FalconEngine

PHYSICAL VEHICLE SYSTEM FOR SKYRIM

DEVELOPER DEEP DIVE: FILE & LOGIC BREAKDOWN

The "Physical Bubble" Concept

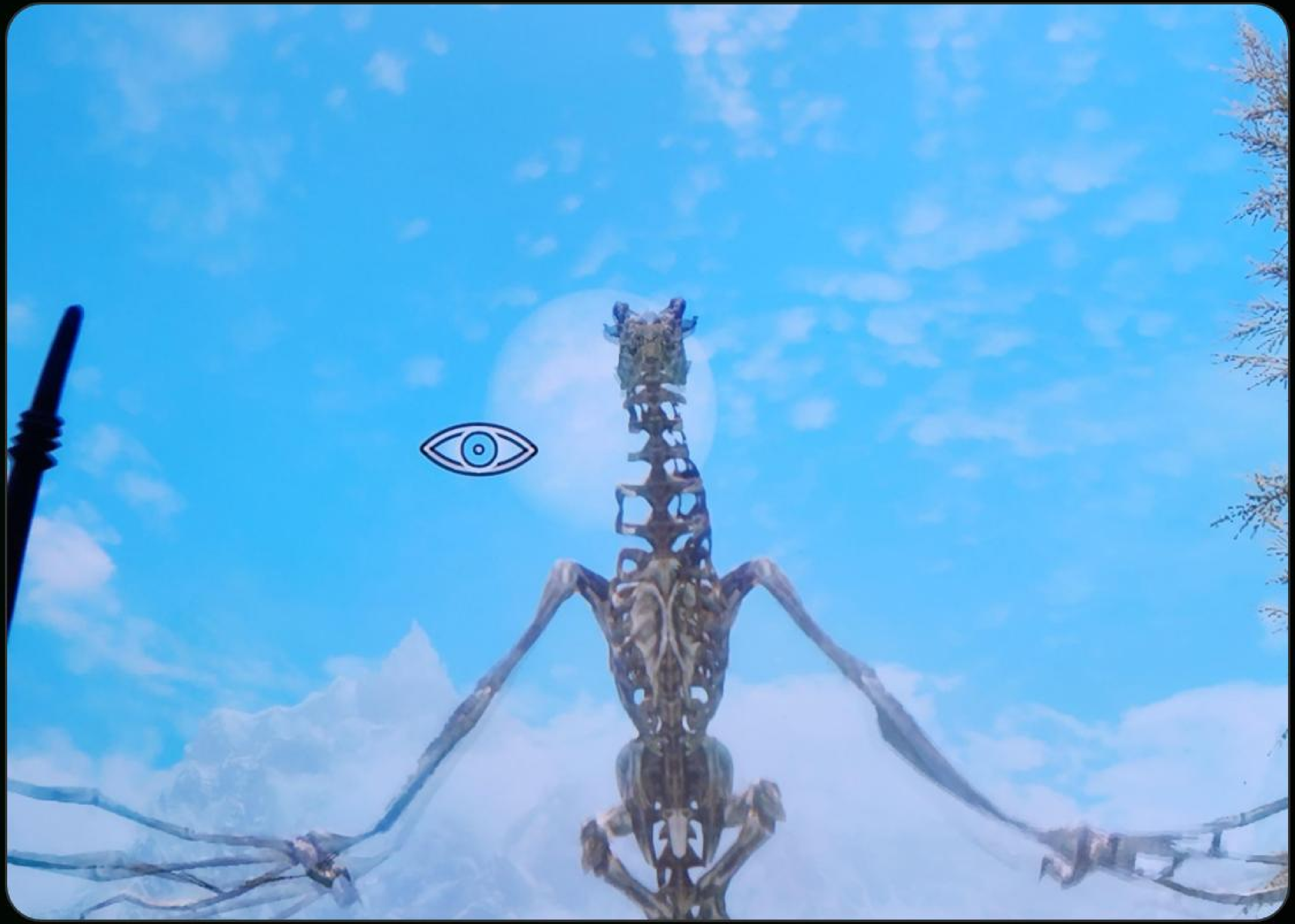
Skyrim's engine was never built for moving platforms. When an object moves, the actors on top usually slip off. I designed **FalconEngine** to create a synchronized coordinate space.

Framework

Built on CommonLibSSE-NG to ensure cross-version compatibility between SE and AE.

The Glue

My code uses a "Hook" into the game's heartbeat to update positions exactly 60+ times per second.



Main.cpp: The Entry Point

SKSEPluginLoad

This is the first line the game executes. I use `Init(a_skse)` to hand over control to my plugin and start the logging system.

