How I spent my summer vacation

or, my adventure with Skyrim modding

I tried to do a WASM startup last here. We had some fun ideas, but we went to pitch VCs at exactly the moment every single one of them aligned like iron filings near a magnet toward Large Language Models instead of cryptocurrencies. So we gave up. I went to sit at home and sulk for a while. I don't enjoy sulking, so I thought, well, why don't I play some video games? Specifically, I went for a video game comfort food. I went for Skyrim.

fus-ro-dah!

What's Skyrim anyway?

Skyrim is an award-winning 2011 game by Bethesda Game Studio, the fifth in their Elder Scrolls series of role-playing games. It has an amazing sound track, middling graphics, and some fascinating lore. It's been released on PC plus every game console there is. There's a VR version. Game director Todd Howard would release it on your toaster if he could.

Skyrim is a region of the larger Elder Scrolls world, roughly equivalent to Scandinavia with a Viking-ish civilization. There's a civil war in progress as a larger human Empire falls apart because of what the player did in the previous two Elder Scrolls games. To make things worse, a dragon surprises everybody by existing and destroying a town.

It opens kinda like this.

What's Skyrim? Here is its opening sequence. The wipe goes from vanilla Skyrim to what my current modlist looks like. I think Lokir of Rorikstead is a bit too cute for the character.

^

Skyrim
== mods

Mods change the look a lot, huh? If you've been playing Fallout again recently because of the TV show, you know that it goes further than this. Bethesda games overall are very moddable.

Skyrim is the sweet spot

— the game is beloved **but**...
— the rendering is terrible
— the bugs are bad
— the UI was designed for consoles
— but all of this can be fixed...
— because the game is moddable!

I built a **gigantic** list of **2000+** mods

So let's go back to my sulk-ending period of Skyrim modding. I built a ridiculous modlist for myself, everything I'd always wanted to play with. It was something like 2000 plugins before I started trying to winnow it down.

But...

I couldn't find a hotkeys mod that wasn't

a. slow
b. buggy
c. both buggy and slow

Except for one HUD mod that worked but wasn't quite right. So I forked it.

Skyrim has a LOT of items in it, and mods add more. You can end up having to dive into menus to eat food, drink potions, and switch weapons a lot. Hotkeys mods make this better. HUD + hotkey mods show you what you've got equipped. I wanted something specific.

This is what I wanted: to set up lists of things that I could cycle through by tapping a single button. So for the right hand I could swap from sword to bow to a spell to a gigantic axe back to sword by tapping one button for the right hand. If you've played a Souls game with a controller, that would be the right direction button. Same for left hand items, then potions and food, then shouts. This saves you menu-diving, which is kinda unimmersive.

Time to learn how to **write** mods

not just use them

a rapid intro to Creation Engine modding

the Creation Engine is designed for modding

Modding is essential to Bethesda's own workflow.

— The games are developed with the same tool they distribute to modders.

— (They do hack out proprietary libraries.)

