Nebula engine

CI passing

Get in contact on Discord! https://discord.gg/wuYPxUF

Check out the documentation (WIP) here: https://gscept.github.io/nebula-doc/

Requirements

1. OS: Windows and Linux(WIP)
2. Compiler with support for C++20.
3. GPU and drivers supporting Vulkan 1.2+
4. CMake 3.21+
5. Python 3.5+

    - Python requirements (Windows):
        1. Matching architecture (64-bit if you're building for 64-bit systems)
        2. Installed for all users
        3. Added to PATH
        4. Installed with debugging symbols and binaries

Setup

There are a few simple projects in the test folder, but a more interesting project is available as well Nebula-demo. Fips will take care of getting the relevant dependencies (including the actual engine in case you are setting up a external project)

Setup config

1. `fips set config win64-vstudio-debug` in your project directory, e.g. the cloned nebula-demo. There are other configs available, see in fips-files/configs
2. `fips fetch` Downloads/checks out all the other required repositories

Select the project and source directory Run `fips nebula` verb to set work and toolkit directory registry variables:

- `fips nebula set work {PATH}` (If you are building an external project, this would be the current path)

- `fips nebula set toolkit {PATH}` (this is the path to where the nebula checkout resides, if an external project that would be ../nebula)

Build project In your project directory:

1. `fips physx build vc17 debug` (if you are running VS 2022, use `vc16` or `vc15` for vs 2019/2017 instead)
2. `fips anyfx setup`
3. `fips ultralight`
4. `fips gen` to generate the required build system files, e.g. a visual studio solution
5. `fips build` to directly compile the project or `fips open` to open the generated solution in your selected environment

Features

Nebula is being developed continuously, which means that features keep getting added all the time. Currently, we support this:

- Completely data-driven design from bottom to top.
- Data structure suite, from containers to OS wrappers, everything is designed for performance and minimal call stacks.
- Multithreading.
- SSE-accelerated and intuitive maths library.
- Full python supported scripting layer.
- Advanced rendering framework and shaders.
- Test-benches and benchmarking.
- Profiling tools.

Rendering A lot of effort has been made to the Nebula rendering subsystem, where we currently support:

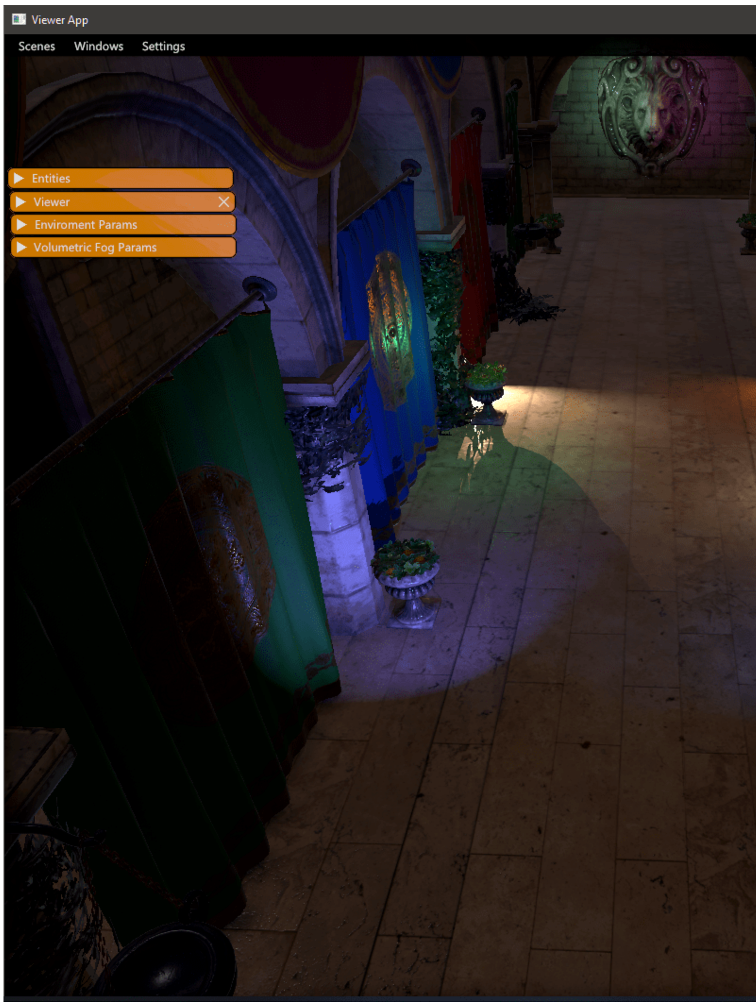
- Unified clustering system - fog volumes, decals and lights all go into the same structure.
- Screen-space reflections - working condition, but still work in progress.
- Horizon-based ambient occlusion done in compute.
- Physically based materials and rendering.
- Multi-threaded subpass recording.
- Shadow mapping for local lights and CSM for global/directional/sun light.
- Volumetric fog and lighting.