AllocTrampoline

I allocate **14 bytes** of executable memory. This is crucial for my "jump" into the game code without crashing the executable.

kDataLoaded

I register a listener that waits until all ESM/ESP files are finished loading before I attempt to find the dragon in memory.

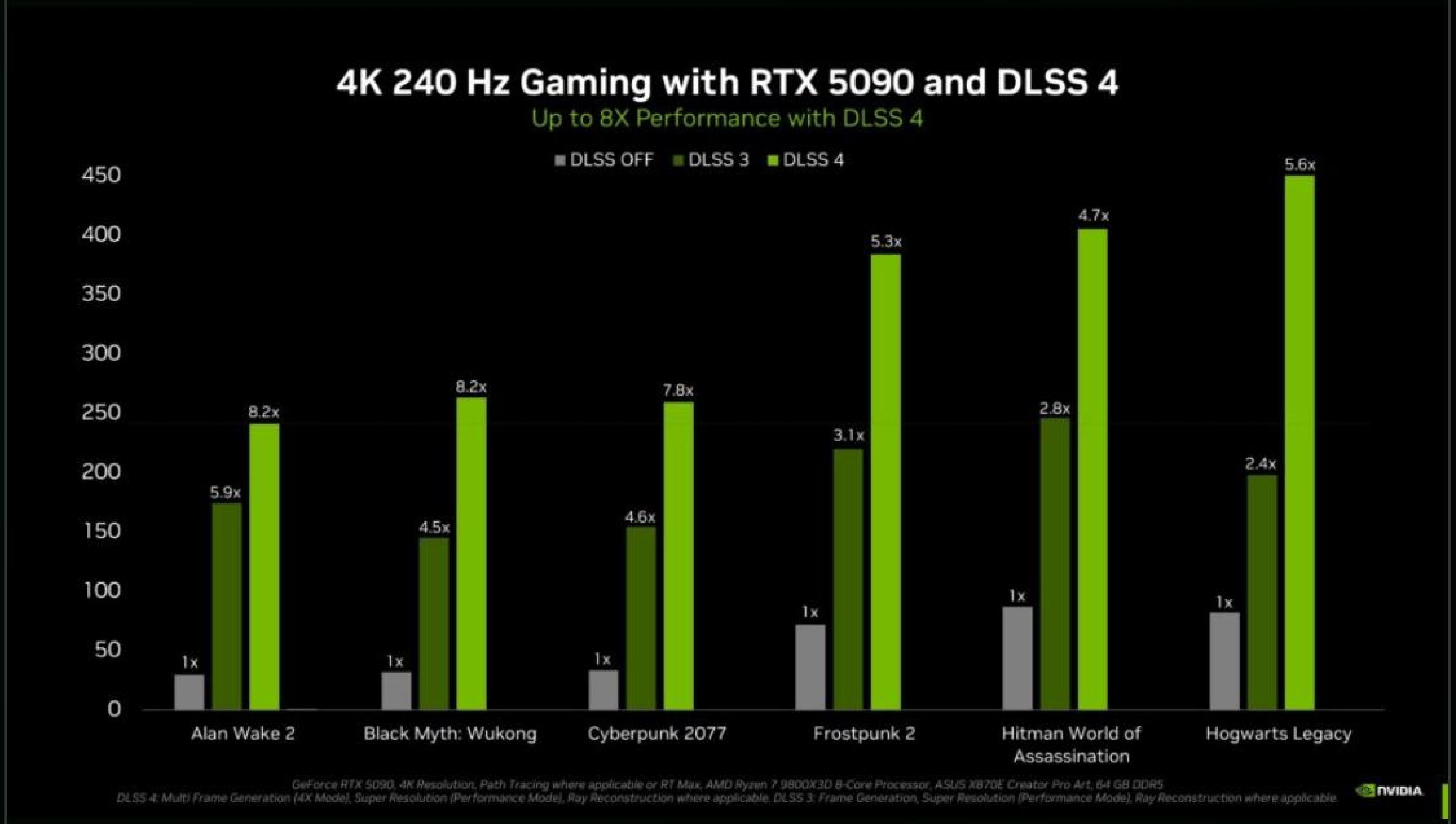
src/Main.cpp

```
// Plugin Lifecycle
SKSEPluginLoad(const LoadInterface* a_skse) { ... GetMessagingInterface()->RegisterListener(...); }
```

Hooks.cpp: The Surgeon

This is where I perform the most dangerous part: overwriting Skyrim's own memory. I use the **Relocation** system to find the Main Loop.

