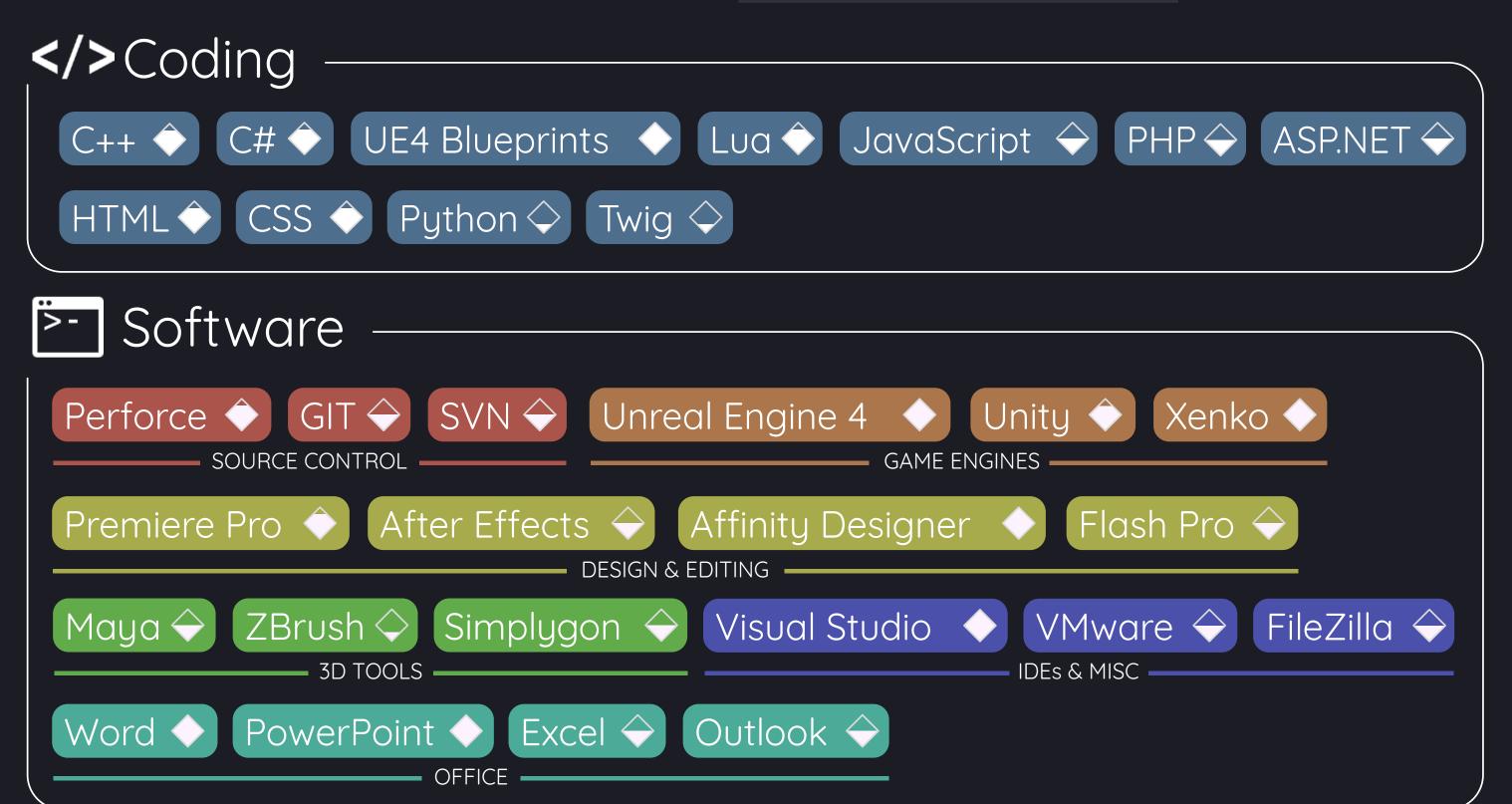
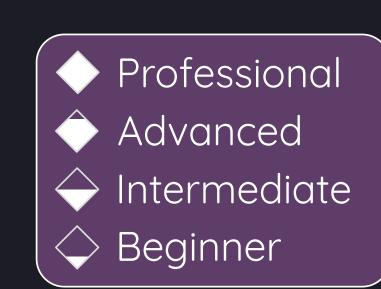
~



