https://ncine.github.io

14 Years of Developing nCine

An Open-Source 2D Game Framework

Angelo "encelo" Theodorou

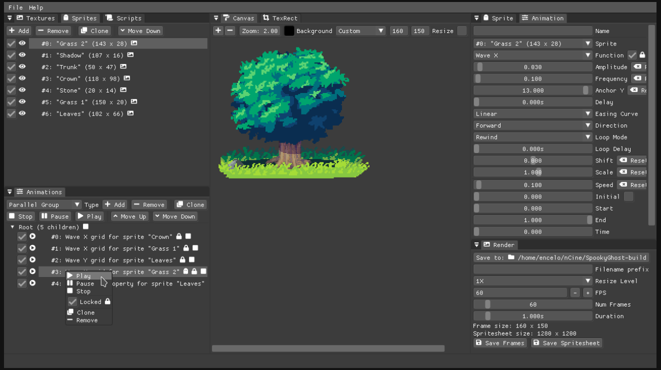
/dev/games, Rome, June 5-6, 2025

What Is the nCine? ⚙️

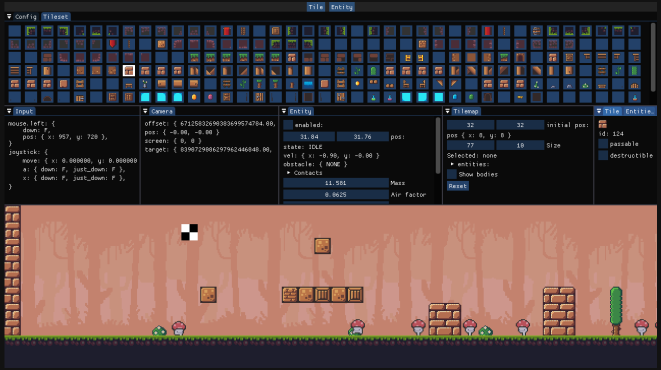
- A portmanteau of "Encelo" and "engine" 😄
- A cross-platform 2D framework for games, tools, and prototypes
  - Supports PC (Linux, Windows, macOS), Android, Raspberry Pi, and the web (Emscripten)
  - Provided as a static/dynamic library with clean API and callbacks
- Written in C++11, with Lua bindings
- Strong emphasis on performance and optimization
- Source available on GitHub under the MIT license
- Based on a transformation scene graph and a node hierachy (no components)
- A *learning opportunity*, both for me and for its users

📸 A Selection of Screenshots

- A few examples of what has been built with or on top of the nCine



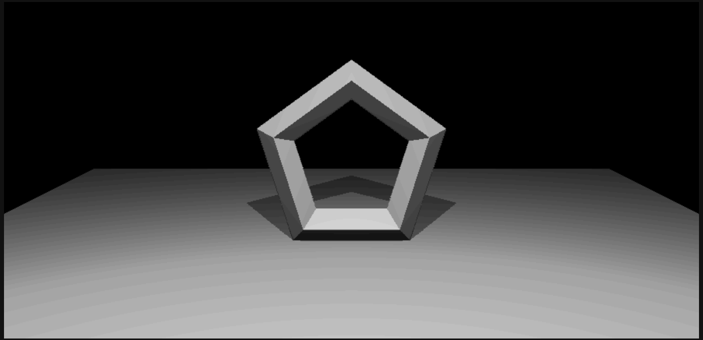
SpookyGhost, *a procedural sprite animation tool*



ncJump, *a platform game with Box2D physics by Fahien*

Why Yet Another Custom Engine? 🤔

- Job *frustration* was the original, powerful source of motivation
  - Today, I keep myself motivated by working on it a bit every day
- I wanted to *learn*, to become a better programmer, to reach my career goals
  - Tackle low-level topics like C++ templates, custom allocators, multi-threading
  - Build a strong codebase for my future 3D learning demos (and ditch old frameworks)
- Today, I want to see *others* using it to build interesting projects
  - I would feel immensely proud if people created something beautiful with it
- We need more custom engines to fight *innovation stagnation*!
  - nCine: a world with custom in-house engines is possible



Unreleased ***ncShadowMapping*** *demo*

My custom engines presentation on a faulty projector

Lines of Code

```
/───────────────────────────────────────────────────────────────────────────────\
│Language                     files          blank        comment           code│
├───────────────────────────────────────────────────────────────────────────────┤
│C++                            378          17127           3298          83956│
│C/C++ Header                   343           7422           4277          30650│
│CMake                           44            716            387           5676│
│YAML                             7            176             27            923│
│XML                              2              0              1            660│
│Lua                             10            137              5            600│
│GLSL                            21             67              0            340│
│Markdown                         1             18              0             87│
│Gradle                           2              1              0             23│
│INI                              1              2              0             10│
├───────────────────────────────────────────────────────────────────────────────┤
│SUM:                           809          25666           7995         122925│
\───────────────────────────────────────────────────────────────────────────────/
```

The main ***nCine*** *repository, counting only the* `master` *branch and excluding external projects as of June 4, 2025.*

Over 25,000 lines are dedicated to unit tests. 😱

How It All Began

- I was born in 1983 and got an Amiga 500 in 1991 👴
- The Amiga had amazing games, but also a vibrant *demoscene*! ❤️
  - The pursuit of beauty and wonder through mastery of the machine 🧙‍♂️
- In 2000, I jumped on the open source and *nix train (still on the Amiga) 🚂

That's me in the early '90s, playing on my first Amiga

🗓️ 2011 - Once Upon a Time

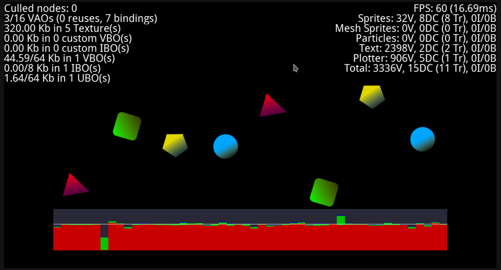
- In 2010 I joined a small indie company in Italy 🇮🇹
- I daydreamed about graphics and engine programming…
  - …but what I actually did all day was GUI work for games 😟
- I took destiny into my own hands! 💪
  - First "*encine2d*" commit on 19 June 2011 🗓️ (-○- 6bf318de)
  - Coming from CVS and SVN, I initially chose Mercurial and hosted on BitBucket
- I deliberately neglected rendering to focus on everything else that makes an "engine"
- I wrote "apptests" to stress-test the API as it evolved

📸 2011 - Small Screen, Big Dreams

My working environment in 2011: an 11.6" Lenovo IdeaPad S205

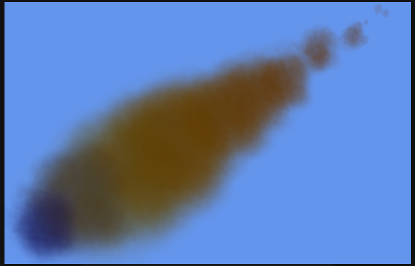
📅 2011 - Laying The Foundations

- Developed exclusively on Arch Linux using Qt Creator, CMake and GCC
- Doxygen documentation generated automatically from inline code comments
- Android support added just one month after the project started 📱
- Performance oriented from day one:
    - XNA-inspired explicit sprite batcher: `Begin()` / `Draw()` / `End()`
    - Texture atlas support to reduce OpenGL binding calls and optimize batching
    - Template-based custom array and list containers
    - Line and stacked histogram plotters for detailed time profiling
    - Support for block-compressed GPU texture formats (ETC1 in pkm, ATITC in DDS)
- Scenegraph implementation with sprites, particle systems, and text nodes (using bitmap fonts)



```
Culled nodes: 0                                   FPS: 60 (16.69ms)
3/16 VAOs (0 reuses, 7 bindings)      Sprites: 32V, 8DC (8 Tr), 0I/0B
320.00 Kb in 5 Texture(s)        Mesh Sprites: 0V, 0DC (0 Tr), 0I/0B
0.00 Kb in 0 custom VBO(s)          Particles: 0V, 0DC (0 Tr), 0I/0B
0.00 Kb in 0 custom IBO(s)           Text: 2398V, 2DC (2 Tr), 0I/0B
44.59/64 Kb in 1 VBO(s)            Plotter: 906V, 5DC (1 Tr), 0I/0B
0.00/8 Kb in 1 IBO(s)            Total: 3336V, 15DC (11 Tr), 0I/0B
1.64/64 Kb in 1 UBO(s)
```

Original histogram plots



`apptest_particles`

Droid Sans bitmap font

📅 2012 - Expanding Horizons

- I moved to Cambridge to work for ARM 🇬🇧 (📅 Dec)
  - nCine was still in early development, but caught attention during my interview
  - Asked and got approval from my manager to release it open source in the future
  - Developer relations with Epic Games, Unity, Frostbite, Gameloft, and more
  - Presented at GDC, Unite, GameLab, and more
- Added an OpenAL based sound system 🔊 (-○- b8c23c54, 📅 Feb)
  - Support for audio buffers (WAV) for effects and streams (Ogg Vorbis) for music playback
- Implemented a new file interface to support Android assets
- Added PNG and WebP texture support using libpng and libwebp
- Introduced GLFW as an alternative to the SDL1 desktop backend
- Designed a threading and synchronization API with POSIX and WinAPI primitives

🗓️ 2013 - Android as a Console

- I always wanted to be a console programmer
- Android was considered a console-like target for the nCine
  - Working at ARM surrounded me with Android devices
  - Set-top boxes running Android TV were becoming popular
  - I received a Google ADT-1 as a gift at Unite 2014 in Seattle 🎁

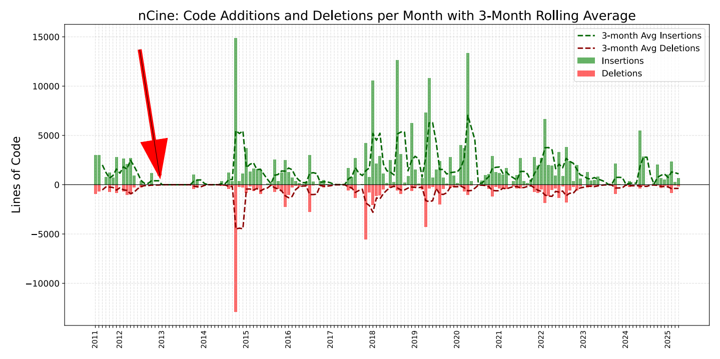
OUYA (2013)

Google ADT-1 (2014)

Nvidia Shield (2015)

📅 2013 - Quiet Progress

- Implemented high-precision monotonic clock (POSIX, Mach, Windows) (419c68f1, 📅 Oct)
  - Unaffected by system time changes, it increments a counter continuously from system boot
- Added MIP mapping support to improve rendering quality and performance
  - Especially important on mobile devices to reduce GPU memory bandwidth and save battery 🔋
- Updated GLFW desktop backend to GLFW 3.0
  - System backends abstract the specifics and provide a generic API for window handling and inputs
  - Backend API proved flexible, enabling ports for SDL2, GLFW3, Qt5, and Android
  - A user has made ports for UWP (Xbox) and homebrew Switch without issues



Additions/Deletions Over Time

💊 2013 - Monotonic Clocks

```cpp
#if defined(_WIN32)
    if (hasPerfCounter_) QueryPerformanceCounter(reinterpret_cast<LARGE_INTEGER *>(&counter));
    else counter = GetTickCount();
#elif defined(__APPLE__)
    #if __MAC_10_12
    counter = clock_gettime_nsec_np(CLOCK_MONOTONIC_RAW);
    #else
    counter = mach_absolute_time();
    #endif
#else
    if (hasMonotonicClock_)
    {
        struct timespec now;
        clock_gettime(CLOCK_MONOTONIC, &now);
        counter = static_cast<uint64_t>(now.tv_sec) * frequency_ + static_cast<uint64_t>(now.tv_nsec);
    }
    else
    {
        struct timeval now;
        gettimeofday(&now, nullptr);
        counter = static_cast<uint64_t>(now.tv_sec) * frequency_ + static_cast<uint64_t>(now.tv_usec);
    }
#endif
```

Three different backends, each one with a fallback (`src/base/Clock.cpp` 🔗)

🗓️ 2014 - Gamepads Support

- Gamepads were essential for a console-like experience (⑂ `joysticks`, -o- 11d7799a, 🗓️ Aug - Sep)
  - On Android, I had to use JNI to call Java from C++, ouch 🤕
- Stylistic coherence with Artistic Style (⑂ `codestyle`, -o- d111d9c1, 🗓️ Oct - Dec)
  - Moved private headers to `src/include` so only public API headers remain in `include`
  - Removed the `nc` prefix from class names in favour of the `ncine` namespace
  - Kept a prefix for interfaces (abstract classes): `IAppEventHandler`
- Custom string class with iterator
- Templated static array class with iterator
  - Uses stack storage, capacity is fixed (`template <class T, unsigned int C>`)

📅 2015 - From Hobby to Interview Material

- I moved to Oxford to work as an Android Technology Programmer for Natural Motion (📅 Jun)
  - I showed some nCine code during my interview to demonstrate my C++ skills
  - The nCine was still not ready for showtime, yet the company was supportive about its release
  - I worked on the custom engine of Clumsy Ninja and Dawn of Titans
- Merged the OpenGL 2 ⎇ `new_renderer` branch (⊸ 6e8070f3, 📅 Aug)
- Added algorithms for containers and refactored iterators
- Shaders can be embedded in the source as `char` arrays generated by CMake (📅 Dec)