- Geometric decals.
- CPU-GPU hybrid particle system.
- Skinning and animation.
- Scripted rendering path.
- Vulkan.
- Tonemapping.
- Asynchronous compute.
- Virtual texturing using sparse binding.
- Adaptive virtual textured terrain.
- Fast and conservative GPU memory allocation.
- Area lights.

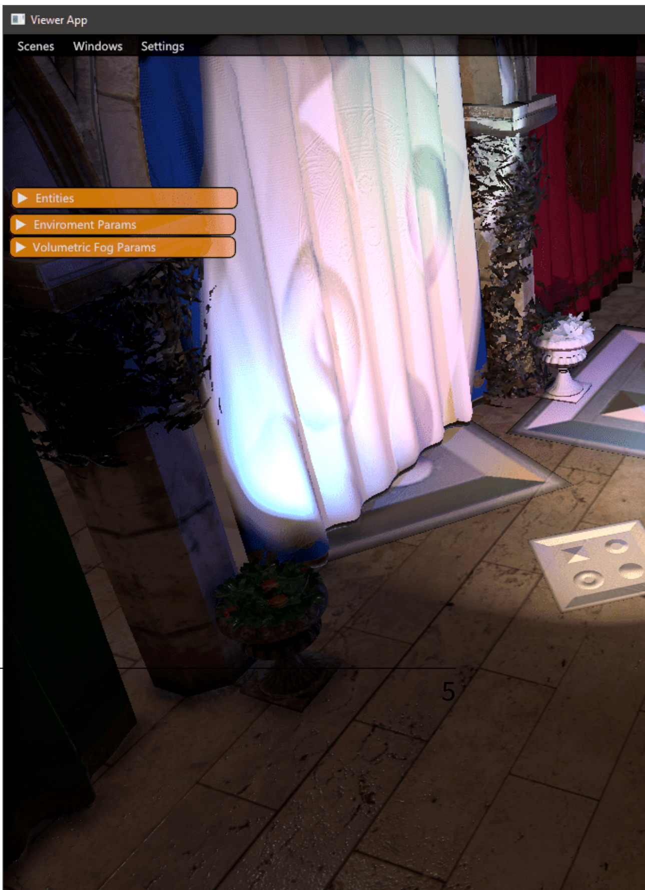
Entity system Nebula has historically had a database-centric approach to entities. With the newest iteration of Nebula, we've decided to keep improving by adopting an ECS approach, still keeping it database-centric.

