Nathan Braswell

Junior Computer Science student at Georgia Tech

Cell: 678 457 6856 - nbraswell12@gatech.edu
GitHub: https://github.com/Limvot

EDUCATION

Georgia Institute of Technology, *Atlanta, GA*

- Candidate for Bachelor of Science in Computer Science (4.0 Major Requirements, 3.74 Overall GPA)
    - Dean's list Fall 2012, Fall 2013, Faculty Honors Spring 2013, Summer 2013, and Spring 2014.
    - Has chosen the Systems & Architecture and Artificial Intelligence threads
    - A in all programming classes
- Proficient in **C++, Python**, and **Java**, and enjoy researching and playing with new and different languages, like **Rust** and **Haskell**

EXPERIENCE

Work:

Intern at Mozilla on the JavaScript Engine team
Summer 2014

Working at Mozilla in an internship improving Firefox's JavaScript Engine, SpiderMonkey.

2110 TA
2014-Ongoing

I am a 2110 TA under Professor Leahy, which I find very fulfilling and quite enjoyable. Helping students in a class that I loved understand hard concepts and being a part of a team of fun and talented people has been a wonderful part of my college career.

ResNET Temp for Fall 2013 Move-In
2013

Worked in a team with other students for ResNET, the on campus Internet and IT fix-it group. Together, we made sure over 2 thousand students successfully configured and connect their devices to the wired and/or wireless Internet.

Personal:

RNGLR parser/AST/C backend for in-progress compiler for a new programming language (Kraken)
2013-ongoing -https://github.com/Limvot/kraken

After experimenting with creating WTS-language below, I decided to create a “real” systems programming language using industry standard techniques. After researching different parsing techniques and reading papers describing different techniques, I eventually settled on RNGLR, an extremely powerful algorithm that can parse any context-free grammar (CFG). I have also written a grammar for a new language that allows the RNGLR to parse Kraken source code, the resulting tree of which is pruned into an AST, which is traversed by the C backend to emit C code. This is all fully functional, and the language, Kraken, even supports some advanced features such as function and class templates, as well as operator overloading.

Features

- The language supports classes, mutually recursive definitions, function and class templates.
- Compiler can use any context-free grammar (CFG) for parsing.
- Creates a parse tree that can be exported into a .dot file, for use with graph rendering software.
- Contains multiple tree-walking passes that turn the parse tree into an AST, which can then be traversed by the C backend to generate functional C code.

Compiler and Interpreter for test language, written in C++, "WTS-Language"
2012-2013 - https://github.com/Limvot/WTS-Language

Created a compiler/interpreter for a very simple integer-based test language. This project was certainly experimental, but currently can both interpret the language and compile it to C code. The language is very basic, but has been used to write a factorial program, a Fibonacci number program, and a prime number sieve program.

Features

- Language can be interpreted or compiled to C code.
- The parser creates an abstract syntax tree representing the program which is then traversed by either an interpreter or compiler.

Python-based LinuxBuildSystem
2012-2013 - https://github.com/Limvot/LinuxBuildSystem

Wrote a package-manager / Linux From Scratch system builder. It can currently compile an entire GNU/Linux system from scratch based on the Linux From Scratch project, then install it self on the new system and be used to install additional programs, including Python, Xorg, and many others.

Features

- Written in Python
- Creates Bash Shell scripts that can be executed at a later date.
- Resolves dependencies between packages.
- Uses a simple text-based file format for package specification.

C++ OpenGL 3D Graphics Engine
2012 - https://github.com/Limvot/Phenomenon-Engine
Programmed a game-focused graphics engine in C++ and OpenGL, using SDL, GLEW, and stb_image that works in both Linux and Windows. As a hobby/self-teaching project, it focuses on fleshing out the concepts of 3D rendering, and (as of now) does not contain many optimizations.

Features

- Fully deferred renderer, unlimited point lights.
- Flexible node-based scene graph.
- Simple .obj and .mtl object and material loader.