💊 2015 - OpenGL 2 Renderer

- Moved away from fixed pipeline
  - Added vector, matrix, and quaternion classes
  - Added more OpenGL wrappers (buffer objects, FBOs, render buffers, shader programs, textures)
  - Introduced classes to handle shader attributes and uniforms (cached in a new hashmap container)
  - Implemented a general render command with new material and geometry classes

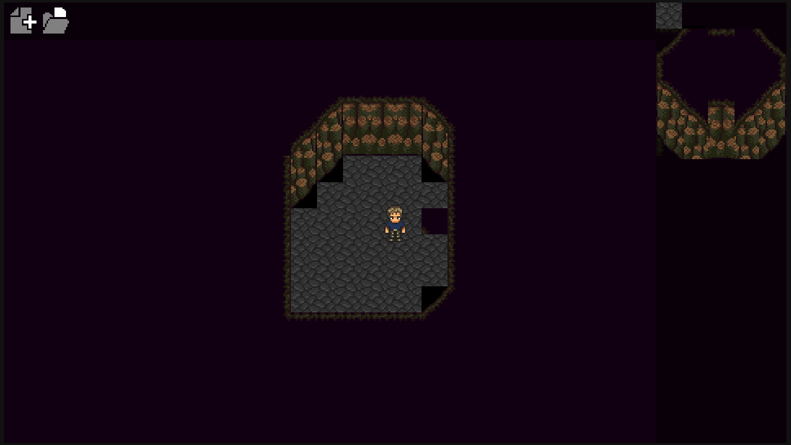
```cpp
void RenderCommand::issue()
{
    geometry_.bind();

    material_.bind();
    setTransformation();
    material_.commitUniforms();
    material_.defineVertexPointers(geometry_.vboHandle());

    draw();
}
```

Binding the geometry and material before issuing the draw call (`src/graphics/RenderCommand.cpp` 🔗*)*

📅 2016 - Slightly More Public

- Created the GitHub organization, the website, and the Discord server (📅 Jun)
  - Initially distributing the library as a binary on GDrive 😄
  - Fahien was the first to jump on board with ncRogue
- Added support for MinGW/MSYS2 (📅 Mar)
- `ncPong` is the first official nCine project, a Pong clone (📅 May)
- CMake scripts to build dependency libraries for all platforms (📅 May)
  - `nCine-libraries`, `nCine-android-libraries` repositories

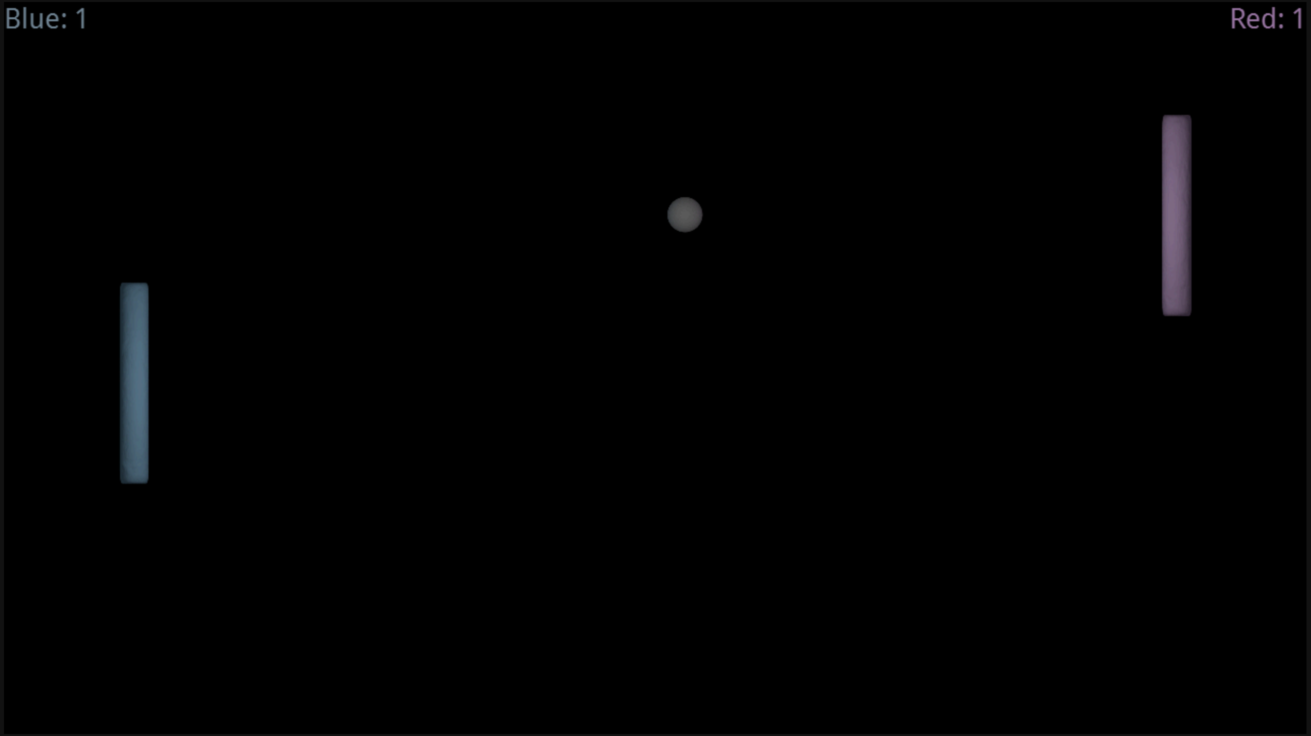
Old nCine logo



*The in-game editor for **ncRogue**, an RPG prototype for Android*

📸 2016 - ncPong

- The repository contains both a C++ and a Lua version



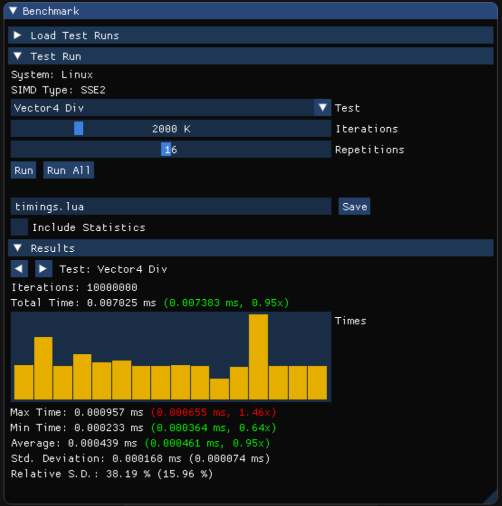
ncPong, *a minimal example game*

📆 2017 - A Leap Into The Snow ☃️

- I moved to Stockholm to work as a Rendering Engineer for Frostbite 🇸🇪 (📆 Mar)
  - To an interview question, I replied: "I don’t know how STL handles it, but in my framework…" 😎
  - Shortly after, I was ready to release the sources, but contract restrictions prevented it
  - Kept sharing binary builds with friends on Discord as a fallback
  - Credited in multiple EA titles, including FIFA 18-20, Battlefront II, Battlefield V, and Anthem
- Published the first "nCine Dev Update" article on my blog 📰 (📆 Aug)
- Dropped support for SDL1 in favour of SDL2 (also dropped `SDL_image`)
- Automatic culling of out-of-screen sprites (📆 Jun)
- Parsing of SDL2 gamepad axes and buttons mapping database for all backends
  - Out of the box support for a lot of "Xbox style" gamepads

📅 2018 - Catching up With the Times

- Atomic counters on all platforms (📅 Jan)
- Merged the `c++11` branch (df69bde1, 📅 Feb)
  - The birth of the *nCine Template Library* (nCTL)
- Merged the OpenGL 3.3 `new_renderer2` branch (b68f2de1, 📰 Dev Update 4, 📅 Jun)
- Added Lua bindings to support scripting (📅 Aug - Sep)
- Worked on SSE and NEON intrinsics for SIMD (unmerged `simd`, 📰 Dev Update 6, 📅 Nov)



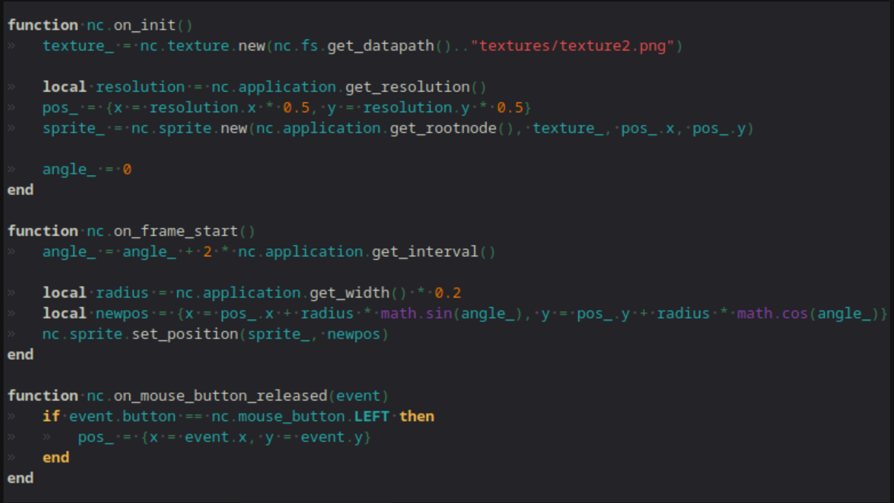
System: Linux
SIMD Type: SSE2
Test: Vector4 Div
Iterations: 2000 K
Repetitions: 16

timings.lua

```
Test: Vector4 Div
Iterations: 10000000
Total Time: 0.007025 ms (0.007383 ms, 0.95x)
```

```
Max Time: 0.000957 ms (0.000655 ms, 1.46x)
Min Time: 0.000233 ms (0.000364 ms, 0.64x)
Average: 0.000439 ms (0.000461 ms, 0.95x)
Std. Deviation: 0.000168 ms (0.000074 ms)
Relative S.D.: 38.19 % (15.96 %)
```

`apptest_simdbench`



```lua
function nc.on_init()
	texture_ = nc.texture.new(nc.fs.get_datapath().."textures/texture2.png")

	local resolution = nc.application.get_resolution()
	pos_ = {x = resolution.x * 0.5, y = resolution.y * 0.5}
	sprite_ = nc.sprite.new(nc.application.get_rootnode(), texture_, pos_.x, pos_.y)

	angle_ = 0
end

function nc.on_frame_start()
	angle_ = angle_ + 2 * nc.application.get_interval()

	local radius = nc.application.get_width() * 0.2
	local newpos = {x = pos_.x + radius * math.sin(angle_), y = pos_.y + radius * math.cos(angle_)}
	nc.sprite.set_position(sprite_, newpos)
end

function nc.on_mouse_button_released(event)
	if event.button == nc.mouse_button.LEFT then
		pos_ = {x = event.x, y = event.y}
	end
end
```

Snippet from `ncpong.lua`, the Lua version of the example game

💊 2018 - Atomic Counters

- Using compiler intrinsics to atomically update an integer value
  - Load, store, exchange, compare and exchange, test and set, fetch add/sub, …
- Enable lock-free data structures like the work stealing job queue (💊 Job System)
- 📜 Reference: Preshing on Programming

```
const int32_t t = top_; // nctl::Atomic32 top_;

    // ...
    if (top_.cmpExchange(t + 1, t) == false)
    {
      // a concurrent steal or pop operation removed an element from the deque in the meantime.
      return nullptr;
    }
```

We are trying to write `t + 1` *to* `top_`*, but only if no thread has modified it in the meantime.*
We expect its value to be `t`*; should it be different, we return a* `nullptr` *(*`src/threading/JobQueue.cpp` *🔗).*

💊 2018 - C++11 Subset in nCine

- Replace `NULL` with `nullptr`
- Mark disabled special member functions explicitly with `=delete`
- Adopt the `override` specifier
- Use delegating constructors to remove initialization functions
- Convert most enumerations to `enum class`
- Replace `typedef`s with type alias declarations (`using`)
- Introduce range-based loops in a few places (`for (IAudioPlayer *player : pausedPlayers_)`)
- Use `auto` sparingly with iterators
- Add support for move semantics in containers 💪
- Replace almost all raw pointers with smart pointers 🤓

💊 2018 - OpenGL 3.3 Renderer (1/2)