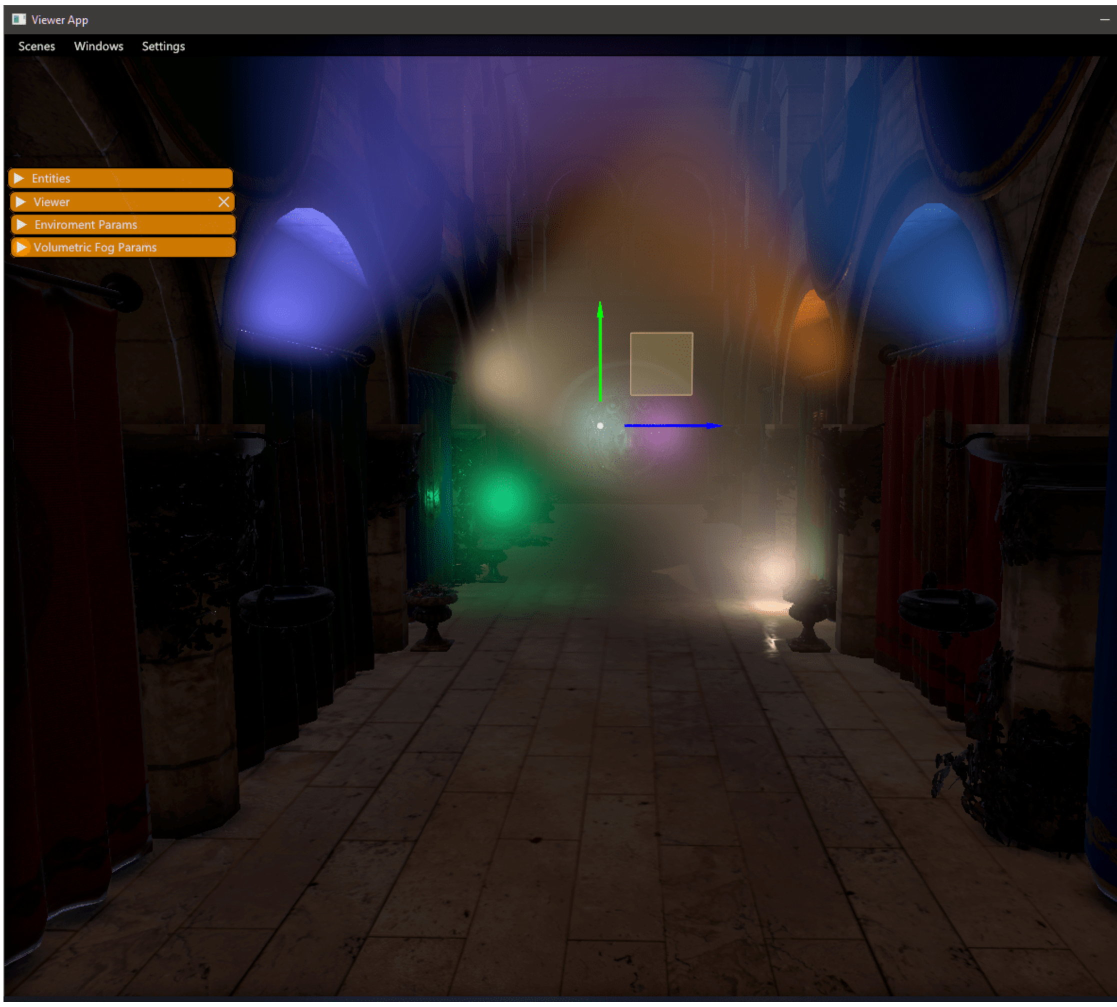
- Data-oriented
- Data-driven
- Minimal memory overhead per entity.
- High performance without compromising usability or simplicity
- Blueprint and template system for easily instantiating and categorizing entity types.
- Automatic serialization and deserialization