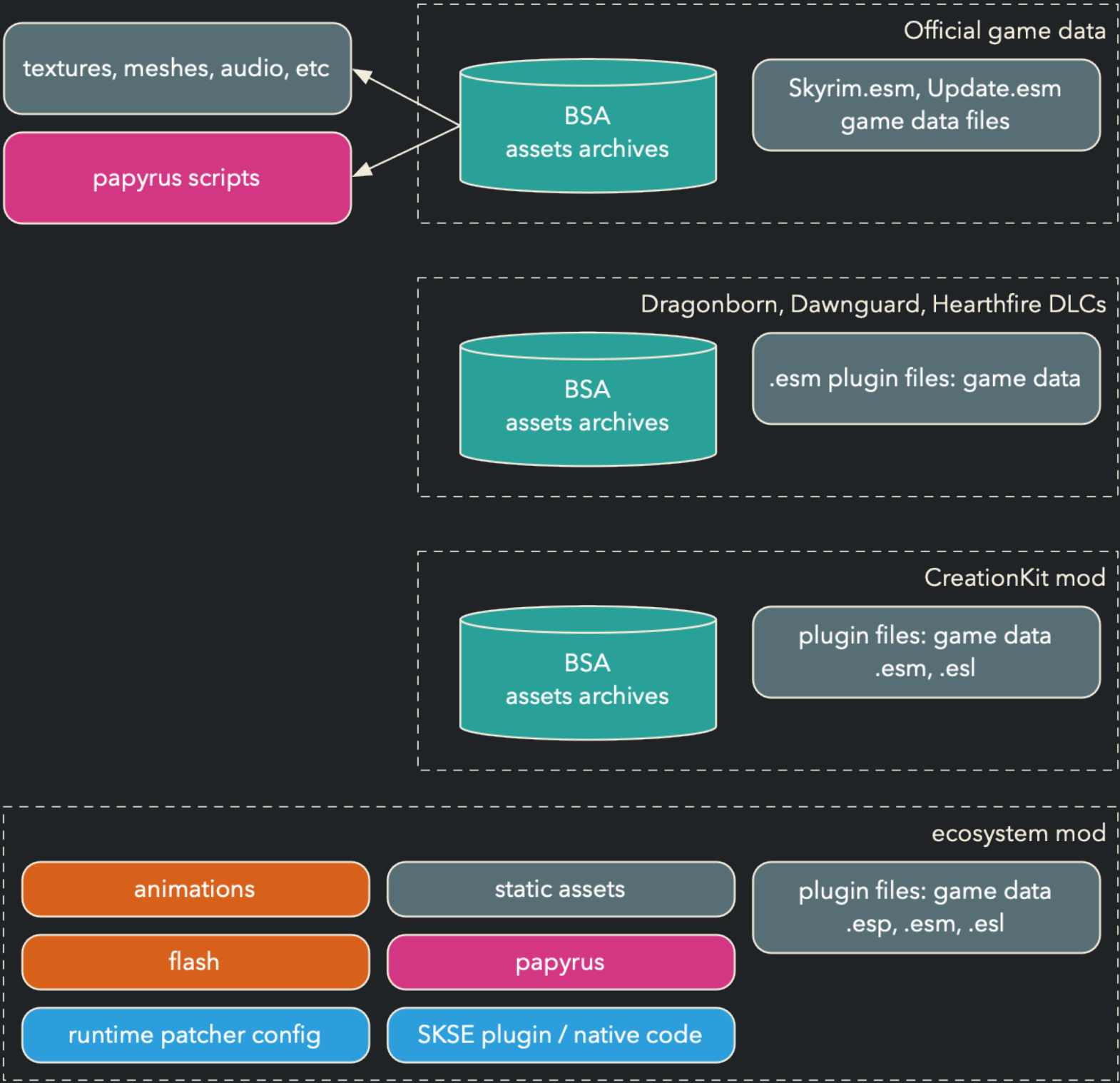— Official DLCs use the same mechanisms as mods to update game data.

The engine itself has nowhere near the wow factor of Unreal Engine. It has all the basics, for sure, but it doesn't have high frame rates or raytracing. Bethesda made all of its trade-offs differently.

the game data is designed for overlays

— "form" is the unit of game data: a sound, an armor component, a skill, etc

— form types are implemented in the game engine

— plugins define new form records or override existing form records

— game engine loads plugins in a user-controlled order

— **last-loaded** form record wins

— last-loaded asset archive wins; **loose files** > archives

What's a mod?

A mod is a package of game assets linked together by one or more plugin files. Plugins are databases of game records that reference assets and other game records. Static assets might be packaged in an archive or left loose in the game's data directory.

Official game data

textures, meshes, audio, etc

papyrus scripts

BSA assets archives

Skyrim.esm, Update.esm game data files

Dragonborn, Dawnguard, Hearthfire DLCs

BSA assets archives

.esm plugin files: game data

CreationKit mod

BSA assets archives

plugin files: game data .esm, .esl

ecosystem mod

animations

flash

runtime patcher config

static assets

papyrus

SKSE plugin / native code

plugin files: game data .esp, .esm, .esl

the file formats

— slowly evolving for 20 years

— fully reverse-engineered

— community toolset is less buggy than Bethesda's provided tools

— no seriously: `xEdit` fixes problems with plugin data generated by the CreationKit