Print-friendly version: http://hugo.fyi/resume_print

SKILLS

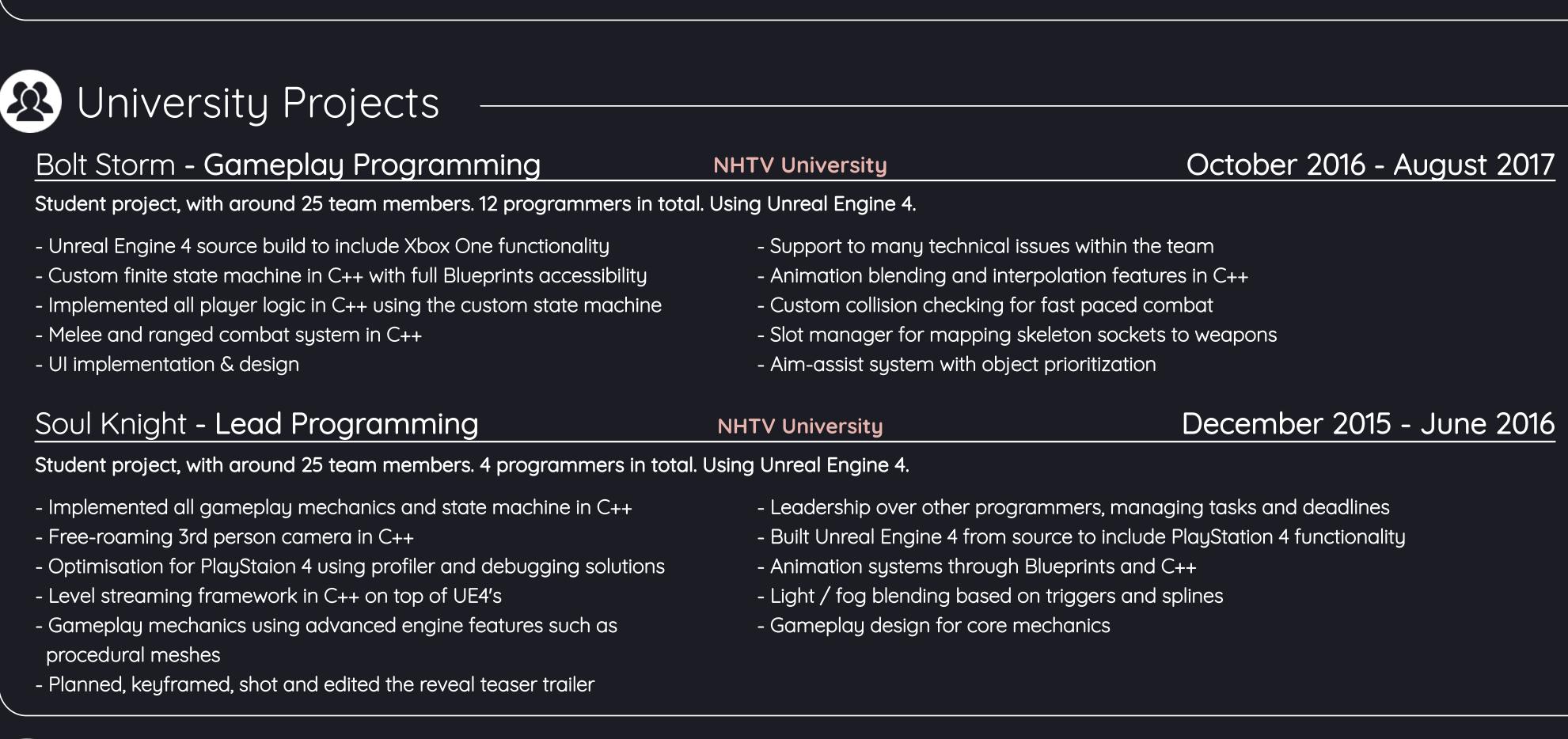


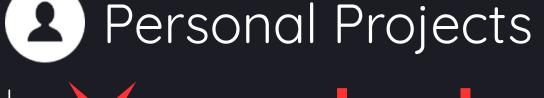


EXPERIENCE

Learn more about these and other projects on my portfolio: http://hugo.fyi/









July 2016 - Present

April 2014 - ?