- `REL::ID(35551)` : I target the specific address of `BSMain::Update` .
- `write_call<5>` : I overwrite 5 bytes to force the game to call my function every single frame.
- `_Update()` : I call the original game code first so I don't break standard physics.



GeForce RTX 5090, 4K Resolution, Path Tracing where applicable or RT Max, AMD Ryzen 7 9800X3D 8-Core Processor, ASUS X870E Creator Pro Art, 64 GB DDR5
DLSS 4: Multi Frame Generation (4X Mode), Super Resolution (Performance Mode), Ray Reconstruction where applicable. DLSS 3: Frame Generation, Super Resolution (Performance Mode), Ray Reconstruction where applicable.

MovementHandler.cpp: The Math

3x3

Rotation Matrix

Local to World Transformation

To keep you on the skeleton, I perform 3D matrix multiplication every frame.

```
RE::NiPoint3 worldPos = shipPos + (shipRot * localPos);
```

I take your **Local Position** (relative to the dragon), rotate it by the dragon's **Rotation Matrix**, and then add the dragon's **World Position**.

AIPathing.cpp: Crew Persistence

I designed this to handle multiple NPCs simultaneously using an **ObjectRefHandle** map. This ensures if an NPC unloads, the game doesn't crash.

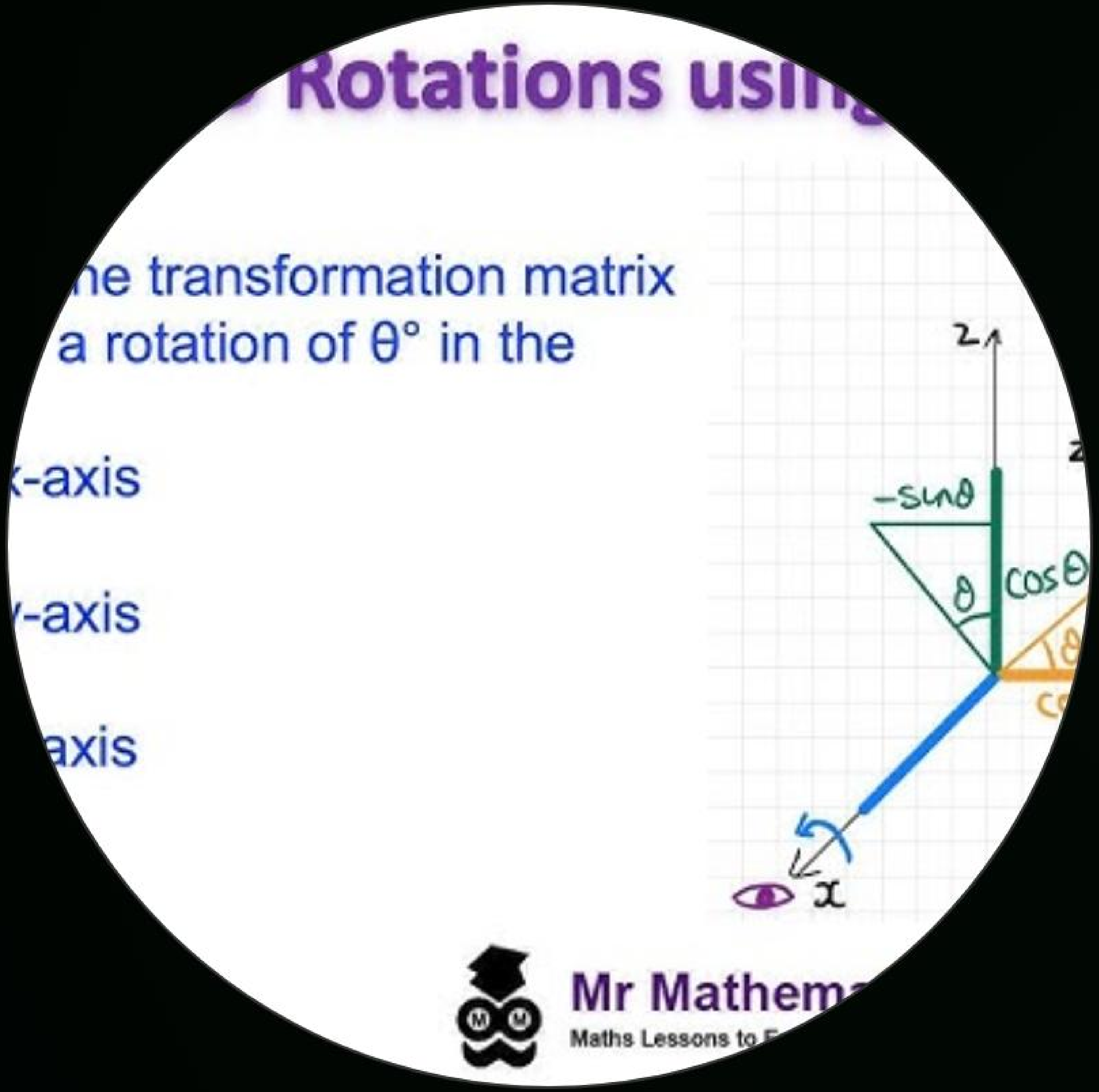
- **it->first.get()**: I attempt to resolve the handle. If it returns null, the NPC is gone.
- **IsDisabled()**: I check if the skeleton or crew is disabled before processing math.
- **Transpose()**: I use the inverse rotation to calculate where an NPC "should" be relative to the ship.
- **SetPosition(pos, true)**: The "True" flag forces an instant warp, preventing jitter.