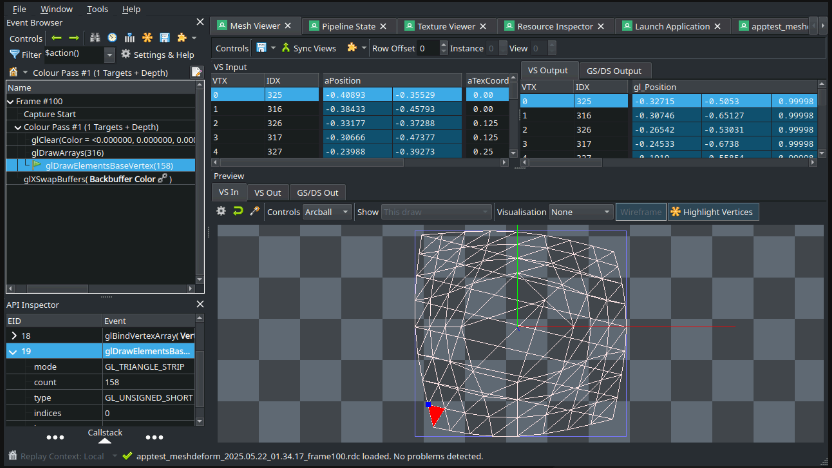
- Update to OpenGL 3.3 Core Profile and OpenGL ES 3.0
    - `KHR_debug` extension (`glDebugMessageCallback()`, `glPushDebugGroup()`, `glObjectLabel()`)
    - Vertex Array Objects (VAO) pool to efficiently switch VBOs and vertex formats with a single bind
    - Uniform Buffer Objects (UBO) to supply arbitrary data to multiple shaders at once
    - Immutable texture storage to skip per-draw texture checks (`glTexStorage2D()`)
- Rewrite the batcher to work with rendering commands (degenerate vertices, patched indices)
- Add a `RenderBufferManager` to use a single VBO, IBO, and UBO for all scene data
    - `GL_MAP_WRITE_BIT | GL_MAP_INVALIDATE_BUFFER_BIT | GL_MAP_FLUSH_EXPLICIT_BIT`
    - 📜 Reference: Buffer Object Streaming OpenGL Wiki page
- Add mesh sprites with custom vertices and UV coordinates

```cpp
// Should split if the lower part of a material's sort key or the primitive type differ
const bool shouldSplit = command->lowerMaterialSortKey() != prevCommand->lowerMaterialSortKey() ||
                         prevPrimitive != primitive;
```

Split condition for a batch of render commands.
Sort key encodes layer information in the upper 32 bits, and textures, shaders, and blending data in the lower 32 bits.
From `src/graphics/RenderBatcher.cpp` 🔗

💊 2018 - OpenGL 3.3 Renderer (2/2)

📸 2018 - RenderDoc and the New Renderer



Filter $action()

Colour Pass #1 (1 Targets + Depth)

Name
- Frame #100
  - Capture Start
  - Colour Pass #1 (1 Targets + Depth)
    - glClear(Color = <0.000000, 0.000000, 0.000
    - glDrawArrays(316)
    - glDrawElementsBaseVertex(158)
  - glXSwapBuffers(**Backbuffer Color**)

API Inspector

EID	Event
18	glBindVertexArray(**Vert**
19	glDrawElementsBas...
mode	GL_TRIANGLE_STRIP
count	158
type	GL_UNSIGNED_SHORT
indices	0

Row Offset 0 Instance 0 View 0

VS Input

VTX	IDX	aPosition		aTexCoord
0	325	-0.40893	-0.35529	0.00
1	316	-0.38433	-0.45793	0.00
2	326	-0.33177	-0.37288	0.125
3	317	-0.30666	-0.47377	0.125
4	327	-0.23988	-0.39273	0.25

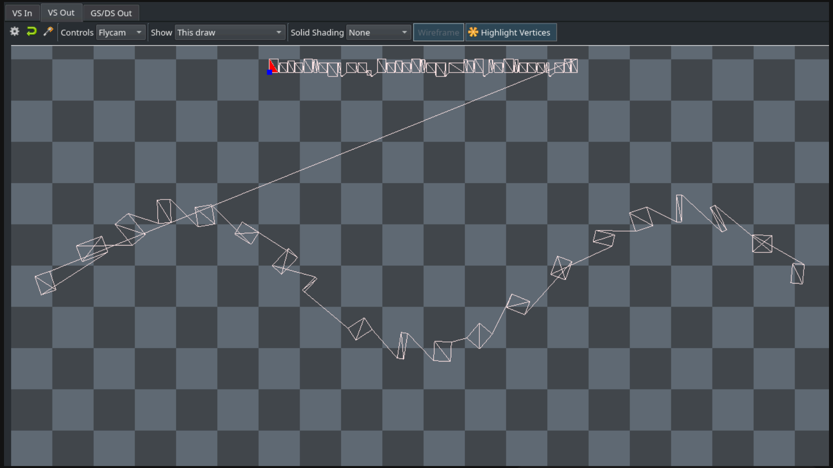
VS Output

VTX	IDX	gl_Position		
0	325	-0.32715	-0.5053	0.99998
1	316	-0.30746	-0.65127	0.99998
2	326	-0.26542	-0.53031	0.99998
3	317	-0.24533	-0.6738	0.99998

Controls Arcball Show This draw Visualisation None Wireframe Highlight Vertices

Replay Context: Local ✓ apptest_meshdeform_2025.05.22_01.34.17_frame100.rdc loaded. No problems detected.

RenderDoc showing a deformed mesh sprite from `apptest_meshdeform`

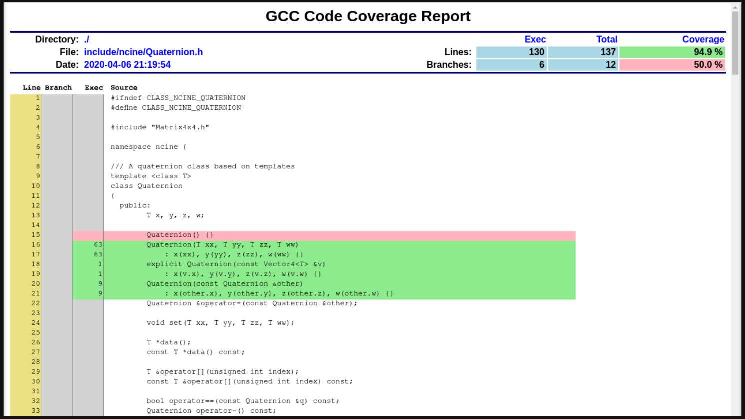


Controls Flycam Show This draw Solid Shading None Wireframe Highlight Vertices

RenderDoc showing degenerate vertices from `apptest_sinescroller`

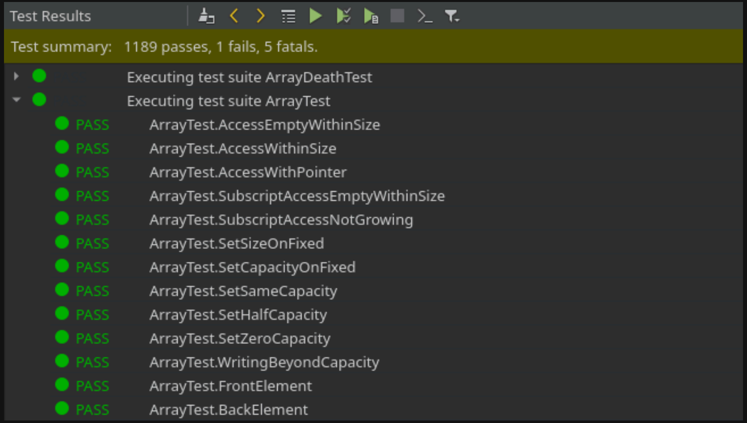
📅 2018 - The Birth of nCTL

- A template library with containers, iterators, algorithms, and more (-○- fc96c897)
- Using GoogleTest for unit testing
  - Using Gcovr for code coverage
  - Tests are taking a large portion of nCine codebase, but coverage results are decent
- Using Google Benchmark support library
  - nCTL vs STL: now I'm sure my library is both correct and fast 🚀



```
                         GCC Code Coverage Report

 Directory: ./                                         Exec   Total   Coverage
      File: include/ncine/Quaternion.h      Lines:      130     137     94.9 %
      Date: 2020-04-06 21:19:54          Branches:        6      12     50.0 %

 Line Branch   Exec  Source
    1                #ifndef CLASS_NCINE_QUATERNION
    2                #define CLASS_NCINE_QUATERNION
    3
    4                #include "Matrix4x4.h"
    5
    6                namespace ncine {
    7
    8                /// A quaternion class based on templates
    9                template <class T>
   10                class Quaternion
   11                {
   12                  public:
   13                        T x, y, z, w;
   14
   15                        Quaternion() {}
   16                    63  Quaternion(T xx, T yy, T zz, T ww)
   17                    63      : x(xx), y(yy), z(zz), w(ww) {}
   18                     1  explicit Quaternion(const Vector4<T> &v)
   19                     1      : x(v.x), y(v.y), z(v.z), w(v.w) {}
   20                     9  Quaternion(const Quaternion &other)
   21                     9      : x(other.x), y(other.y), z(other.z), w(other.w) {}
   22                        Quaternion &operator=(const Quaternion &other);
   23
   24                        void set(T xx, T yy, T zz, T ww);
   25
   26                        T *data();
   27                        const T *data() const;
   28
   29                        T &operator[](unsigned int index);
   30                        const T &operator[](unsigned int index) const;
   31
   32                        bool operator==(const Quaternion &q) const;
   33                        Quaternion operator-() const;
```

A unit test code coverage report by Gcovr



```
Test summary:  1189 passes, 1 fails, 5 fatals.

▸ ●          Executing test suite ArrayDeathTest
▾ ●          Executing test suite ArrayTest
    ● PASS      ArrayTest.AccessEmptyWithinSize
    ● PASS      ArrayTest.AccessWithinSize
    ● PASS      ArrayTest.AccessWithPointer
    ● PASS      ArrayTest.SubscriptAccessEmptyWithinSize
    ● PASS      ArrayTest.SubscriptAccessNotGrowing
    ● PASS      ArrayTest.SetSizeOnFixed
    ● PASS      ArrayTest.SetCapacityOnFixed
    ● PASS      ArrayTest.SetSameCapacity
    ● PASS      ArrayTest.SetHalfCapacity
    ● PASS      ArrayTest.SetZeroCapacity
    ● PASS      ArrayTest.WritingBeyondCapacity
    ● PASS      ArrayTest.FrontElement
    ● PASS      ArrayTest.BackElement
```

Google Test integration and support by the Qt Creator IDE

💊 2018 - nCine Template Library (1/3)

- Arrays, atomics, hash functions, hashmaps, hashsets, lists, unique/shared pointers, sparse sets, strings
  - Most components are unit tested and benchmarked against the STL
  - Containers support custom allocators and stack allocation, iterators support templated algorithms
- Uses C++ generic programming techniques like type traits, tag dispatching, and SFINAE
  - SFINAE stands for "Substitution Failure Is Not An Error"
  - SFINAE disables invalid templates instantiations instead of triggering compile-time errors

```cpp
template <class T>
struct isTriviallyConstructible
{
  static constexpr bool value = __is_trivially_constructible(T);
};

template <class T>
void destructArray(T *ptr, unsigned int numElements)
{
    detail::destructHelpers<isTriviallyDestructible<T>::value>::destructArray(ptr, numElements);
}
```

No STL, everything is built from scratch, sometimes requiring compiler intrinsics.
`destructArray()` *uses SFINAE to specialize behavior based on type traits (before C++20 concepts).*
From `include/nctl/type_traits.h` 🔗

💊 2018 - nCine Template Library (2/3)

```
template <class T>
Array<T>::~Array()
{
    destructArray(array_, size_);
#if !NCINE_WITH_ALLOCATORS
    ::operator delete(array_);
#else
    alloc_.deallocate(array_);
#endif
}
```

Before deallocating memory, array elements are destroyed using `destructArray()`*.*
Thanks to SFINAE, destruction is skipped for trivially destructible types.
From `include/nctl/Array.h` 🔗

```
/// It is used to indicate that an object may be "moved from"
template <class T>
inline typename removeReference<T>::type &&move(T &&arg)
{
  return static_cast<typename removeReference<T>::type &&>(arg);
}
```

A common misconception: `move()` *simply casts an* ***lvalue*** *to an* ***xvalue****.*
From `include/nctl/utility.h` 🔗

💊 2018 - nCine Template Library (3/3)

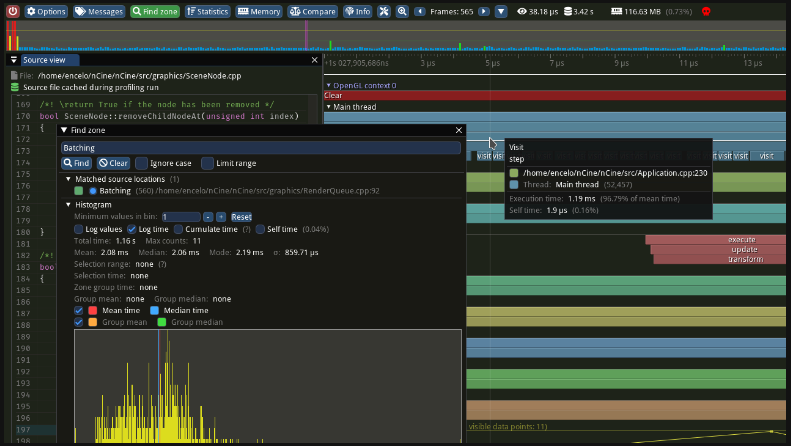
```cpp
/// Returns the distance between two random access iterators with a pointer subtraction
template <class RandomAccessIterator>
inline int distance(RandomAccessIterator &first, const RandomAccessIterator &last, RandomAccessIteratorTag)
{
    return last - first;
}

/// Returns the distance in number of increments between two forward iterators
template <class ForwardIterator>
inline int distance(ForwardIterator &first, const ForwardIterator &last, ForwardIteratorTag)
{
    int counter = 0;
    for (; first != last; ++first)
        counter++;

    return counter;
}
```

Tag dispatching uses function overloading to choose the best implementation based on iterator type at compile time.