— complex workflows are supported

Help Videos NexusMods GitHub Discord Patreon Ko-Fi PayPal

Filter by Name: and by Value: Keep children siblings parent's siblings

FormID	EditorID	Name		[00] Skyrim.esm	[10] Unofficial Skyrim Special Edition Patch.esp	[3D] AI Overhaul.esp	[5B] BUVARP SE RE.esp	[6D] NPC Appearances Merged.esp	[FE 412] synthesis_npcs.esp	[FE 413] synthesis_fusion.esp
00013390	Froki	Froki	Record Header	[v44] Sven [NPC_:0001347F] (Compressed)	[v44] Sven [NPC_:0001347F] (Compressed)	[v44] Sven [NPC_:0001347F]	[v44] Sven [NPC_:0001347F] (Compressed)	[v44] Sven [NPC_:0001347F]	[v44] Sven [NPC_:0001347F]	[v44] Sven [NPC_:0001347F]
0001339E	Hroki	Hroki	EDID - Editor ID	Sven	Sven	Sven	Sven	Sven	Sven	Sven
0001339F	Imedhnain	Imedhnain	VMAD - Virtual Machine Ada...							
000135A1	Klepyr	Klepyr	Version	5	5	5	5	5	5	5
000135A5	Lisbet	Lisbet	Object Format	2	2	2	2	2	2	2
000135A6	Logrolf	Logrolf the Willful	Scripts (sorted)							
0001339B	Erith	Erith	Script							
0001339E	Omluag	Omluag	ScriptName				TrainerGoldScript		TrainerGoldScript	TrainerGoldScript
000133B0	Orla	Orla	Flags				Local		Local	Local
000133B1	PavoAttius	Pavo Attius	Properties (sorted)							
000133B4	Rhiada	Rhiada	Property				Gold001: Object = Gold001 [MISC:0000000F]		Gold001: Object = Gold001 [MISC:0000000F]	Gold001: Object = Gold001 [MISC:0000000F]
000135B5	Rogatus	Rogatus Salvius	Property				TrainerType: Int32 = 2		TrainerType: Int32 = 2	TrainerType: Int32 = 2
000135B6	Rondach	Rondach	Script							
000135B7	Senna	Senna	ScriptName	WIDeadBodyCleanupScript	WIDeadBodyCleanupScript		WIDeadBodyCleanupScript	WIDeadBodyCleanupScript	WIDeadBodyCleanupScript	WIDeadBodyCleanupScript
000135B8	Skaggi	Skaggi Scar-Face	Flags	Local			Local		Local	Local
000135B9	Sosia	Sosia Tremellia	Properties (sorted)							
0001338A	Uaile	Uaile	Property	DeathContainer: Object = WIDeadBodyNonAccessibleDeathC...	DeathContainer: Object = WIDeadBodyNonAccessibleDeathC...	DeathContainer: Object = WIDeadBodyNonAccessibleDeathC...	DeathContainer: Object = WIDeadBodyNonAccessibleDeathC...		DeathContainer: Object = WIDeadBodyNonAccessibleDeathC...	DeathContainer: Object = WIDeadBodyNonAccessibleDeathC...
0001338E	UrzogaGraShugurz	Urzoga gra-Shugurz	Property	WI: Object = WI [QUST:00015D64]	WI: Object = WI [QUST:00015D64]	WI: Object = WI [QUST:00015D64]	WI: Object = WI [QUST:00015D64]		WI: Object = WI [QUST:00015D64]	WI: Object = WI [QUST:00015D64]
000133BF	Vigdis	Vigdis Salvius	OBND - Object Bounds	(-22, -14, 0), (22, 14, 128)	(-22, -14, 0), (22, 14, 128)	(-22, -14, 0), (22, 14, 128)	(-22, -14, 0), (22, 14, 128)	(-22, -14, 0), (22, 14, 128)	(-22, -14, 0), (22, 14, 128)	(-22, -14, 0), (22, 14, 128)
000133C9	Voada	Voada	ACBS - Configuration							
00013477	Dorthe	Dorthe	Flags (sorted)	Auto-calc stats, Unique, PC Level Mult	Auto-calc stats, Unique, PC Level Mult	Auto-calc stats, Unique, PC Level Mult	Auto-calc stats, Unique, PC Level Mult	Auto-calc stats, Unique, PC Level Mult	Auto-calc stats, Unique, Doesn't affect stealth meter, PC Level...	Auto-calc stats, Unique, PC Level Mult
00013479	Orgnar	Orgnar	Auto-calc stats	Auto-calc stats	Auto-calc stats	Auto-calc stats	Auto-calc stats	Auto-calc stats	Auto-calc stats	Auto-calc stats
0001347A	LucanValerius	Lucan Valerius	Unique	Unique	Unique	Unique	Unique	Unique	Unique	Unique
0001347B	CamillaValerius	Camilla Valerius	Doesn't affect stealth...						Doesn't affect stealth meter	
0001347C	Gerdur	Gerdur	PC Level Mult	PC Level Mult	PC Level Mult	PC Level Mult	PC Level Mult	PC Level Mult	PC Level Mult	PC Level Mult
0001347D	Hod	Hod	Protected						Protected	