Screenshots

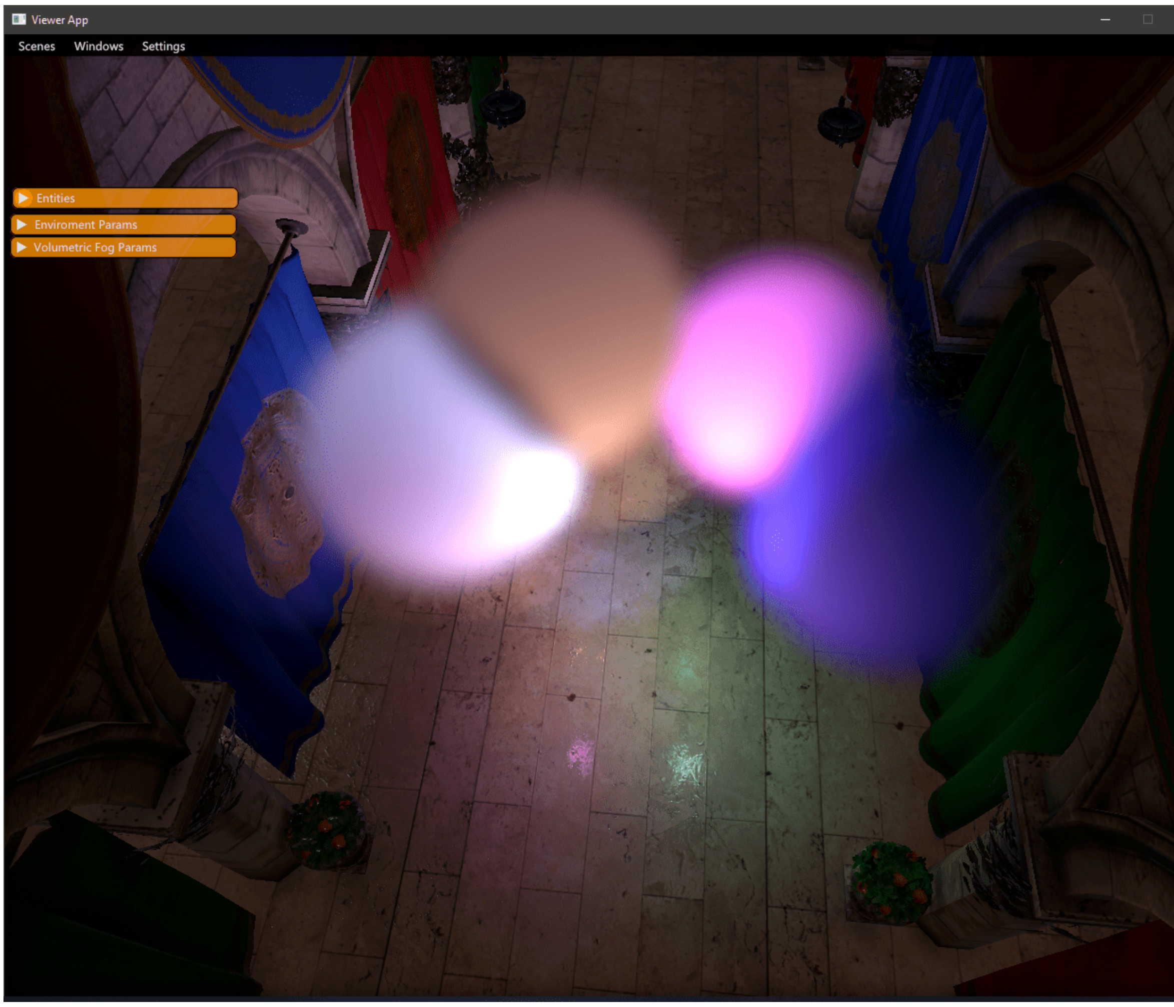


Deferred Lighting using 3D clustering and GPU culling.