From `include/nctl/iterator.h` 🔗

📅 2018 - Instrumentation and UI Integration

- Small String Optimization for the string class (📅 Feb)
- Integration with Dear ImGui, an immediate mode GUI toolkit (-○- aff7e611, 📅 Aug)
- Add a debug overlay interface made with ImGui
- Integration with the Tracy frame profiler (-○- c8338ace, 📅 Dec)
- `ncParticleEditor`, an ImGui editor for particle systems and emitters



File: /home/encelo/nCine/nCine/src/graphics/SceneNode.cpp
Source file cached during profiling run

```
/*! \return True if the node has been removed */
bool SceneNode::removeChildNodeAt(unsigned int index)
{




        


}

/*!
boo
{
```

Batching

Batching (560) /home/encelo/nCine/nCine/src/graphics/RenderQueue.cpp:92

Minimum values in bin: 1

☐ Log values ☑ Log time ☐ Cumulate time (?) ☐ Self time (0.04%)

Total time: 1.16 s Max counts: 11

Mean: 2.08 ms Median: 2.06 ms Mode: 2.19 ms σ: 859.71 μs

Selection range: none (?)

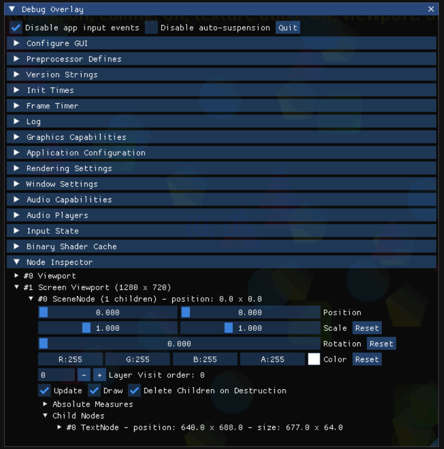
Selection time: none

Zone group time: none

Group mean: none Group median: none

Visit
step
/home/encelo/nCine/nCine/src/Application.cpp:230
Thread: Main thread (52,457)
Execution time: 1.19 ms (96.79% of mean time)
Self time: 1.9 μs (0.16%)

A performance capture analysed by the Tracy frame profiler



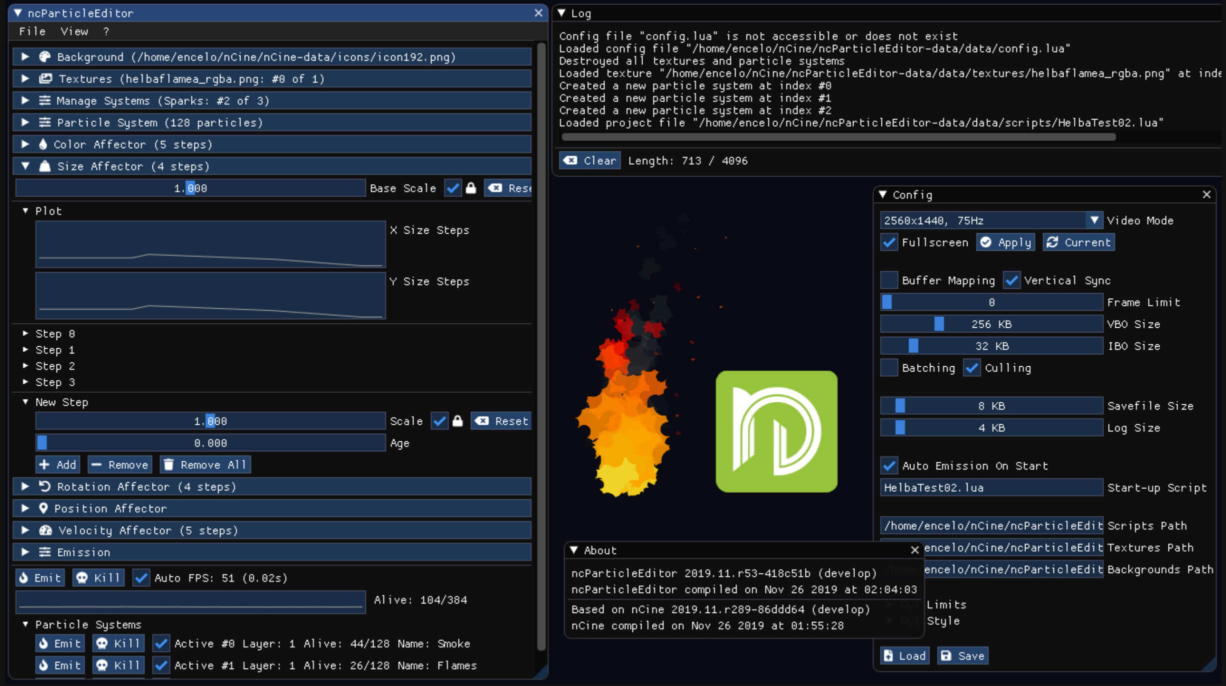
☑ Disable app input events ☐ Disable auto-suspension

#0 Viewport
#1 Screen Viewport (1280 x 720)
#0 SceneNode (1 children) - position: 0.0 x 0.0
0.000 0.000 Position
1.000 1.000 Scale
0.000 Rotation
R:255 G:255 B:255 A:255 Color
0 Layer Visit order: 0
☑ Update ☑ Draw ☑ Delete Children on Destruction
#0 TextNode - position: 640.0 x 688.0 - size: 677.0 x 64.0

The Debug Overlay interface

📸 2018 - ncParticleEditor

- Developed alongside an artist for real-world feedback
  - Integrated CrashRpt on Windows to receive crash mini-dumps
  - Now transitioning to Google Crashpad, as CrashRpt is no longer maintained



Background (/home/encelo/nCine/nCine-data/icons/icon192.png)
Textures (helbaflamea_rgba.png: #0 of 1)
Manage Systems (Sparks: #2 of 3)
Particle System (128 particles)
Color Affector (5 steps)
Size Affector (4 steps)
1.000 Base Scale ☑
Scale 1.000 ☑
Age 0.000
☑ Auto FPS: 51 (0.02s)
Alive: 104/384
☑ Active #0 Layer: 1 Alive: 44/128 Name: Smoke
☑ Active #1 Layer: 1 Alive: 26/128 Name: Flames

```
Config file "config.lua" is not accessible or does not exist
Loaded config file "/home/encelo/nCine/ncParticleEditor-data/data/config.lua"
Destroyed all textures and particle systems
Loaded texture "/home/encelo/nCine/ncParticleEditor-data/data/textures/helbaflamea_rgba.png" at inde
Created a new particle system at index #0
Created a new particle system at index #1
Created a new particle system at index #2
Loaded project file "/home/encelo/nCine/ncParticleEditor-data/data/scripts/HelbaTest02.lua"
```

Length: 713 / 4096

Video Mode: 2560x1440, 75Hz
☑ Fullscreen
☐ Buffer Mapping
☑ Vertical Sync
Frame Limit: 0
VBO Size: 256 KB
IBO Size: 32 KB
☐ Batching
☑ Culling
Savefile Size: 8 KB
Log Size: 4 KB
☑ Auto Emission On Start
Start-up Script: HelbaTest02.lua
Scripts Path: /home/encelo/nCine/ncParticleEdit
Textures Path: encelo/nCine/ncParticleEdit
Backgrounds Path: encelo/nCine/ncParticleEdit

```
ncParticleEditor 2019.11.r53-418c51b (develop)
ncParticleEditor compiled on Nov 26 2019 at 02:04:03
Based on nCine 2019.11.r289-86ddd64 (develop)
nCine compiled on Nov 26 2019 at 01:55:28
```

ncParticleEditor *showing a project by Helba*

💊 2018 - Small String Optimization

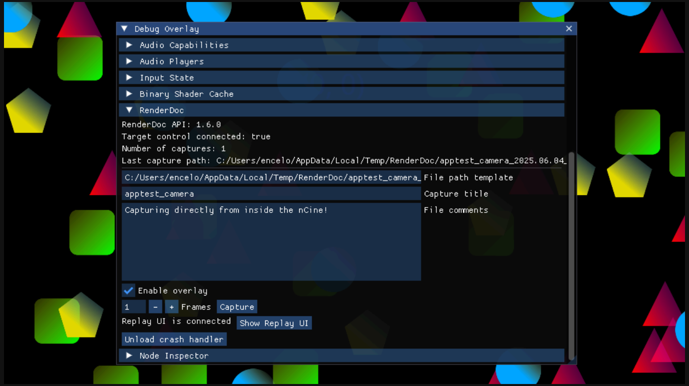
- Short strings stored inside the string object, avoiding heap allocation
- Buffer size chosen to fit the whole string object in one CPU cache line
- Since CPUs load full cache lines, accessing short strings is essentially free
- Larger strings trigger heap allocation and copying

```cpp
class String
{
// ...
  private:
    static const unsigned int SmallBufferSize = 16;

    union Buffer
    {
        char *begin_;
        char local_[SmallBufferSize];
    };

    Buffer array_;
    unsigned int length_;
    unsigned int capacity_;
};
```

SSO, a common optimization also found in `std::string` (`src/base/String.cpp` 🔗)

📅 2019 - nCine Goes Open Source

- I left Sweden and EA behind for Granada in Spain 🇪🇸 (📅 Feb)
- I could finally release the nCine on GitHub! (📅 Jun)
  - Featured on Phoronix and GameFromScratch.com
  - Using Azure Pipelines for Continuous Integration ✅ (📰 Dev Update 10, 📅 May)
- Some experiments with ECS (unmerged `ecs`, 📅 Mar)
- Porting to Emscripten for web support (📰 Dev Update 11)
- Integration with the RenderDoc GFX debugger and the Nuklear immediate GUI
- New hashmap with open addressing and probing (98f2364d, 📅 Jan)



```
RenderDoc API: 1.6.0
Target control connected: true
Number of captures: 1
Last capture path: C:/Users/encelo/AppData/Local/Temp/RenderDoc/apptest_camera_2025.06.04_
```

File path template: C:/Users/encelo/AppData/Local/Temp/RenderDoc/apptest_camera_

Capture title: apptest_camera

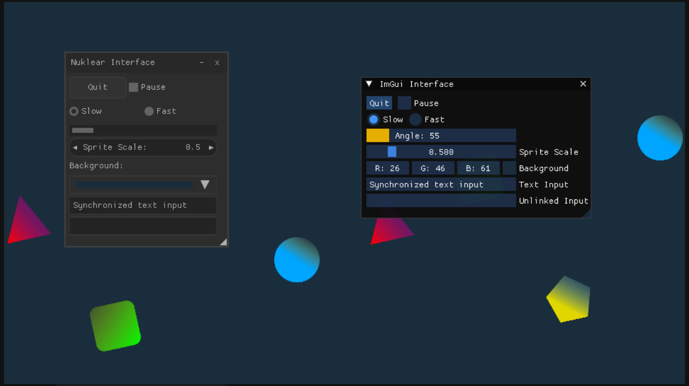
File comments: Capturing directly from inside the nCine!

☑ Enable overlay

1 Frames

Replay UI is connected

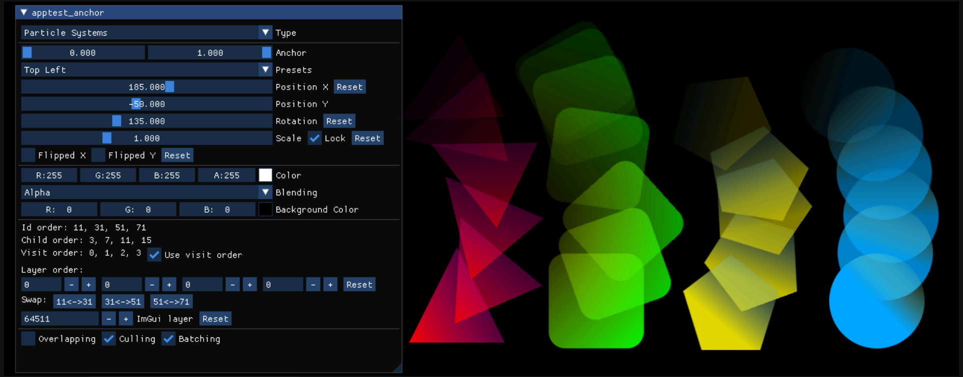
Integration with the RenderDoc API



`apptest_gui`, Nuklear and ImGui at the same time

📅 2019 - Broadening the Ecosystem

- `ncIsometric`, a prototype isometric turn-based game that was never released (📅 Apr)
- `ncInvaders`, my Space Invaders clone with some data-oriented design (📅 Jul)
- `ncTemplate`, a template CMake project to clone and use as a starting point (📅 Jul)
- `ncTracer`, a CPU path-tracer with multi-threading and an ImGui interface (📅 Aug)
- `ncline`, a command line tool to download and compile dependencies, nCine, and projects (📅 Sep)
- Collaboration with Jugilus begins for the JugiMap integration (📅 Nov)
  - Sprite enhancements: non-uniform scaling, anchor points, blending factors (📰 Dev Update 13)
  - The collaboration continues to this day 🫶