Papyrus.cpp: The Script Bridge

I exposed three key functions to the Skyrim script engine so I can control the physics from inside the game world.

Function Name	Input Argument	C++ Internal Action
`SetShip`	ObjectReference	Converts Reference to FormID and caches it in `cachedShip` .
`AddCrew`	Actor	Calculates local offset and stores it in the Handle Map.
`ClearCrew`	None	Safely purges all memory associated with the crew list.

InputHandler.cpp: Interaction



Rotations usi

he transformation matrix
a rotation of θ° in the

-axis

-axis

axis

Intercepting Controls

I don't let Skyrim move the player naturally while riding. I use `ControlMap` to check inputs.

Instead of `Player→Move()`, I increment `localPos.y`. This effectively moves your "virtual seat" on the dragon. My MovementHandler then teleports you to that spot in the world space.

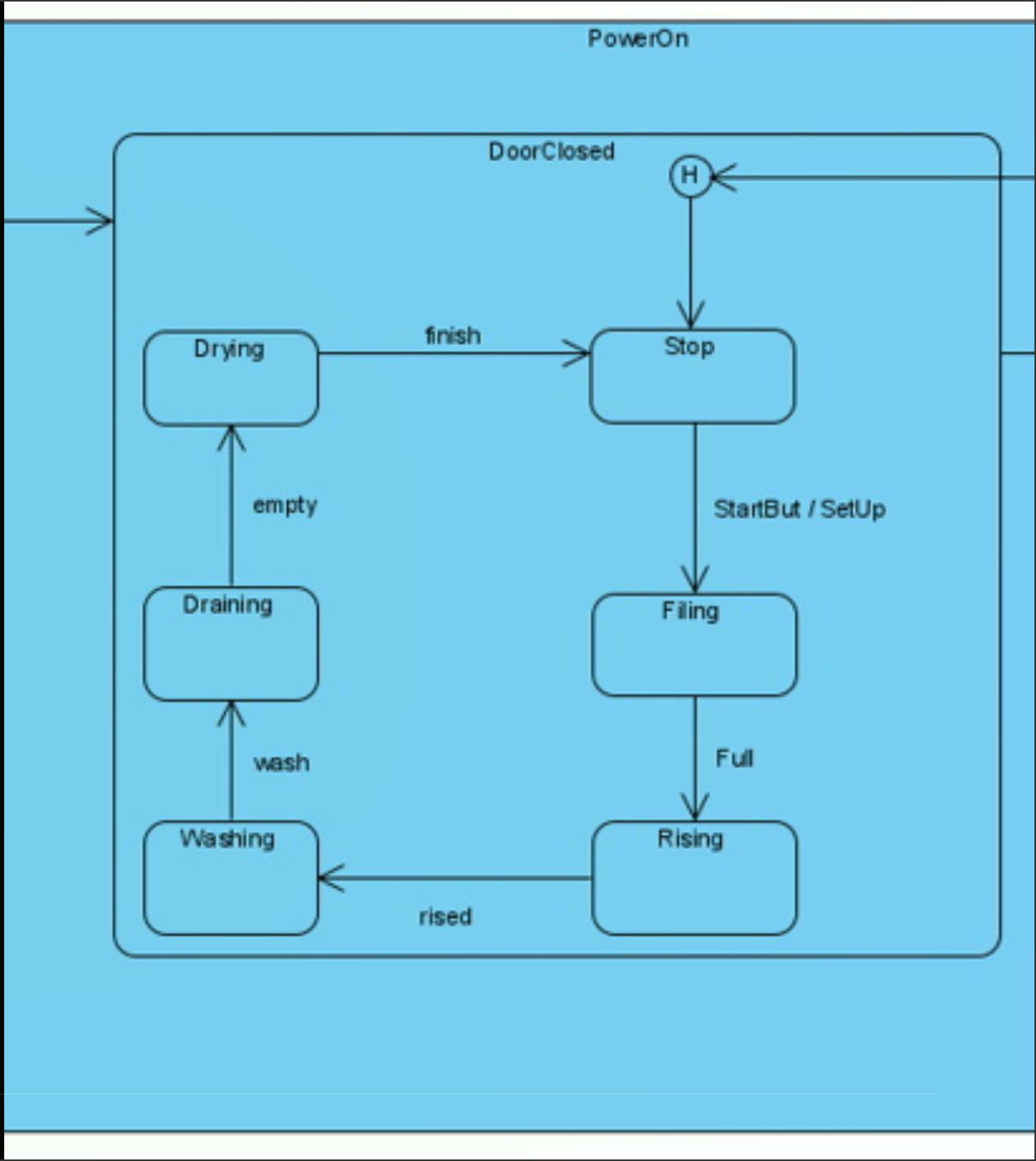
The Blueprint

CMakeLists.txt

I configured the build system to automatically find CommonLibSSE-NG and link it as a **Private** dependency.

This file ensures that my DLL is compiled with the `MultiThreadedDLL` runtime, preventing "Mismatch" errors when loading into Skyrim's executable environment.

I also set the **C++ Standard to 23** to utilize the latest compiler optimizations for my 3D math.



Dependency Management

vcpkg.json