The Netherlands

Game engine from scratch in C++, with custom C# build tools. This description will be condensed once project page is up. - Cross-platform support for Windows x86/x64, WebAssembly / Emscripten. DX11 / OpenGLES (emulated using Google ANGLE on Windows)

- Easily extendible and flexible systems, most built on a small custom type registry system, including a material editor (generated shaders, custom shader pipeline using MCPP, hlslparser, glsl-optimizer), model editor (import using FBX SDK), sequence editor (component based tracks, "clips", curve channels), curve editor (interpolation types ala Blender), animation editor (using an entirely custom nodegraph system), level editor (component based entity system, combining ideas from Unreal and Unity), template editor (like UE4's Blueprints or Unity's prefabs), asset pipeline (custom asset "cooker" tool built on top of the engine, does things like platform-dependent shader generation, texture conversion, platform packaging), json / binary based serialization (supports custom types, emphasis on fast binary deserialization, during production assets are stored in json, get baked to binary using the asset cooker) - Libraries/APIs used include: Bullet Physics, Emscripten, FBX SDK, hIslparser, mcpp (shader generation / preprocesser parsing), glsl-optimizer, NoesisGUI, dear imgui

(stripped out in release mode), OpenAL (audio with plugin system that supports custom decoders like FluidSynth), rapidjson, plf-colony (used for storing things like entity components), stb, **sdl** (on Emscripten, OpenGL ES on Windows emulation), **zlib** - Rendering is currently limited to forward rendering, using a PBR implementation based on Google's Filament renderer. Uses a custom baked light probe solution to

achieve IBL. I have written a deferred renderer on top of the engine as well, but this is not mainline (integration of different rendering modes is pending..) - Entirely modular, each module is a separate Visual Studio project, with dependency rules. Code has a module framework, with at least one module per "project"

- Build toolchain written in C#, features a module rules compiler (compiles module rules to single DLL, keeps tracks of changes etc.), custom incremental compilation, taskscheduler for compilation/linking tasks, very fast "nothing to do" detection, generation of engine version stamp, Visual Studio project generation)

DirectX 11 Renderer November 2015 - January 2016 Basic renderer using abstracted DirectX 11 API, supports Physically Based Rendering

- Basic implementation of render windows in Qt - Physically Based Rendering (using pre-existing shaders)

- Input handling for keyboard and gamepad

The Runthrough Music / rythm action-arcade game. Went through multiple redesigns / rewrites, now working on the final revision using my 'Warlock' engine. - 'Track Development Tool' - slick level creator with backgrounds effects editor,

December 2011 - Present

music scrubbing, login / account / licensing system and more. Reverse Engineering / Porting 'Beyond: Two Souls' - Went from Game Maker to C# to C++ - A lot of hours put into this...

I was asked to stop working on it by the CEO... - Reversed class system, type/id registration code

- Reversed Lua bytecode by making a converter for big/little endian - Implemented a custom Lua framework for auto-generated game scripts - Implemented custom class system with binary components - Around 6 full rewrites of my 'port' from the ground up - Most of the game playable on PC, with models, but no shaders :(

'Higher General Secondary Education' (HAVO)

- Reversed and implemented sequences (camera shots, dialog, audio, script events, etc), audio streaming, model / vertex formats, GUI middleware "Menus Master", choice events / branching story, user actions, Lua function handlers, area / scene loading, videos, more? - Literally boots the game like a PS3 would, natively - not a remake

M EDUCATION

'International Game Architecture & Design' NHTV Breda University of Applied Sciences, The Netherlands September 2014 - June 2018

2010 - 2014

Bachelor of Science (BSc) - graduated cum laude





Bolt Storm - Winner Best Student Technical Achievement I designed and implemented most gameplay systems in the game, from player movement to combat, gameplay scripting, the

game's tutorial and more. We were nominated alongside two other projects, from a total of 20 projects.



Bolt Storm won this award at our University, NHTV Breda University of Applied Sciences, in my 3rd year. There were 7 other projects eligible for these rewards.

Best Code in a Student Project



Dutch Game Awards 2016

Soul Knight - Nominee Best Student Game Design

Best Code in a Student Project

I was heavily involved in the design process of Soul Knight, designing most of the gameplay mechanics.



Unreal Engine Community Highlights Feature

Soul Knight was featured in the July edition of Epic Games' community highlights video.

We were nominated alongside two other projects, from a total of 25 projects.



Soul Knight won this award at our University, NHTV Breda University of Applied Sciences, in my 2nd year, as well as 'Best Art