▼ apptest_anchor

Type: Particle Systems

Anchor: 0.000 1.000

Presets: Top Left

Position X: 185.000

Position Y: -50.000

Rotation: 135.000

Scale: 1.000 ☑ Lock

☐ Flipped X ☐ Flipped Y

Color: R:255 G:255 B:255 A:255

Blending: Alpha

Background Color: R: 0 G: 0 B: 0

Id order: 11, 31, 51, 71
Child order: 3, 7, 11, 15
Visit order: 0, 1, 2, 3 ☑ Use visit order

Layer order: 0 0 0 0

Swap: 11<->31 31<->51 51<->71

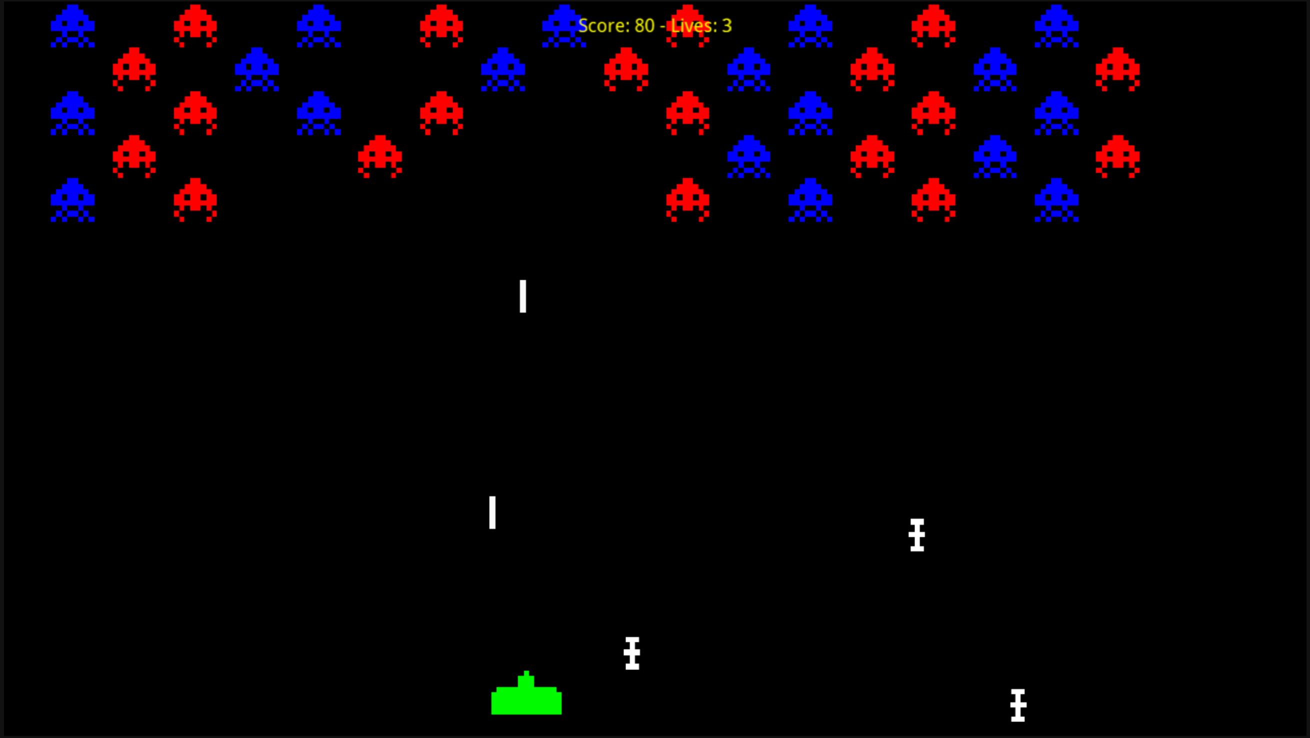
ImGui layer: 64511

☐ Overlapping ☑ Culling ☑ Batching

`apptest_anchor`, showing sprites rotating around their anchor points

📸 2019 - ncInvaders

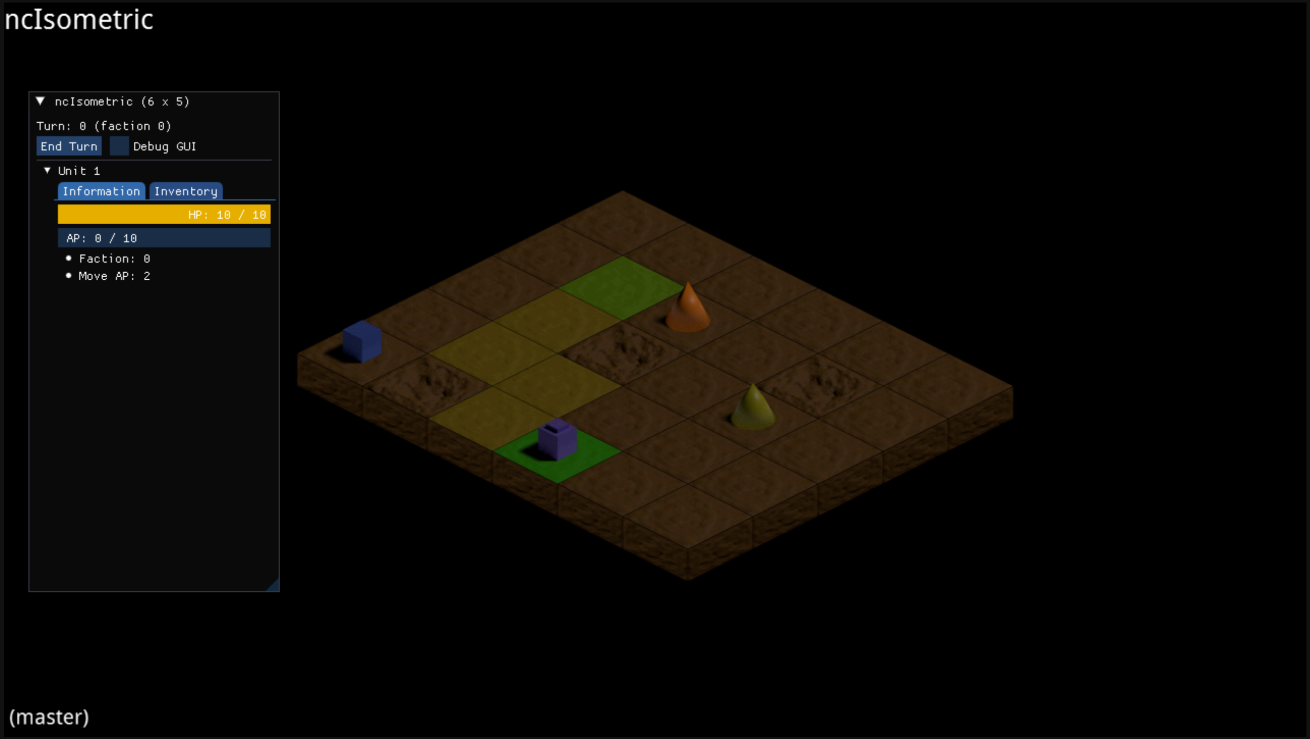
- A simplified Space Invaders clone with some Data Oriented Design principles
  - I adapted part of the code I wrote for the Frostbite technical assignment 😉



ncInvaders, a simplified Space Invaders clone

📸 2019 - ncIsometric

- Unreleased game prototype made in Spain while unemployed
- A temporary ImGui interface, A* pathfinding, an incomplete *utility AI* for enemies, and Blender graphics


ncIsometric

ncIsometric (6 x 5)
Turn: 0 (faction 0)

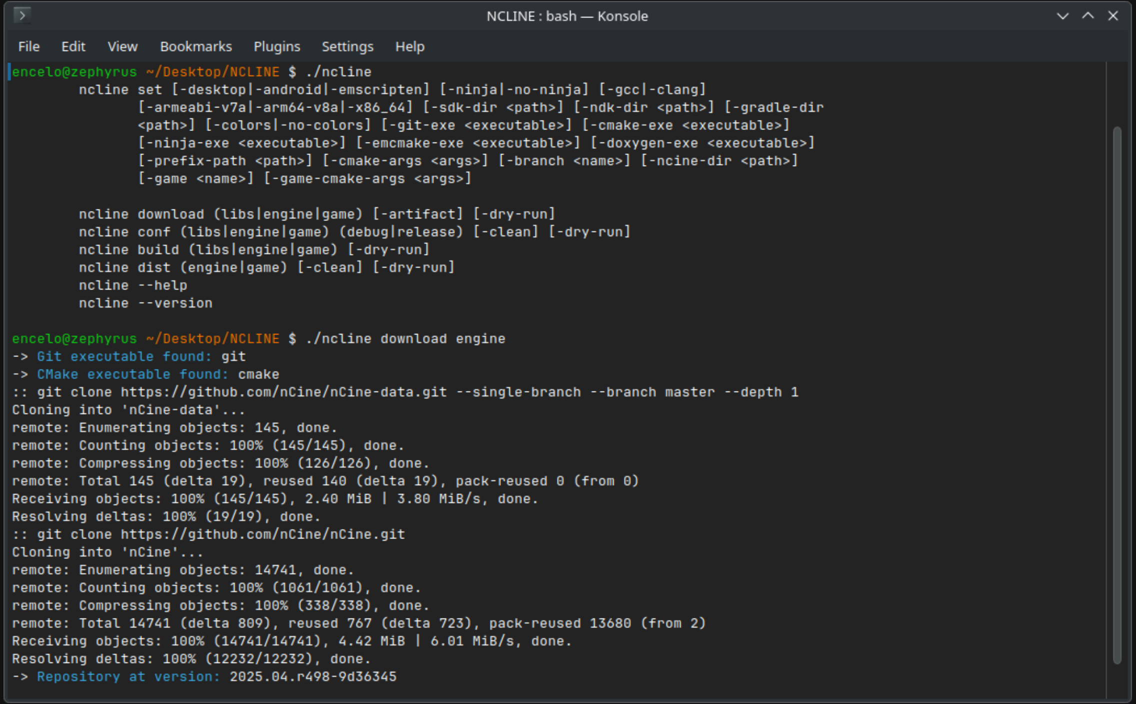
HP: 10 / 10
AP: 0 / 10
- Faction: 0
- Move AP: 2

(master)

ncIsometric, an isometric turn-based combat prototype

📸 2019 - ncline

- Works on all supported desktop platforms leveraging Git and CMake
- Can download sources or artifacts, then compile dependencies, the nCine, and your project
- Inspired by a similar tool we had in Frostbite

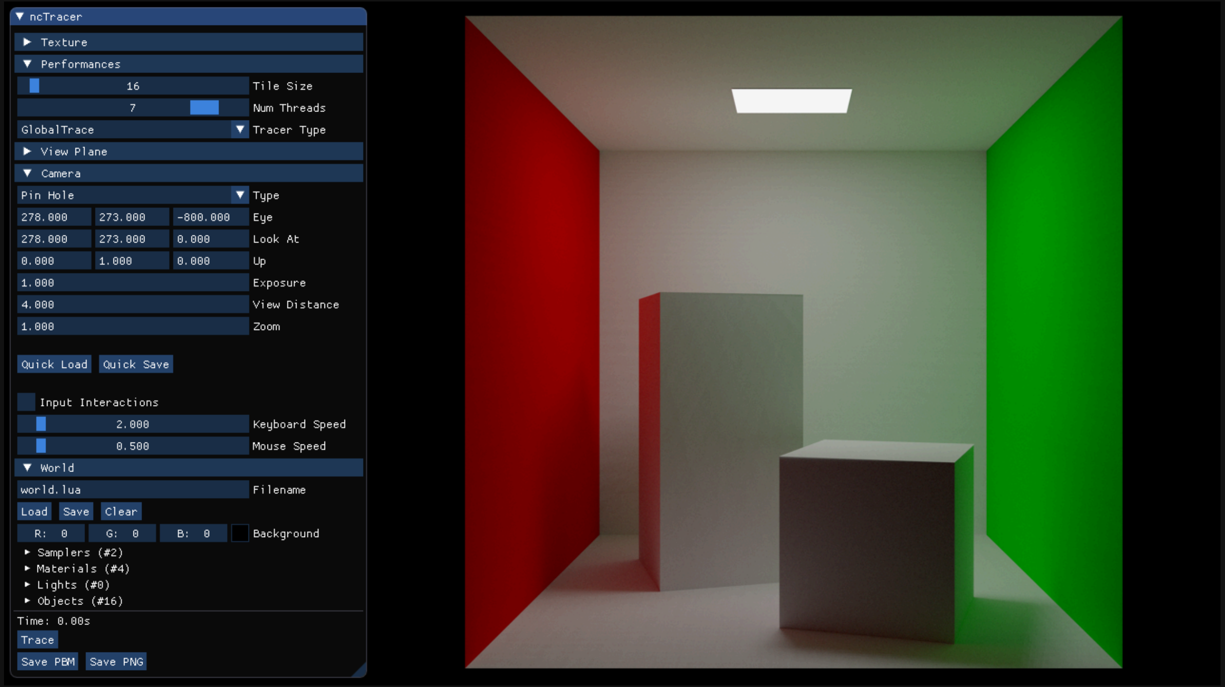


```
encelo@zephyrus ~/Desktop/NCLINE $ ./ncline
        ncline set [-desktop|-android|-emscripten] [-ninja|-no-ninja] [-gcc|-clang]
               [-armeabi-v7a|-arm64-v8a|-x86_64] [-sdk-dir <path>] [-ndk-dir <path>] [-gradle-dir
               <path>] [-colors|-no-colors] [-git-exe <executable>] [-cmake-exe <executable>]
               [-ninja-exe <executable>] [-emcmake-exe <executable>] [-doxygen-exe <executable>]
               [-prefix-path <path>] [-cmake-args <args>] [-branch <name>] [-ncine-dir <path>]
               [-game <name>] [-game-cmake-args <args>]

        ncline download (libs|engine|game) [-artifact] [-dry-run]
        ncline conf (libs|engine|game) (debug|release) [-clean] [-dry-run]
        ncline build (libs|engine|game) [-dry-run]
        ncline dist (engine|game) [-clean] [-dry-run]
        ncline --help
        ncline --version

encelo@zephyrus ~/Desktop/NCLINE $ ./ncline download engine
-> Git executable found: git
-> CMake executable found: cmake
:: git clone https://github.com/nCine/nCine-data.git --single-branch --branch master --depth 1
Cloning into 'nCine-data'...
remote: Enumerating objects: 145, done.
remote: Counting objects: 100% (145/145), done.
remote: Compressing objects: 100% (126/126), done.
remote: Total 145 (delta 19), reused 140 (delta 19), pack-reused 0 (from 0)
Receiving objects: 100% (145/145), 2.40 MiB | 3.80 MiB/s, done.
Resolving deltas: 100% (19/19), done.
:: git clone https://github.com/nCine/nCine.git
Cloning into 'nCine'...
remote: Enumerating objects: 14741, done.
remote: Counting objects: 100% (1061/1061), done.
remote: Compressing objects: 100% (338/338), done.
remote: Total 14741 (delta 809), reused 767 (delta 723), pack-reused 13680 (from 2)
Receiving objects: 100% (14741/14741), 4.42 MiB | 6.01 MiB/s, done.
Resolving deltas: 100% (12232/12232), done.
-> Repository at version: 2025.04.r498-9d36345
```