0001347E	Frodnar	Frodnar	Magicka Offset	0	0	0	0	0	0	0
0001347F	Sven	Sven	Stamina Offset	0	0	0	0	0	0	0
000134A8	Faendal	Faendal	Level Mult	0.750000	0.750000	0.750000	0.750000	0.750000	0.750000	0.750000
000135E5	Agni	Agni	Calc min level	6	6	6	6	6	6	6
000135E8	Benor	Benor	Calc max level	20	20	20	20	20	20	20
000135ED	Jonna	Jonna	Speed Multiplier	100	100	100	100	100	100	100
000135EE	Joric	Joric	Disposition Base (unused)	0	0	0	0	0	0	0
000135EF	Lemi	Lemi	Template Flags							
000135F0	Hrogar	Hrogar	Health Offset	0	0	0	0	0	0	0
000135F1	Vikmund	Vikmund	Bleedout Override	0	0	0	0	0	0	0
00013685	GesturRockbreaker	Gestur Rockbreaker	Factions (sorted)							
00013687	Sirgar	Sirgar	SNAM - Faction	TownRiverwoodFaction [FACT:00013481] [Rank: 0]	TownRiverwoodFaction [FACT:00013481] [Rank: 0]	TownRiverwoodFaction [FACT:00013481] [Rank: 0]	TownRiverwoodFaction [FACT:00013481] [Rank: 0]	TownRiverwoodFaction [FACT:00013481] [Rank: 0]	TownRiverwoodFaction [FACT:00013481] [Rank: 0]	TownRiverwoodFaction [FACT:00013481] [Rank: 0]
00013688	Aeri	Aeri	SNAM - Faction	CrimeFactionWhiterun [FACT:0002678A] [Rank: 0]	CrimeFactionWhiterun [FACT:0002678A] [Rank: 0]	CrimeFactionWhiterun [FACT:0002678A] [Rank: 0]	CrimeFactionWhiterun [FACT:0002678A] [Rank: 0]	CrimeFactionWhiterun [FACT:0002678A] [Rank: 0]	CrimeFactionWhiterun [FACT:0002678A] [Rank: 0]	CrimeFactionWhiterun [FACT:0002678A] [Rank: 0]
0001368D	Kodir	Kodir	SNAM - Faction	RiverwoodSvensHouseFaction [FACT:00035D41] [Rank: 0]	RiverwoodSvensHouseFaction [FACT:00035D41] [Rank: 0]	RiverwoodSvensHouseFaction [FACT:00035D41] [Rank: 0]	RiverwoodSvensHouseFaction [FACT:00035D41] [Rank: 0]	RiverwoodSvensHouseFaction [FACT:00035D41] [Rank: 0]	RiverwoodSvensHouseFaction [FACT:00035D41] [Rank: 0]	RiverwoodSvensHouseFaction [FACT:00035D41] [Rank: 0]
0001361C	Leifur	Leifur	SNAM - Faction	JobLumberjackFaction [FACT:00051595] [Rank: 0]	JobLumberjackFaction [FACT:00051595] [Rank: 0]	JobLumberjackFaction [FACT:00051595] [Rank: 0]	JobLumberjackFaction [FACT:00051595] [Rank: 0]	JobLumberjackFaction [FACT:00051595] [Rank: 0]	JobLumberjackFaction [FACT:00051595] [Rank: 0]	JobLumberjackFaction [FACT:00051595] [Rank: 0]
00013613	Bulfrek	Bulfrek	SNAM - Faction	JobBardFaction [FACT:00053514] [Rank: 0]	JobBardFaction [FACT:00053514] [Rank: 0]	JobBardFaction [FACT:00053514] [Rank: 0]	JobBardFaction [FACT:00053514] [Rank: 0]	JobBardFaction [FACT:00053514] [Rank: 0]	JobBardFaction [FACT:00053514] [Rank: 0]	JobBardFaction
00013615	Frida	Frida	SNAM - Faction	PotentialFollowerFaction [FACT:0005C84D] [Rank: -1]	PotentialFollowerFaction [FACT:0005C84D] [Rank: -1]	PotentialFollowerFaction [FACT:0005C84D] [Rank: -1]	PotentialFollowerFaction [FACT:0005C84D] [Rank: -1]	PotentialFollowerFaction [FACT:0005C84D] [Rank: -1]	PotentialFollowerFaction [FACT:0005C84D] [Rank: -1]	PotentialFol...
00013617	Froki	Froki	SNAM - Faction	CurrentFollowerFaction [FACT:0005C84E] [Rank: -1]	CurrentFollowerFaction [FACT:0005C84E] [Rank: -1]	CurrentFollowerFaction [FACT:0005C84E] [Rank: -1]	CurrentFollowerFaction [FACT:0005C84E] [Rank: -1]	CurrentFollowerFaction [FACT:0005C84E] [Rank: -1]	CurrentFollowerFaction [FACT:0005C84E] [Rank: -1]	CurrentFol...
00013619	Karl	Karl	SNAM - Faction	BardSingerFaction [FACT:000CF8F9] [Rank: -1]	BardSingerFaction [FACT:000CF8F9] [Rank: -1]	BardSingerFaction [FACT:000CF8F9] [Rank: -1]	BardSingerFaction [FACT:000CF8F9] [Rank: -1]	BardSingerFaction [FACT:000CF8F9] [Rank: -1]	BardSingerFaction [FACT:000CF8F9] [Rank: -1]	BardSinger...
0001361A	Karita	Karita	SNAM - Faction				JobTrainerFaction [FACT:000E3A49] [Rank: 0]		JobTrainer...	