I use this to manage the CommonLib package. It ensures every time I build, the versions match perfectly.

Version Baseline

I locked the baseline to a specific commit hash to prevent my mod from breaking when the libraries update.

Skyrim AE Feature

I explicitly enabled the `skyrim-ae` feature flag in the JSON to ensure the offsets are correct for modern versions.

Memory Safety

By using `RE::ObjectRefHandle` and `NiPointer`, I've created an engine that is virtually immune to the "Unload CTD" that plagues many other physics mods.

Deployment Ready

The FalconEngine framework is now complete and modular.

github.com/FalconEngine-Mod

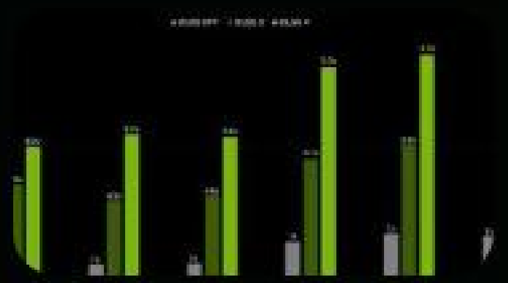
Skyrim SE/AE Plugin

Questions?

Image Sources

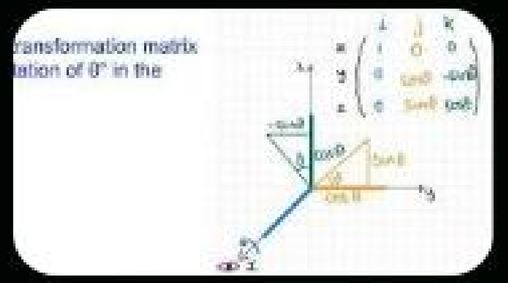
https://i.redd.it/um-skeleton-dragon-flying-and-attacking-me-no-mods-on-never-v0-8zx8jxglopw91.jpg?width=3000&format=pjpg&auto=webp&s=0eaeb09657a4e816033df06a470d3edb0724e270

Source: www.reddit.com



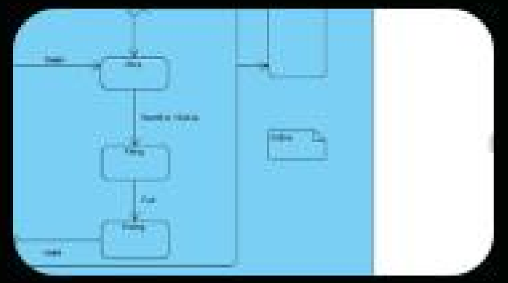
https://builttofrag.com/wp-content/uploads/2025/08/up-to-8x-faster-performance-1024x576.jpg

Source: builttofrag.com



https://i.ytimg.com/vi/4mqjvYRTRy8/hq720.jpg?sqp=-oaymwEhCK4FEIIDSFryq4qpAxMIARUAAAAAGAElAADIQj0AgKJD&rs=AOn4CLBkeC2eYSfLXxAX9PhjGq7R8HSNIQ

Source: www.youtube.com



https://www.incredibuild.com/wp-content/uploads/2020/12/Architecture-Diagrams-C_7.jpg

Source: www.incredibuild.com