***ncline**, the nCine command line tool*

📸 2019 - ncTracer

- Simple CPU-only multi-threaded path tracer using my pmTracer library as backend
- Links nCine statically to bypass symbol visibility and access threads and OpenGL directly
- 📜 Reference: *Ray Tracing from the Ground Up*



16 Tile Size
7 Num Threads
GlobalTrace Tracer Type
Pin Hole Type
278.000 273.000 -800.000 Eye
278.000 273.000 0.000 Look At
0.000 1.000 0.000 Up
1.000 Exposure
4.000 View Distance
1.000 Zoom
☐ Input Interactions
2.000 Keyboard Speed
0.500 Mouse Speed
world.lua Filename
R: 0 G: 0 B: 0 Background
▸ Samplers (#2)
▸ Materials (#4)
▸ Lights (#0)
▸ Objects (#16)
Time: 0.00s

The classic Cornell box scene as rendered by ncTracer

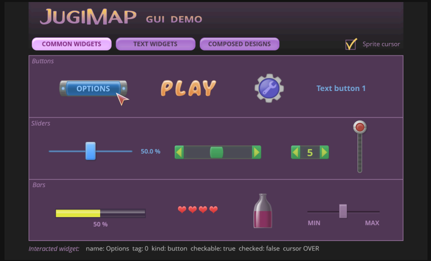
📸 2019 - JugiMap Framework and Projects

- One day, a user wrote me a pretty long list of feature requests on Discord 📝
- He was developing a map editor with Qt 5
  - The tool included a runtime to load maps in various 2D engines (Cocos2d-x, AGK-tier 2, SFML)
- His demos really put the nCine under stress 🥵
  - I fixed a lot of bugs and added all the requested features 💪
- The nCine was the fastest supported engine and was chosen for the web demos on the site

ncJugiMapFrameworkDemo

ncJugiMapParallaxScrolling

ncJugiMapSpriteTimelineAnimation

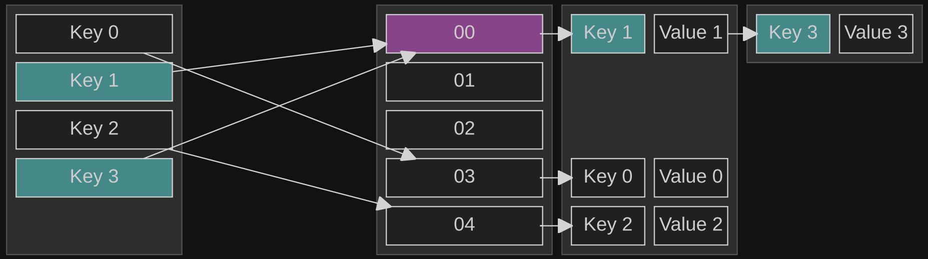


Interacted widget: name: Options tag: 0 kind: button checkable: true checked: false cursor OVER

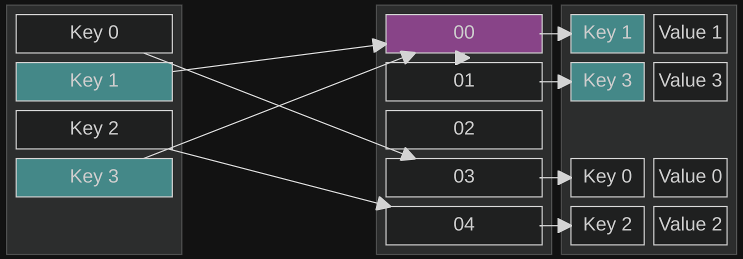
ncJugiMapGuiDemo

💊 2019 - Leapfrog Probing for Hashmaps (1/2)

- There are two main strategies for key collision resolution
- *Separate Chaining*: entries with the same array/bucket index are stored in a linked list
  - Performance declines as the load factor grows, but there is no limit on the number of entries
  - Cache inefficiencies due to poor space-locality of lists
- *Open Addressing*: entries are all stored directly into the array, with a *probe sequence* for collision
  - They require rehashing into a larger array as the load factor approaches 1
  - More cache-friendly as all entries are stored sequentially in the array
- More information and performance results in Dev Update 7 📰



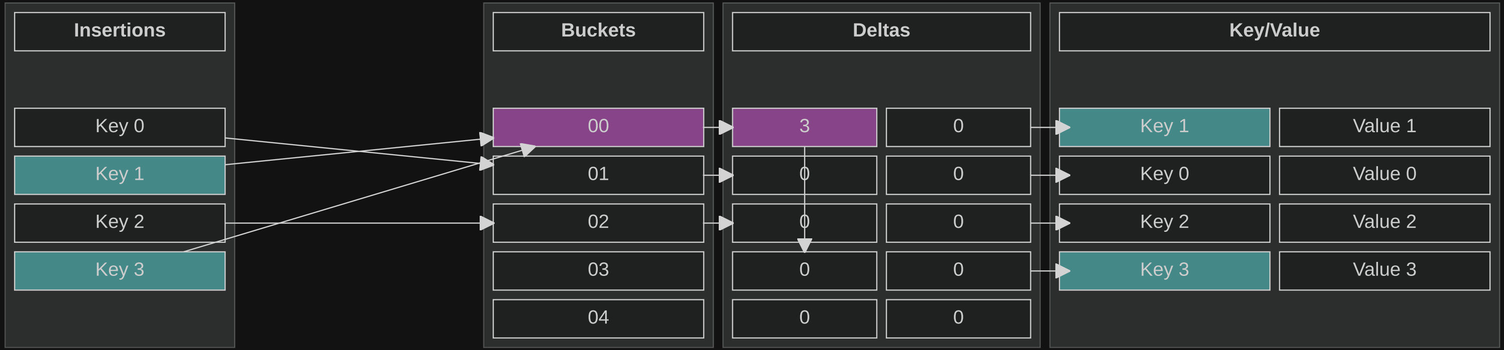
Separate Chaining



Open Addressing with Linear Probing

💊 2019 - Leapfrog Probing for Hashmaps (2/2)

- A collision resolution probing strategy, using two additional delta values per array cell
  1. Hash the key and compute the bucket index. That's the ideal index, which we check first
  2. If the item is not found, add that cell's first delta value to determine the next cell index to check
  3. If the item is not found, use the second delta value for all subsequent cells
  4. Stop when the delta is zero, marking the end of the probe chain
- 📜 Reference: Preshing's article



Insertions

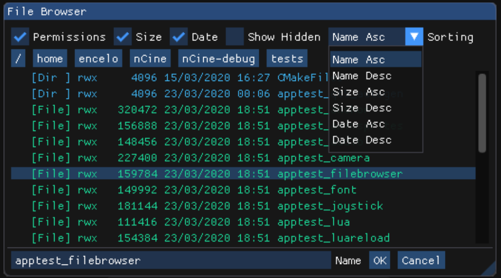
Key 0
Key 1
Key 2
Key 3

Buckets	Deltas		Key/Value	
00	3	0	Key 1	Value 1
01	0	0	Key 0	Value 0
02	0	0	Key 2	Value 2
03	0	0	Key 3	Value 3
04	0	0		

Open Addressing with Leapfrog Probing

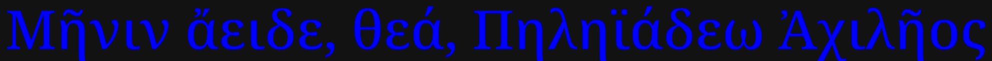
🗓️ 2020 - A Token of Support 💲

- Added the Qt 5 backend on desktop (📰 Dev Update 14)
- Added a filesystem class (POSIX and WinAPI implementations) (-○- 1a82d94a, 🗓️ Mar)
- Added support for UTF-8 decoding in strings, enabling proper display of non-ASCII characters
- Thinking about building an editor… 🤔 (#1)
    - Don't `FATAL_ASSERT` if a resource (texture, font, audio buffer) can't be loaded
    - Allow sprites with no textures, text nodes with no fonts, audio players with no buffers
- Add support for custom memory allocators (-○- defb333a, 📰 Dev Update 15, 🗓️ Apr)
- Migrated from Azure to GitHub Actions for C.I. (-○- c370ad59, 🗓️ Nov)
- Didn't get an Epic MegaGrant (🗓️ May), got 250$ for the Icculus Microgrant 2020 (🗓️ Dec)



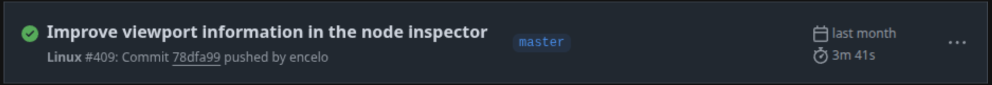
```
[Dir ] rwx      4096 15/03/2020 16:27 CMakeFil
[Dir ] rwx      4096 23/03/2020 00:06 apptest_
[File] rwx    320472 23/03/2020 18:51 apptest_
[File] rwx    156888 23/03/2020 18:51 apptest_
[File] rwx    148456 23/03/2020 18:51 apptest_
[File] rwx    227400 23/03/2020 18:51 apptest_camera
[File] rwx    159784 23/03/2020 18:51 apptest_filebrowser
[File] rwx    149992 23/03/2020 18:51 apptest_font
[File] rwx    181144 23/03/2020 18:51 apptest_joystick
[File] rwx    111416 23/03/2020 18:51 apptest_lua
[File] rwx    154384 23/03/2020 18:51 apptest_luareload
apptest_filebrowser
```

`apptest_filebrowser`



Μῆνιν ἄειδε, θεά, Πηληϊάδεω Ἀχιλῆος

`apptest_font` rendering the first line of the Iliad, thanks to UTF-8 support



Improve viewport information in the node inspector `master`

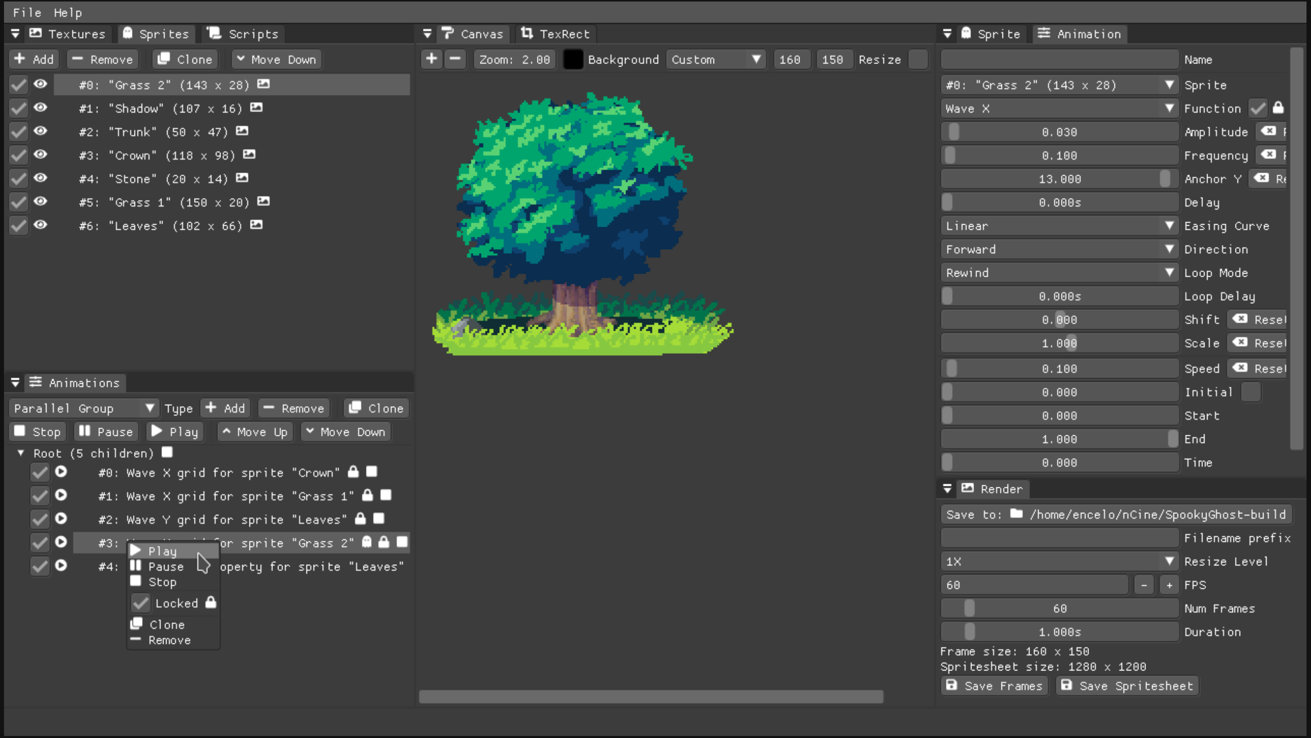
Linux #409: Commit 78dfa99 pushed by encelo

last month
3m 41s

An entry in the nCine GitHub Actions workflow runs page

📸 2020 - SpookyGhost

- I tried selling a tool for artists on itch.io, but it didn't gain traction
- It's now free and open-source on GitHub, with optional donations still available