View Referenced By (56) Messages Information Weapon Spreadsheet Armor Spreadsheet Ammunition Spreadsheet What's New

[01%] Background Loader: finished

ActionScript

— UI is all Scaleform

— this is a custom Flash engine designed for embedding in games

— FromSoft games also use it, even though it's been dead for a few years

— limited to only UI use cases and a PITA to work with

Papyrus scripts

— Bethesda's scripting language with hooks into the engine

— it's compiled but...

— it's pretty slow

— it hooks into the game engine to affect game state

— the game itself is implemented in papyrus

Fix and extend via reverse engineering

Game reverse engineering has a long and storied history.

SKSE, F4SE, SFSE script extenders

The "script extender" series of mods originally extended Papyrus by hooking into more game functions. It's now important as a native-code plugin loader. It also provides reliable ways for mods to store save data alongside game data.

This is the basis of UI replacement mods like SkyUI. Todd Howard says SkyUI is his favorite Skyrim mod, so Bethesda knows this exists.

RE projects are organized & professional

— they're serious C++ experts
— with a side helping of RE & x86 assembly experience
— RE tools in use: mostly IDA, some Ghidra (the free NSA RE tool)
— they do work to reduce their own toil

Before Starfield turned out to be disappointing, with a fundamentally flawed engine, the RE community had made and executed on a plan to coordinate on working on archive format changes, plugin format changes, new form records, and so on. A working script extender was available in record time. The community also unlocked plugin loading, which allowed modding to take place immediately. (Bethesda still has not released the Starfield Creation kit. The RE community believes this is because they figured out how badly they'd broken modding with this engine release via form item layering screwups.)

example: Address Library

— designed to make coping with engine updates easier
— changed vtable offsets are RE-ed at least partially with tools like addresslibgen
— the offset library mod is updated
— mod authors don't have to do anything
— players need download only one update

During a period when Skyrim was getting frequent releases, the RE community invented a way for SKSE plugin mods to look up game offsets on the fly, from a library that provides offsets for a number of versions.

CommonLibSSE-NG aka CLib

— a reverse-engineered C++ library

— everything we understand about the game engine and its classes

— field structure/sizes and vtables; with types if a human has investigated

— with tools for offset look-up

— requires SKSE at runtime, not at compile time-- completely separate code & far more extensive

— supports all known game builds

xbyak jitted assembly

— injects code at runtime

— allows you to hook functions

— can trampoline from original function to your code and back

— CLib wraps all this up nicely so I didn't have to learn this

What native-code mods do

— call methods to get game information/state/events

— insert your own functions before or after existing game functions

— replace game functions entirely (more rarely)

— example: my mod intercepts menu selection events, inventory changes

— it also grabs the DX11 render context so it can draw a HUD

Skyrim / Creation Engine

Bethesda's internal engine, designed to support their game concepts. Descendent of the Gamebryo engine. Orange indicates tech Bethesda licenses that they do not support in modding.

