Oromë Pathfinder

by

Jonas Brown

1. Features

- *Blazing fast version of A* path-finding (up to 10x faster then generic MSVC++ versions).*
- *Very light memory footprint (58 bytes per node).*
- *Customizable path-finding (dijkstra's, greedy BFS, include diagonals).*
- *System independent ("pure" x86-64 code, no syscalls or API usage).*
- *Wrappers for different programming languages (C, C++, .NET).*
- *Low-level threading support is abstracted away from the user (pre-threaded interfaces).*
- *Link-based path-finding for use in navmeshes (WIP – not done yet).*
- *True 3D path-finding (WIP – not done yet).*

2. Motivation

- *Proving that confined algorithms written in assembler are faster then compiled code.*
- *Proving that I have the knowledge to write confined algorithms in assembler.*
- *Proving that I can interface the assembler code with any x86 based system or architecture.*
- *Giving game AI programmers and robotic enthusiasts a fast alternative to other path-finding systems.*
- *Although I make use of these 'concepts' throughout the system,* ***THIS IS NOT*** *:*
  - *An x86-64 assembly guide!-->*
  - *An A* Path-finding guide!-->*
  - *An NASM guide!-->*
  - *An managed/native inter-operability guide!-->*

3. NASM Source

3.0. INTRO

The NASM source code is heart and soul of the Oromë path-finding system. Written in NASM for extreme speed, it contains the actual path-finding algorithm. It is system neutral and only requires a processor capable of executing x86-64 instructions. It works on Linux, Windows and MacOS. The system is strictly 64-bit, do NOT use it in any 32-bit programming environment, nor any 32-bit OS. You can interface with the FindPath interface, by using MS fastcall64 calling convention to pass the params to the NASM source code.

4. Modules

4.0. INTRO

A description of the modules which make up the Oromë path-finding system. These modules "wrap" the algorithm code, so it can be used in different programming environments and on different systems.

4.1. OFP.DLL

The NASM source code linked with a PE-header for use on the Win64 platform.
Calling Convention: https://msdn.microsoft.com/en-us/library/ms235286.aspx
NASM Windows (v2.12.02rc6): ***nasm*** *-O 3 -f win64 *.asm -o *.o64 -l *.lst*
Cygwin64 LD (v2.25.2): ***ld*** *-shared -O 3 -s -e DllMain -o opf.dll "all_files".o64 -Map=opf.map*

4.2. LIBOFP.SO

(WIP – Not done yet) The NASM source code linked with a ELF-header for use on Unix64 platforms.
Calling Convention: https://github.com/hjl-tools/x86-psABI/wiki/X86-psABI
NASM Linux (v2.10.09): ***nasm*** *-O 3 -f elf64 *.asm -o *.elf64 -l *.lst*
GNU LD (v2.24): ***ld*** *-shared -soname -O 3 -s -o libopf.so "all_files".elf64 -Map=libopf.map*

4.3. OPF_NATIVE_WIN_WRAPPER.DLL

This module has two main purposes; firstly, it provides an entry point to the native windows world. You should use it, if you wish to write a native *C* or *C++* program (or any native language with windows dynamic linking support) using the Oromë Pathfinder. The second purpose of the module is, to contain the operation of different path-finding algorithms, within their own execution units (aka threading and synchronizing). Since an Oromë path-finding algorithm rely heavily on stack usage, it is always best to run it within an isolated execution unit, with the correct stack size. The "threaded" path-finding interfaces contained within this module, handles this for you. If you wish to provide your own threading, the "non-threaded" interfaces are also available. But be warned; using the non-threaded interfaces with in a larger application (such as a game) is error-prone as hell, and simply a recipe for disaster, if you don't know what you are doing. Bottom line; use the threaded interfaces or provide your own threading support for the normal interfaces.

4.4. OPF_MANAGED_WIN_WRAPPER.DLL

This module provides an entry point to the managed windows world (aka *.Net*). It relies heavily on the "threading" system of the native wrapper. Therefore, any application using this module, will also, by proxy, be using the native wrappe (opf_native_win_wrapper.dll). Use this wrapper if you plan on using the Oromë path-finding system in a *.Net* application. I have implemented some managed *interfaces* to describe the minimum application requirements.

5. Tools

5.0. INTRO

A desciption of the tools for testing the path-finding system. The tools have two purposes; firstly, they serve as a demonstration of how to interface your native or managed code with the Oromë Pathfinder. Their second purpose is, to test, stress-test and debug different aspects of the pathfinder algorithm. The tools are accessible by running the executable files in the project. The tools will remain undocumented/uncommented, because they are not considered part of the core path-finding system.

5.1. OPF BENCHMARK

Written using the native wrapper, the benchmark'er works on large random data. Use it to determine whether your configurations are better served using the 'threaded' or 'non-threaded' module. Use with care; large maps use a lot of memory and your system might become unstable, if you don't have enough memory to support the benchmarking operation. There is no fail safe, since different systems can handle different workloads. As a rule of thumb; if you keep crashing or get buggy results, the workload is probably too stressful for your system.

5.2. OPF VISUAL

Written using the managed wrapper, this is a visual demonstration of the Oromë Pathfinder, working in real time. You can use it to plot paths on randomly generated grids, to get an idea of how the algorithm works and what to expect. It is not meant for large maps, since it should fit within screen-space. For 'large map' testing and algorithm stress-testing, use OPF Benchmark.

6. Author

- **Name -** Jonas Brown
- **Contact -** brown8220@hotmail.com
- **About -** .NET developer by trade and education. C++ and Assembly programmer by passion.

7. Credits

- **ANeDe -** Thanks for the beautiful art piece of Oromë.
  - Site **-** http://anede.deviantart.com/
- **Netwide Assembler** - For providing the assembler.
  - Site - http://www.nasm.us/
- **Cygwin** - For providing the gnu bintools on Windows.
  - Site - https://www.cygwin.com/
- **Notepad++** - For providing a no nonsense IDE.
  - Site - https://notepad-plus-plus.org/
- **x64dbg** - For providing the best 64-bit alternative to ollydbg.
  - Site - http://x64dbg.com/