#0: "Grass 2" (143 x 28)
#1: "Shadow" (107 x 16)
#2: "Trunk" (50 x 47)
#3: "Crown" (118 x 98)
#4: "Stone" (20 x 14)
#5: "Grass 1" (150 x 20)
#6: "Leaves" (102 x 66)

Parallel Group
Root (5 children)
#0: Wave X grid for sprite "Crown"
#1: Wave X grid for sprite "Grass 1"
#2: Wave Y grid for sprite "Leaves"
#3: … for sprite "Grass 2"
#4: … operty for sprite "Leaves"

Play
Pause
Stop
Locked
Clone
Remove

Sprite: #0: "Grass 2" (143 x 28)
Function: Wave X
Amplitude: 0.030
Frequency: 0.100
Anchor Y: 13.000
Delay: 0.000s
Easing Curve: Linear
Direction: Forward
Loop Mode: Rewind
Loop Delay: 0.000s
Shift: 0.000
Scale: 1.000
Speed: 0.100
Initial: 0.000
Start: 0.000
End: 1.000
Time: 0.000

Save to: /home/encelo/nCine/SpookyGhost-build
Filename prefix:
Resize Level: 1X
FPS: 60
Num Frames: 60
Duration: 1.000s
Frame size: 160 x 150
Spritesheet size: 1280 x 1200

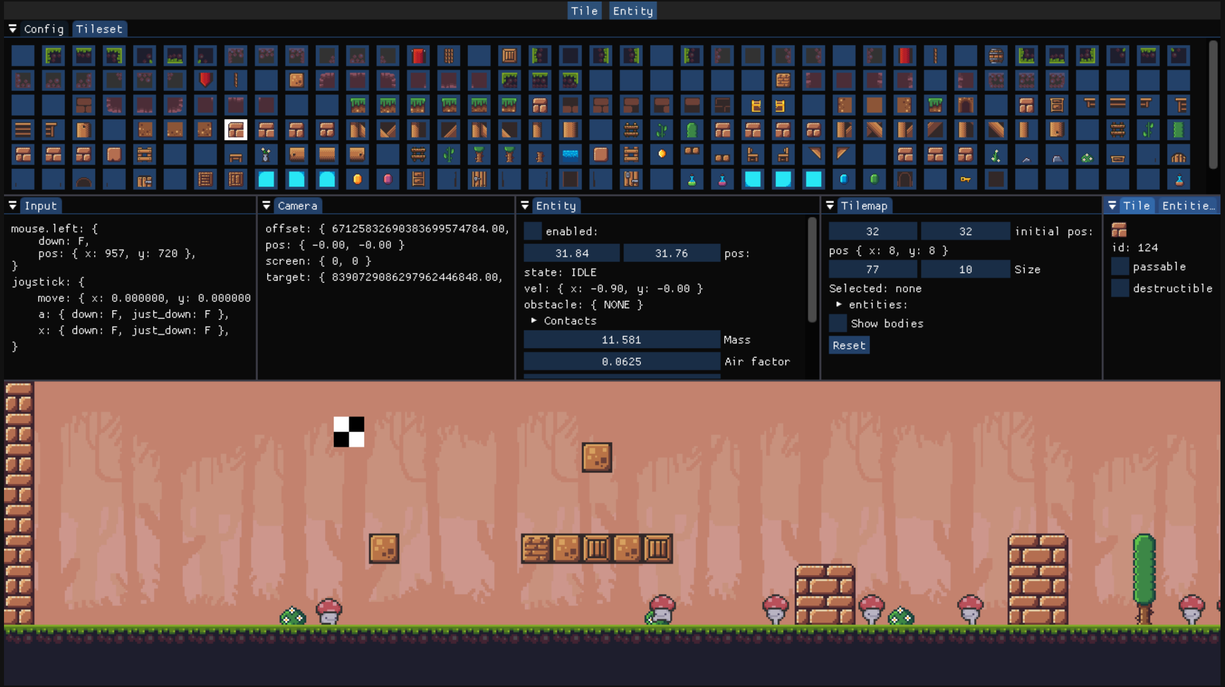
SpookyGhost, *a procedural animation tool for 2D sprites*

Laundry animation

Tree animation

📸 2020 - ncJump

- Started in December by Fahien, to demonstrate what nCine could do
- Uses Box2D for physics and Dear ImGui for on the fly editing
- Among the first nCine projects successfully tested on the Steam Deck



```
mouse.left: {
    down: F,
    pos: { x: 957, y: 720 },
}
joystick: {
    move: { x: 0.000000, y: 0.000000
    a: { down: F, just_down: F },
    x: { down: F, just_down: F },
}
```

```
offset: { 6712583269038369957478 4.00,
pos: { -0.00, -0.00 }
screen: { 0, 0 }
target: { 8390729086297962446848.00,
```

```
☐ enabled:
31.84    31.76    pos:
state: IDLE
vel: { x: -0.90, y: -0.00 }
obstacle: { NONE }
 ▸ Contacts
11.581   Mass
0.0625   Air factor
```

```
32    32    initial pos:
pos { x: 8, y: 8 }
77    10    Size
Selected: none
 ▸ entities:
☐ Show bodies
Reset
```

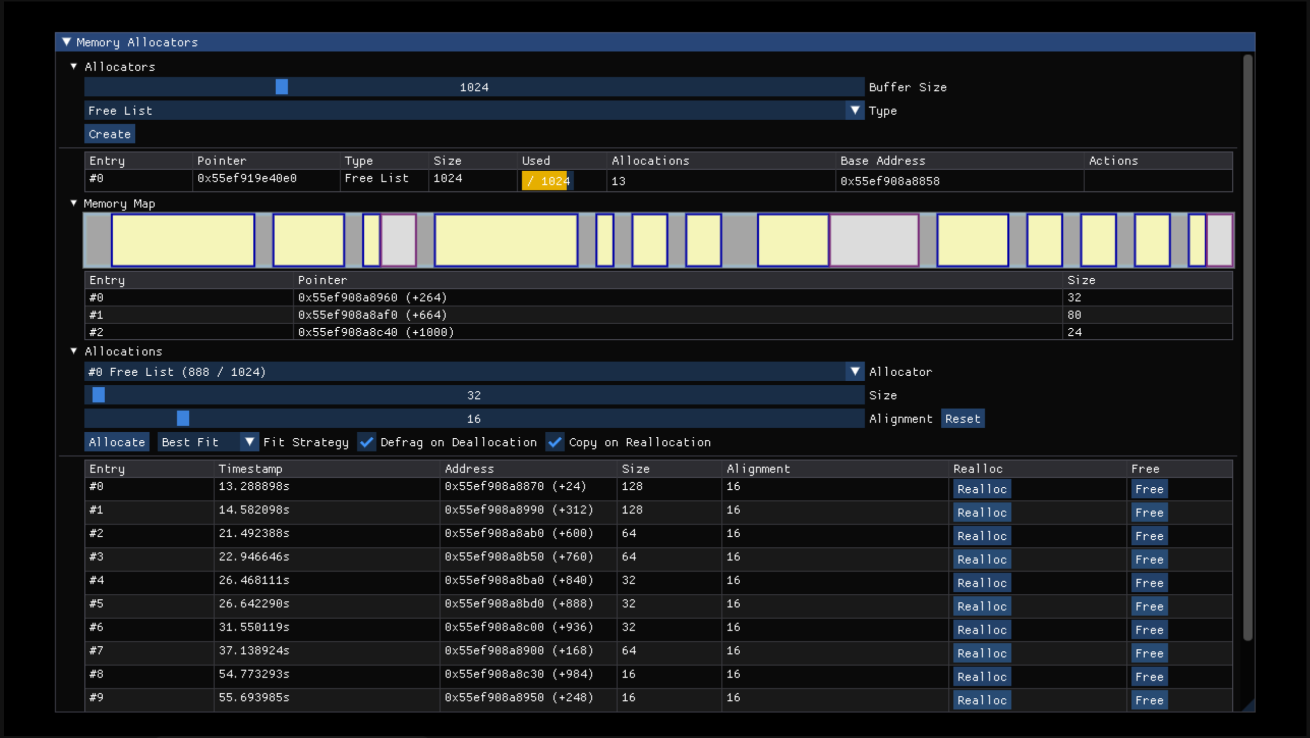
```
id: 124
☐ passable
☐ destructible
```

ncJump, a jumping project powered by nCine 🦘

ncJump running on my Steam Deck

📸 2020 - Allocators Application Test

- Visually demonstrates allocator behavior using ImGui low-level widget API
- A similar table with recorded allocations can also be seen in the debug overlay



▼ Memory Allocators

▼ Allocators

1024 Buffer Size

Free List ▼ Type

Entry	Pointer	Type	Size	Used	Allocations	Base Address	Actions
#0	0x55ef919e40e0	Free List	1024	/ 1024	13	0x55ef908a8858	

▼ Memory Map

Entry	Pointer	Size
#0	0x55ef908a8960 (+264)	32
#1	0x55ef908a8af0 (+664)	80
#2	0x55ef908a8c40 (+1000)	24

▼ Allocations

#0 Free List (888 / 1024) ▼ Allocator

32 Size

16 Alignment

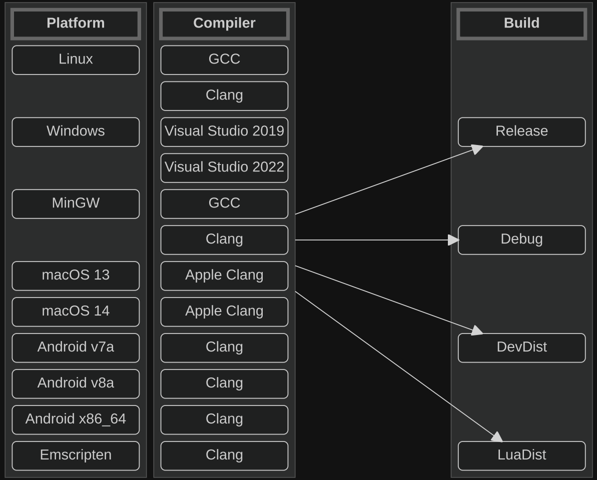
Best Fit ▼ Fit Strategy ☑ Defrag on Deallocation ☑ Copy on Reallocation

Entry	Timestamp	Address	Size	Alignment	Realloc	Free
#0	13.288898s	0x55ef908a8870 (+24)	128	16	Realloc	Free
#1	14.582098s	0x55ef908a8990 (+312)	128	16	Realloc	Free
#2	21.492388s	0x55ef908a8ab0 (+600)	64	16	Realloc	Free
#3	22.946646s	0x55ef908a8b50 (+760)	64	16	Realloc	Free
#4	26.468111s	0x55ef908a8ba0 (+840)	32	16	Realloc	Free
#5	26.642290s	0x55ef908a8bd0 (+888)	32	16	Realloc	Free
#6	31.550119s	0x55ef908a8c00 (+936)	32	16	Realloc	Free
#7	37.138924s	0x55ef908a8900 (+168)	64	16	Realloc	Free
#8	54.773293s	0x55ef908a8c30 (+984)	16	16	Realloc	Free
#9	55.693985s	0x55ef908a8950 (+248)	16	16	Realloc	Free

`apptest_allocators` allows testing all the operations supported by every allocator type