DX11

Flash/AS game ui / menus	rendering layer

audio	form impls	actors	skill system	more...

input	papyrus VM	plugin loader	Havok physics	hkx anims
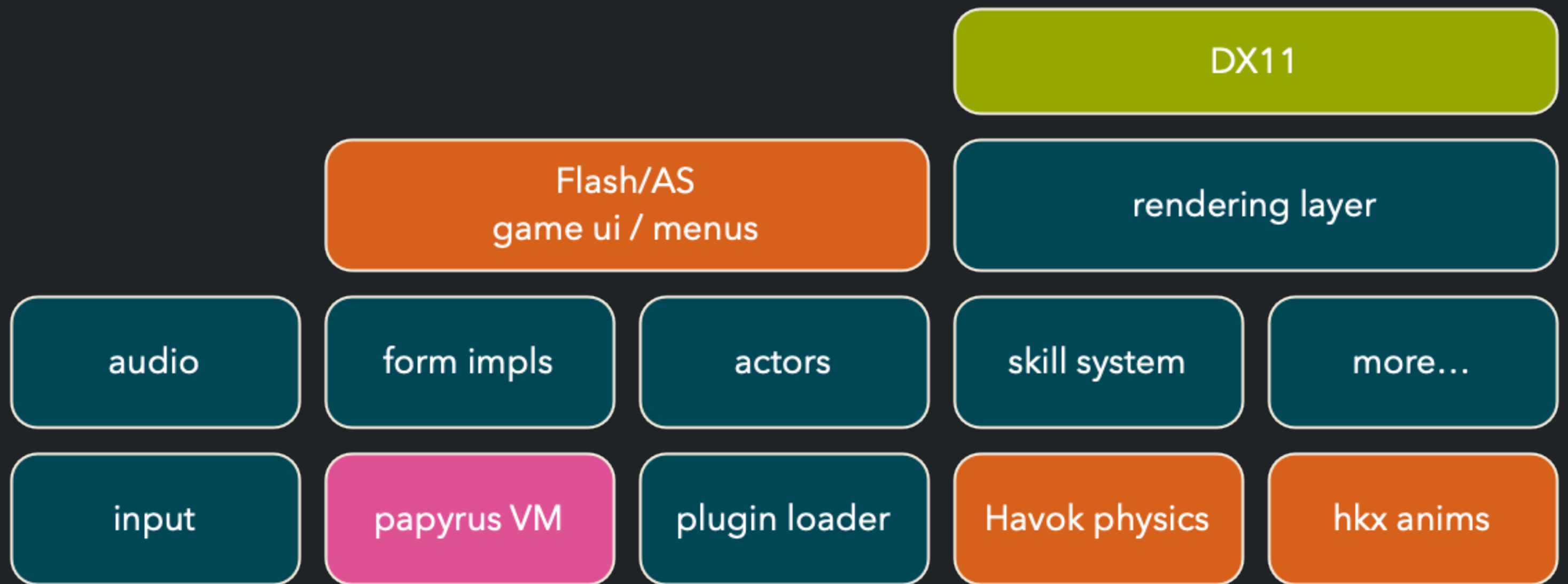
If a block diagram looks kinda like this...

Reverse-engineering

Skyrim's engine is not fully reverse-engineered, but the community knows enough to have extensively modified its behavior. Bugs are fixed, rendering is given modern enhancements, gameplay features are extended and hooked into. Animations in particular have been modernized, enhancing combat as well as making NPCs more life-like.

DX11

SKSE

IMGUI ui overlays — **ENB graphics post-processing** — **CommonLibSSE**

SkyUI — **Community Shaders rendering modernization**

Flash/AS game ui / menus — **rendering layer** — **lighting engine fixes** — **game crash fixes** — **game system fixes** — **game system extensions**

audio — **form impls** — **actors** — **skill system** — **more…**

input — **papyrus VM** — **plugin loader** — **Havok physics** — **hkx anims** — **behaviors & animation chaining**

papyrus extensions — **back-compat ports of ESL** — **collision physics** — **animation annotations**

run-time patchers — **hair/cloth physics** — **on-the-fly animation replacement**

This is which parts of the game engine have been reverse engineered. Anything a modder wanted to replace or fix, ever-growing.

