Really Unorthodox Language) a futtatókörnyezetet fogja képezni, az erről külön is leválasztható adatreprezentácó pedig a „tbuf” (vagy turbobuf) nevű alrendszerben található. Ez utóbbit a fő forrásokról leválasztva, külön kezeljük, mert egy gyorsan olvasható, egyedi faszerkezetet leíró reprezentációt ad. Az így kapott megoldás „nyelv”-ként történő elnevezését az adja, hogy bár egy futtatókörnyezetről van szó, annak adatszerkezete - és az ezzel történő „scriptelés” - egy programnyelvet képez. Az utóbbiak alapján tehát a megoldásra nem csak egy programozási és fejlesztési környezetként, hanem egy eddig nem látott metaprogramozási képességgel felruházott önmódosító nyelvként is tekinthetünk: az elnevezésben ezt a terminológiát követtük.

A dolgozat célja elsősorban a strukturált formában tárolt és feldolgozott programszöveg, a scriptelhető elképzelések által felvetett lehetőségeknek, illetve annak az elemzése, hogy ez az elképzelés milyen módon különbözik a fordító-fordítók (pl. lex, yacc) keretrendszer-jellegű képességeitől. A dolgozat másodsorban kitér az elképzelés megvalósításához használható algoritmusokra, a hatékony implementációs lehetőségekre. A dolgozattal párhuzamosan készülő szoftver-komponensek egy kiterjeszthető futtató-motort adnak, amely további kutatások alapját képezheti és nem csak oktatási célra - de egyszerűbb kísérlegi fordítóprogramok, vagy programozási nyelvek kiértékeléséhez is használható.

Abstract

In this document we investigate a structured source code representation and processing approach with a goal to create „programming language engines" with similar architectural choices of the so-called „3D engines". These 3D engines are visibly dominating the field of 3D game and content development, but for a long time not only compilers, but also the games were created in ad-hoc ways, which later evolved into code „libraries", „frameworks" and more recently/finally complete „engines". While a code library is just a collection of useful methods, and a framework is just a set of rules that define an extensible system - engines provide an all-encompassing and structured way to handle the specifics of a field. Until now I do not know about any works that approach the field of compiler construction using this structured way.

In this document we define, analyse and investigate the structured data-based approach and „plugin scripting" that serves as an extensible architecture of a modern C++14 implementation for the forementioned system. The system is called turul (Tree-Using Really Unorthodox Language) and the readable-binary representation of trees that are used through the solution is called „tbuf" (or turbo-buf) for its hoped processing speed. Despite we are talking about an „engine" for languages, the system can be thought as an extensible language with unseen metaprogramming support too and in the final naming we stick to this terminology.

In our goal to investigate possibilities of more structured ways of compiler constructrion using the engine approach, the differences of the engine approach and compiler-compilers (like lex and yacc) and the underlying algorithms of the efficent implementation is documented as a reference with proper baseline analysis of the ideas. Implementation serves as an extensible engine that can serve as the basis for further research and useful not only in teaching - but for smaller experimental compiler construction to evaluate programming language ideas too.

1. fejezet

Bevezetés

TODO: A bevezető tartalmazza a diplomaterv-kiírás elemzését, történelmi előzményeit, a feladat indokoltságát (a motiváció leírását), az eddigi megoldásokat, és ennek tükrében a hallgató megoldásának összefoglalását.

TODO: A bevezető szokás szerint a diplomaterv felépítésével záródik, azaz annak rövid leírásával, hogy melyik fejezet mivel foglalkozik.

2. fejezet

A strukturált formában tárolt programszöveg bevezetése

TODO: rövid bevezető a fejezethez

2.1. A tbuf-faszerkezetek reprezentációja

TODO

2.2. Egy tbuf-alapú FORTH-szerű programnyelv

TODO

2.3. Nyelvtanok leírása tbuf/tbnf alakban

TODO

2.4. Levezetési fák ábrázolása tbuf szerkezetben

TODO

2.5. Szemantikai lépések ábrázolása tbuf-al

TODO

2.6. Futtatókörnyezet-konfiguráció tbuf-al

TODO

3. fejezet

A futtatómotor megvalósítása

TODO: Rövid bevezető a fejezethez

3.1. Célkitűzések

A futtatómotor megvalósítása során a következő szempontok vezették a tervezési lépéseket:

- Lightweight és hatékony megoldások kialakítása szükséges, gyors fordítási időket támogatva.
- Cél az alacsony hardver igény - akár régi egy-magos, vagy RPi is megfelelő kell legyen.
- Újonnan fejlesztett rendszerként a modern C++14 megoldások alkalmazását helyezzük előtérbe.
- Alapvetően open source, linux-alapú környezetekre telepíthető megoldást kell kialakítani.
- Fontos továbbá, hogy a lehető legkevesebb külső függőséggel rendelkezzen a készülő rendszer.

TODO: További Magyarázat

3.2. Fordítási idejű C++ modulok

TODO

3.3. Fordítási idejű C++ pluginok a szódefiníciókhoz X-macro technikával

TODO

3.4. Egy megfelelő multi-stack implementáció

TODO

3.5. Parser előtétmodulok

TODO

TODO: egyéb pontok - itt át kell gondolni ezt még

4. fejezet

A futtatómotor alkalmazása és felhasználása

TODO: Bevezetés a fejezethez

4.1. Fordítás és telepítés

TODO

4.2. Standalone működés

TODO

4.3. Beágyazás más rendszerbe

TODO

4.4. Használat interpreterként

TODO

4.5. Használat fordítóprogramként

TODO

4.6. Példák

TODO

TODO: itt is át kell még gondolni a fejezeteket is

5. fejezet

Összefoglaló

TODO: A diplomaterv összefoglaló fejezete.

Köszönetnyilvánítás

TODO: KÖszönetnyilvánítás

Táblázatok jegyzéke

Ábrák jegyzéke

Irodalomjegyzék

A. függelék

Függelék

A.1. Válasz az „Élet, a világmindenség, meg minden” kérdésére

A Pitagorasz-tételből levezetve

$$c^2 = a^2 + b^2 = 42.$$

A Faraday-indukciós törvényből levezetve

$$\mathrm{rot}E = -\frac{dB}{dt} \quad \longrightarrow \quad U_i = \oint\limits_{\mathbf{L}} \mathbf{Edl} = -\frac{d}{dt} \int\limits_{A} \mathbf{Bda} = 42.$$