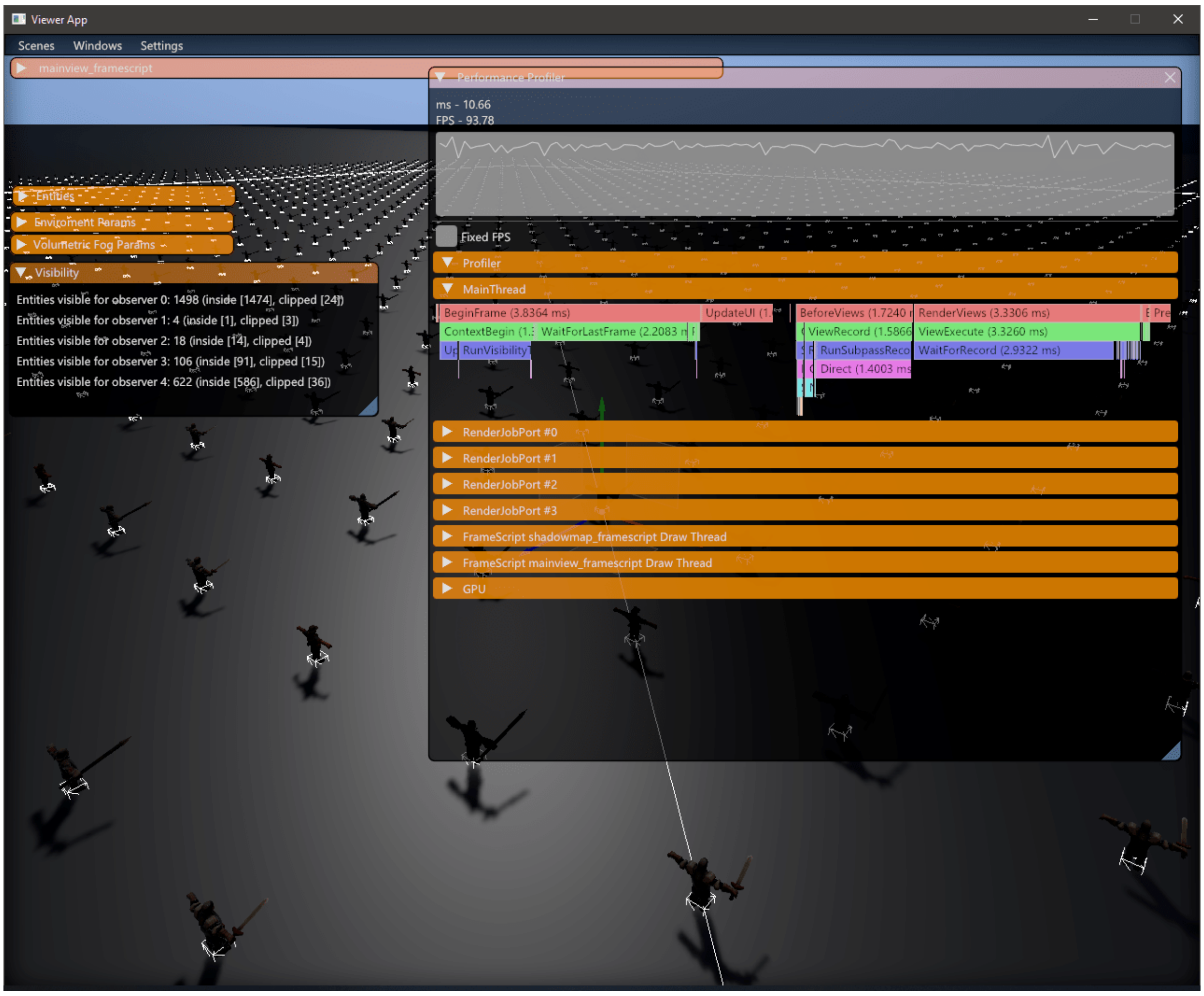




Volumetric fog lighting.



Local fog volumes.



```
Entities visible for observer 0: 1498 (inside [1474], clipped [24])
Entities visible for observer 1: 4 (inside [1], clipped [3])
Entities visible for observer 2: 18 (inside [14], clipped [4])
Entities visible for observer 3: 106 (inside [91], clipped [15])
Entities visible for observer 4: 622 (inside [586], clipped [36])
```

Profiling tools.