Back to my forked HUD mod

— it's a CommonLibSSE mod: C++

— it uses **imgui** to draw UI not Flash

— this is why it was fast

— but other design decisions made it flakey

— and I had an idea about how to do menu interactions

I can fix it, right?

a learning project

— modern C++

— Windows development

— imgui

— Skyrim itself

— could I learn about Rust/C++ interop?

how **hard** can it be?

cue A-Team theme song

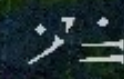# say hi to Soulsy HUD

I can no longer imagine playing the game without it.

division of labor

— C++: SKSE plugin responsibilities
— C++: source of game state info
— C++: render loop
— Rust: controller for HUD state (all app logic)
— Rust: layout parsing & settings enforcement
— Rust: unified logging

So how hard was it?

Harder than I thought, but not the way I expected.

— Windows strings were a pain until I understood them
— Skyrim internals are not always sensible
— mod interactions made the problem more complex than expected
— had lots of bugs for a long time until it fell together
— players asked for far more features than I thought I wanted (but they were right)

understanding Skyrim itself

— SKSE and CLib are undocumented

— CLib is more methodical but much larger

— found my favorite C++ mod authors and read all of their code

— $%@! extra data & inventory what the what now?

— a lot of trial and error

I cannot overstate the importance of the skill of reading undocumented code and figuring out what it does.

Windows development

This was a surprising challenge to me. I had not done a lot of development on Windows through my career, so I had to learn how to cope with a new ecosystem.

— CMake - I had to write my own CMake files because I was linking to Rust

— vcpkg - for C++ libraries

— Visual Studio & msvc flag conventions

Redmond is a different developer country. I've had an adjustment period here to the JVM, which is a different ecosystem with its own local customs and trivia, but the adjustment to Redmondland was bigger.

Command-line workflows

— developer tools are better on MacOS & Linux (familiarity bias?)
— found a Windows terminal app that isn't awful (wezterm)
— WSL saved me in the end, because I had bash
— forced back to VSCode instead of my preferred editor
— all the Rust dev can be done on MacOS
— implemented Rust tests to avoid needing to link to CLib

It was a pretty bad adjustment, to be honest. I really loathe editing code in Visual Studio, which is amazing to me because of how many people use these tools worldwide.

Strings are (often) annoying

— Rust is natively `utf-8`; `String` type and `&str` borrowed data type
— `std::ffi::{OsString, OsStr}` are host OS equivalents
— Windows OS strings are wide chars (16 bytes); file paths!
— Skyrim uses `iso-8859-1` internally
— Scaleform/Flash translation strings are `UCS-2`
— player crashes with encodings until I learned all this

String handling in lower-level languages are traditionally tricky because you have to think about ownership of the byte structure behind the string. That was not a problem in this project, but encodings were. UCS-2 is a fixed-width 16 byte encoding that is pretty old-- I had to cope. This is an example of how unfamiliarity with Windows presented me with a learning curve. It's trivia I had to pack into my brain.

Modern C++ > older C++

— all mod cons in the std library
— ergonomics are better
— official documentation is terrible
— null-pointer access still plagued me
— no contract about when a pointer can be null in CLib

Better but not great. 90% of my crashes were null pointers. The remainder was a combo of invalid utf-8 in Rust plus a couple of stray `todo!()` macros + one unwise `unwrap()`.

So how **is** the Rust/C++ **interop** story?

Exactly as good as advertised.

— near zero cost

— tools are mature and well-tested

— bindgen is one approach

— cxx was the one I used

cxx codegen for a safer subset of interop

— safe in the Rust sense: no games with memory; compiler can check

— generates bindings for the functions you want on either side

— C++ can call Rust & vice versa

— not as complete as `bindgen`

— `char*` not supported, for example (impossible to make safe)

— but great interop for utf8 strings and vectors

lib.rs defines the bridge

Let's look at some code briefly!

show lib.rs here

plugins could be 100% Rust

— currently 68.5% Rust
— official Rust bindings for nearly all Windows APIs
— **bindgen** is the path here
— would need to learn xbyak
— CLib is a moving target
— I got a job instead of diving into this
— gotta have a budget for graphics cards, you know?

Theoretically, I could reduce the C++ down to Rust bindings to CommonLibSSE-NG and the SKSE plugin entry point, plus some xbyak glue. Imgui has Rust bindings. DX11 has Rust bindings, so I can create textures for it. The way I chose was the least work-- I could focus on the application itself.

reworks along the way

I found myself having persistent trouble with some design decisions.

