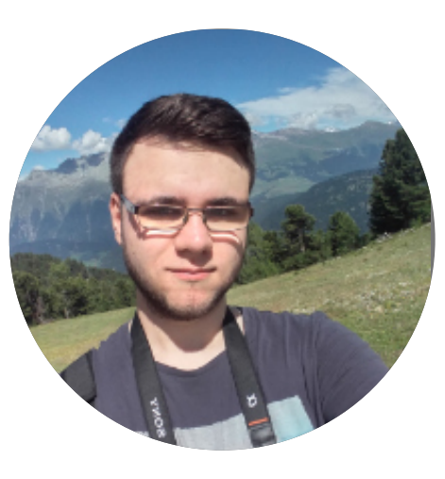

Hugo Peters

GAME DEV, PROGRAMMER, DESIGNER

mail@hugo.fyi ✉
23 years old, birthday 28/12/1996
born in Leiden, The Netherlands

Eye-friendly version: **http://hugo.fyi/resume**

SKILLS

</>Coding

C++ ◈ C# ◈ UE4 Blueprints ◆ Lua ◈ JavaScript ⬙ PHP ⬙ ASP.NET ⬙
HTML ◈ CSS ◈ Python ◇ Twig ◇

- ◆ Professional
- ◈ Advanced
- ⬙ Intermediate
- ◇ Beginner

Software

Perforce ◈ GIT ⬙ SVN ⬙
SOURCE CONTROL

Unreal Engine 4 ◆ Unity ◈ Xenko ◆
GAME ENGINES

Premiere Pro ◈ After Effects ⬙ Affinity Designer ◆ Flash Pro ⬙
DESIGN & EDITING

Maya ⬙ ZBrush ◇ Simplygon ⬙
3D TOOLS

Visual Studio ◆ VMware ⬙ FileZilla ⬙
IDEs & MISC

Word ◆ PowerPoint ◆ Excel ⬙ Outlook ⬙
OFFICE

EXPERIENCE

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

Professional Experience

UBISOFT

Junior Game Programmer

Ubisoft / Massive Entertainment - Malmö, Sweden January 2018 - Present

UBISOFT

Game Programmer Internship

Ubisoft / Massive Entertainment - Malmö, Sweden

University Projects

Bolt Storm - Gameplay Programming **NHTV University** October 2016 - August 2017

Student project, with around 25 team members. 12 programmers in total. Using Unreal Engine 4.

- Unreal Engine 4 source build to include Xbox One functionality
- Custom finite state machine in C++ with full Blueprints accessibility
- Implemented all player logic in C++ using the custom state machine
- Melee and ranged combat system in C++
- UI implementation & design
- Support to many technical issues within the team
- Animation blending and interpolation features in C++
- Custom collision checking for fast paced combat
- Slot manager for mapping skeleton sockets to weapons
- Aim-assist system with object prioritization

Soul Knight - Lead Programming **NHTV University** December 2015 - June 2016

Student project, with around 25 team members. 4 programmers in total. Using Unreal Engine 4.

- Implemented all gameplay mechanics and state machine in C++
- Free-roaming 3rd person camera in C++
- Optimisation for PlayStaion 4 using profiler and debugging solutions
- Level streaming framework in C++ on top of UE4's
- Gameplay mechanics using advanced engine features such as procedural meshes
- Planned, keyframed, shot and edited the reveal teaser trailer
- Leadership over other programmers, managing tasks and deadlines
- Built Unreal Engine 4 from source to include PlayStation 4 functionality
- Animation systems through Blueprints and C++
- Light / fog blending based on triggers and splines
- Gameplay design for core mechanics

Personal Projects

warlockengine *Project page coming soon!* July 2016 - Present

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**, **hlslparser**, **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, task-scheduler 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 December 2011 - Present

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, music scrubbing, login / account / licensing system and more.
- Went from Game Maker to C# to C++
- A lot of hours put into this...

Reverse Engineering / Porting 'Beyond: Two Souls' April 2014 - ?

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 :(
- 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

EDUCATION

'International Game Architecture & Design' NHTV Breda University of Applied Sciences, The Netherlands

September 2014 - June 2018
Bachelor of Science (BSc) - graduated cum laude

'Higher General Secondary Education' (HAVO) The Netherlands

2010 - 2014

AWARDS

Dutch Game Awards 2017

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.

Best Code in a Student Project

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.

Dutch Game Awards 2016

Soul Knight - Nominee Best Student Game Design
I was heavily involved in the design process of Soul Knight, designing most of the gameplay mechanics.
We were nominated alongside two other projects, from a total of 25 projects.

Unreal Engine Community Highlights Feature

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

Best Code in a Student Project

Soul Knight won this award at our University, NHTV Breda University of Applied Sciences, in my 2nd year, as well as 'Best Art in a Student Project' and 'Best Game' after our first few months of development (after each 'block' awards were given).
There were 16 other projects eligible for these rewards.

Version 1.7 PRINT - 05/08/2020