— made some messes initially with cycle advancing

— careful work documenting how next-weapon decisions are made

— could handle grip-switch mods easily once I got this right

— loading and rasterizing svgs

— categorizing the many, many mod-added items

The actual task of advancing cycles with edge case handling was a lot of careful work. I remember what is in each hand so advancing either hand and going from 2-hander to 1-hander restores what was in the other hand. This feels right in-game, but has lots of edge cases.

nanosvg → resvg

— C++ to <u>Rust</u>

— but really, minimal spec support to extensive spec support

— implemented lazy-loading at the same time

— no real memory pressure, but I like being thrifty

People kept reporting bugs with svgs that worked in Flash but didn't in Soulsy. It was always because the svg was actually a PNG in a base64 trenchcoat, which is allowed by the spec but not supported by nanosvg.

object categorization rework

— mod-added objects weren't getting the right icons

— both the C++ and Rust sides of this were awful

— so I started using object keywords to categorize

— the mod <u>Object Categorization Framework</u> adds keywords to thousands of items

— for a few mods, I distribute keywords myself using <u>Keyword Item Distributor</u>

keep it fast

Two facets of performance:

— perceived input latency
— frame rate cost of the render loop

Using a native-code SKSE plugin means doing both is possible, though bad programming can mess it up.

Writing this as a native-code SKSE plugin obliterated the input latency problem. Imgui means it can be visually fast. This is why I started with the HUD mod I picked.

the render loop has to be tight

— reduced branching in the tight loop
— made all layout decisions at layout load time where possible
— internal structures are quite different from the serialized player-editable structures as a result
— mod precalculates and loads data for items that will be drawn

This was a lot of fun to do. Not the kind of programming I get a chance to do much of these days.

Supporting cli projects

I wrote Rust cli tools to help:

— `md2nexus`: convert markdown to NexusMod's BBCode (which is awful)

— `mcm-meta-helper`: manage translation string files; UCS-2 is hard to work with

— `priority-sync`: sync mod order across MO2 profiles; for testing with many setups to repro player problems

All of these are Rust cli tools. The third one is notable because there is a python plugin for Mod Organizer 2 that does this, but it's extremely slow and buggy. I wrote a ridiculously fast tool that does something simpler but easier to make reliable. All of these are in the `ceejbot` github, btw, if you want to see how nice writing CLIs in Rust is, thanks to the `clap` crate.

leveled up as a Rust programmer

— extremely comfortable with the language now

— using the type system in more advanced ways (generic associated types, e.g.)

— appreciating more how type systems improve my data design thinking

— thinking about lifetimes improved my C++

the features I wanted

— shouts & powers cycle

— left item cycle: shields, torches, weapons, spells, scrolls

— right item cycle: weapons, spells, two-handers, scrolls

— potion & poison cycle

— a layout defined in a human-readable text file

— minimal to no settings

the features I implemented

— all of what I wanted! plus!

— too many settings

— customizable layouts in TOML, with hot reload in-game

— two custom icon sets

— smart potion selection

— equipment set management

— full controller support

got to use many skills

— product design
— UX design & writing
— user docs + programmer docs
— programming with attention to perf & memory use
— single-process stateful software architecture (not dist systems for once)

I have a very particular set of skils, skils I have acquired over a very long career. Skills that make projects like this a joy for me. I am not a graphic designer, but I faked it here. Visual transitions use easing curves; I did my best to polish everything I touched.

Incredibly fulfilling work

— Full control of design but took lots of player input.

— Heard player pain immediately.

— Heard player happiness immediately!

— Could have zero tolerance for tech debt.

— Could meet my own quality standards.

— Feature development was extremely rapid when I got the design right.

— Got exactly the HUD mod I wanted.

My own standards for software are much higher than that of most of the startups I've worked for. Most often the ticking clock of company debt means that speed is the only thing that matters and even that is evaluated in a very shallow way. Here I could realize that I'd designed myself into a corner that made feature work hard, and then fix it by rewriting. In the end, this is of course faster than not fixing tech debt.

What I learned

— so much! Skyrim, modern C++, that I can still learn fast when I need to

— Rust > C++ (but they go well together)

— Windows needs a great text editor

— Skyrim is a platform for focused game dev

— Skyrim did in fact fix my bad mood

Questions?

github.com/ceejbot/soulsy