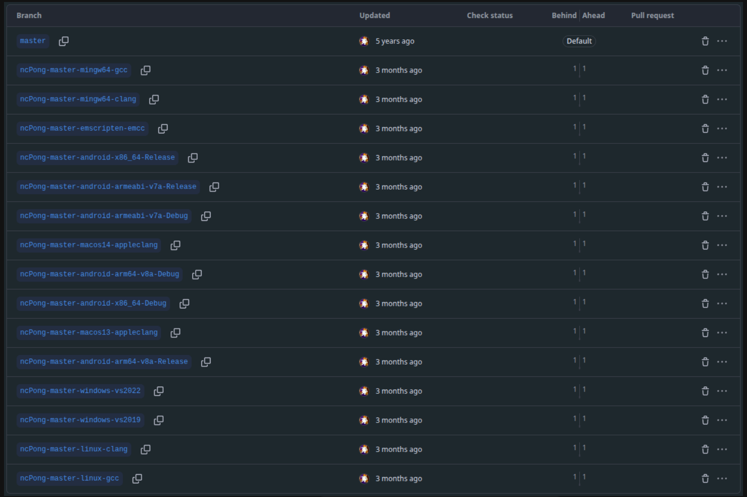
💊 2020 - Continuous Integration

- One YAML script per supported platform in `.github/workflows/`
- Projects upload a build artifact in the `projectName-artifacts` repository
  - This special repository has a branch per platform/compiler combination
- The `nCine` workflow also builds and uploads the documentations (C++ and Lua)
- The matrix of reproducible build combinations is impossible to manually test



Platform	Compiler	Build
Linux	GCC	
	Clang	
Windows	Visual Studio 2019	Release
	Visual Studio 2022	
MinGW	GCC	
	Clang	Debug
macOS 13	Apple Clang	
macOS 14	Apple Clang	
Android v7a	Clang	DevDist
Android v8a	Clang	
Android x86_64	Clang	
Emscripten	Clang	LuaDist

The combinations matrix



Branch	Updated	Check status	Behind	Ahead	Pull request
master	5 years ago		Default		
ncPong-master-mingw64-gcc	3 months ago		1	1	
ncPong-master-mingw64-clang	3 months ago		1	1	
ncPong-master-emscripten-emcc	3 months ago		1	1	
ncPong-master-android-x86_64-Release	3 months ago		1	1	
ncPong-master-android-armeabi-v7a-Release	3 months ago		1	1	
ncPong-master-android-armeabi-v7a-Debug	3 months ago		1	1	
ncPong-master-macos14-appleclang	3 months ago		1	1	
ncPong-master-android-arm64-v8a-Debug	3 months ago		1	1	
ncPong-master-android-x86_64-Debug	3 months ago		1	1	
ncPong-master-macos13-appleclang	3 months ago		1	1	
ncPong-master-android-arm64-v8a-Release	3 months ago		1	1	
ncPong-master-windows-vs2022	3 months ago		1	1	
ncPong-master-windows-vs2019	3 months ago		1	1	
ncPong-master-linux-clang	3 months ago		1	1	
ncPong-master-linux-gcc	3 months ago		1	1	

All the branches of the `ncPong-artifacts` repository

💊 2020 - Custom Allocators (1/3)

- The `IAllocator` interface declares functions for allocation, deallocation, and reallocation
    - Shrinking or expanding memory blocks via `reallocate()` can be faster on some allocators
- Main allocator types implemented:
    - *Linear*: only allocates new blocks, releases all at once
    - *Stack*: deallocates only in reverse order (last allocated first)
    - *Pool*: allocates fixed-size blocks, uses a free list for arbitrary deallocation
    - *Free List*: allocates and deallocates arbitrarily, can defragment adjacent free blocks
- Many libraries support custom allocators (e.g., SDL2, GLFW, Lua, ImGui, Nuklear, Vulkan)
- 📜 Reference: Tiago Sousa’s article

```
#ifdef _MSC_VER
    #pragma init_seg(".CRT$XCT")
#else
static AllocManagerInitializer allocManagerInit __attribute__((init_priority(101)));
#endif
```

Compiler-specific tricks to ensure correct initialization order of global objects

💊 2020 - Custom Allocators (2/3)

- It is possible to override the `new` / `new[]` and `delete` / `delete[]` operators
  - This way *all* process allocations will go through the custom allocator

```cpp
#ifdef OVERRIDE_NEW
void *operator new(std::size_t count)
{
    if (count == 0)
        return nullptr;

    return nctl::theDefaultAllocator().allocate(count);
}

void operator delete(void *ptr) noexcept
{
    if (ptr != nullptr)
        nctl::theDefaultAllocator().deallocate(ptr);
}
#endif
```

Custom `new` and `delete` operators

```cpp
#ifdef OVERRIDE_NEW
void *operator new[](size_t count)
{
    if (count == 0)
        return nullptr;

    return nctl::theDefaultAllocator().allocate(count);
}

void operator delete[](void *ptr) noexcept
{
    if (ptr != nullptr)
        nctl::theDefaultAllocator().deallocate(ptr);
}
#endif
```

Custom `new[]` and `delete[]` operators

💊 2020 - Custom Allocators (3/3)

- Using placement `new` to construct a `FreeListAllocator` inside a preallocated buffer
  - We can't heap-allocate the allocator itself if we want all allocations to go through it 🔄
  - One of the rare cases in C++ where the destructor must be called manually

```cpp
#define FREELIST_BUFFER (16777216) // 16 MB

static const unsigned int FreeListSize = FREELIST_BUFFER;
alignas(IAllocator::DefaultAlignment) static uint8_t freelistMemory[FreeListSize];
alignas(IAllocator::DefaultAlignment) static uint8_t freelistAllocatorBuffer[sizeof(FreeListAllocator)];
static FreeListAllocator &freelistAllocator = reinterpret_cast<FreeListAllocator &>(freelistAllocatorBuffer);

AllocManager::AllocManager()
{
    // ...
    new (&freelistAllocator) FreeListAllocator("Default", FreeListSize, freelistMemory); // placement new
}

AllocManager::~AllocManager()
{
    // ...
    (&freelistAllocator)->~FreeListAllocator(); // explicit call of the class destructor
}
```

Placement `new` in action (`src/base/AllocManager.cpp` 🔗)

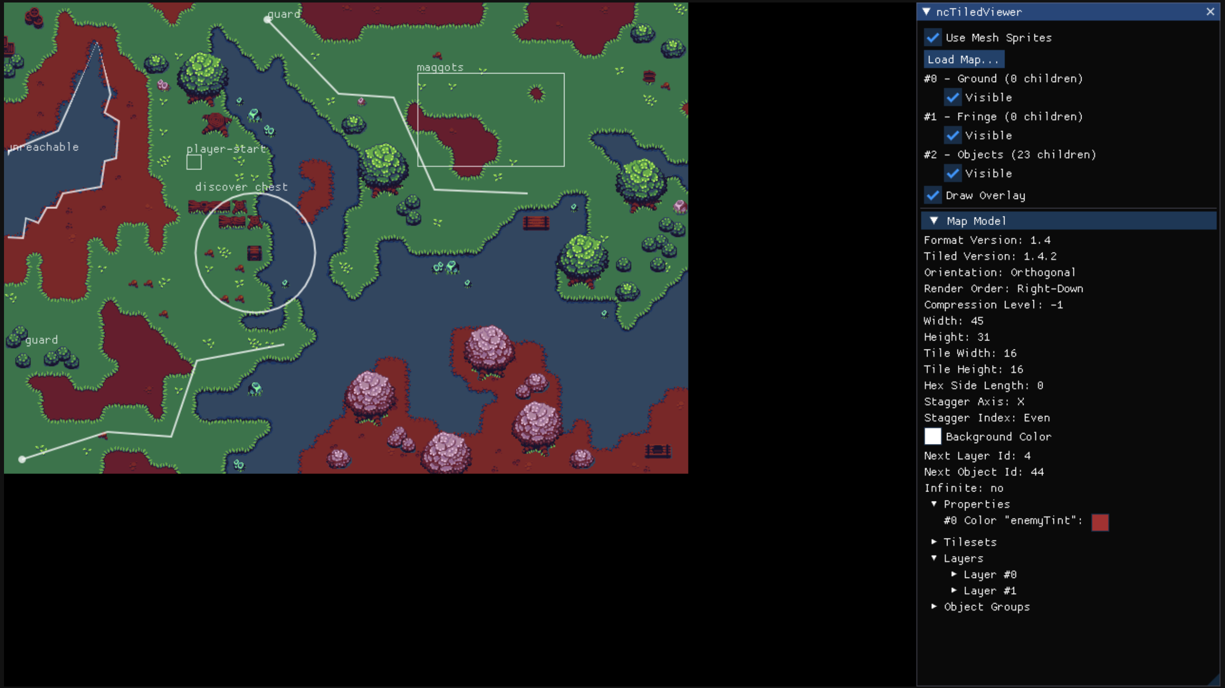
📅 2021 - Working From Home 🏠

- I joined The Multiplayer Group remotely as a Senior Rendering Engineer
  - They had no issues with me continuing my open-source contributions
  - I have worked on the Creation Engine 2 for Starfield
- I wrote a retrospective article about the first *ten years* and got interviewed in a podcast
- I bought a Raspberry Pi 4B and fixed minor build issues (also, SpookyGhost got some attention)
- Some parts of the site were moved to the GitHub Wiki
- Thinking about building an editor… 🤔 (#2)
  - Catch Lua errors with protected calls (📰 Dev Update 17)
- CMake project files moved inside the nCine distribution
- Lua oriented binary distribution (-○- 8ad63ad4, 📅 Jul)

My Raspberry Pi 4B 8GB, inside and Argon One M.2

📸 2021 - ncTiledViewer

- Creating a custom editor from scratch is a lot of work…
    - Started by simply showing how to load a map from the Tiled editor
- Added chroma key support for texture loading in nCine, required by some levels



▼ ncTiledViewer ✕

☑ Use Mesh Sprites

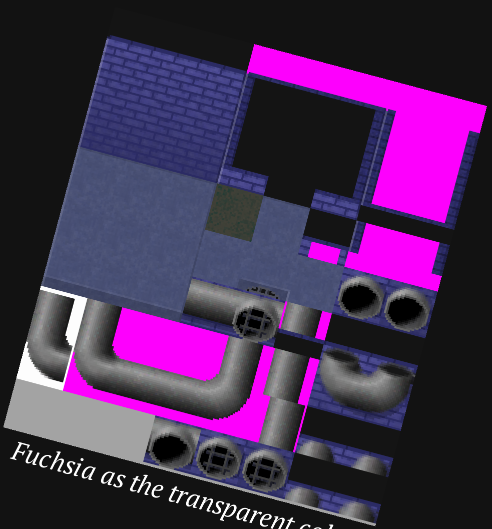
Load Map...

#0 - Ground (0 children)
☑ Visible
#1 - Fringe (0 children)
☑ Visible
#2 - Objects (23 children)
☑ Visible
☑ Draw Overlay

▼ Map Model

Format Version: 1.4
Tiled Version: 1.4.2
Orientation: Orthogonal
Render Order: Right-Down
Compression Level: -1
Width: 45
Height: 31
Tile Width: 16
Tile Height: 16
Hex Side Length: 0
Stagger Axis: X
Stagger Index: Even
⬜ Background Color
Next Layer Id: 4
Next Object Id: 44
Infinite: no
▼ Properties
#0 Color "enemyTint": 🟥
▸ Tilesets
▼ Layers
▸ Layer #0
▸ Layer #1
▸ Object Groups

***ncTiledViewer**, a Tiled TMX map format viewer (using **PugiXML**)*



Fuchsia as the transparent color

💊 2021 - CMake Scripts (1/2)

- More than 3600 lines of scripts in `nCine/cmake/`
  - Declare the list of public/private headers and sources
  - Expose a thorough set of compilation options to the user
  - Find the required and optional dependency libraries
  - Add compile definitions for optional features
  - Download ImGui, Nuklear, Tracy, GTest, and GBenchmark sources
  - Build the library, apptests, unit tests, benchmarks, and documentation
  - Build compressed archives, NSIS Windows installers, macOS bundles, Android APKs
- More than 1600 lines of scripts in `nCine/project/cmake`
  - Try to find the nCine library based on the `nCine_DIR` user variable
  - The user can customize some `NCPROJECT_` variables and some CMake callbacks

```
encelo@zephyrus ~/nCine $ cmake -S ncPong -B ncPong-build -D nCine_DIR=$PWD/nCine-build -D CMAKE_BUILD_TYPE=Debug
-- The C compiler identification is GNU 15.1.1
-- The CXX compiler identification is GNU 15.1.1
...
```

*Example of CMake invocation to build the **ncPong** example game*

💊 2021 - CMake Scripts (2/2)

```cmake
set(NCPROJECT_NAME "ncTemplate")
set(NCPROJECT_EXE_NAME "nctemplate")
set(NCPROJECT_VENDOR "Angelo Theodorou")
set(NCPROJECT_COPYRIGHT "Copyright ©2019-2021 ${NCPROJECT_VENDOR}")
set(NCPROJECT_DESCRIPTION "A template project for applications and games made with the nCine")
set(NCPROJECT_HOMEPAGE "https://ncine.github.io")
set(NCPROJECT_REVERSE_DNS "io.github.ncine.nctemplate")

set(NCPROJECT_INCLUDE_DIRS include)

set(NCPROJECT_SOURCES
  include/main.h
  src/main.cpp
)

# Don't edit beyond this line

# ...
# This part of the script is the same for all the projects and allows
# the scripts in `nCine/project/cmake` to perform their job.
# ...
```

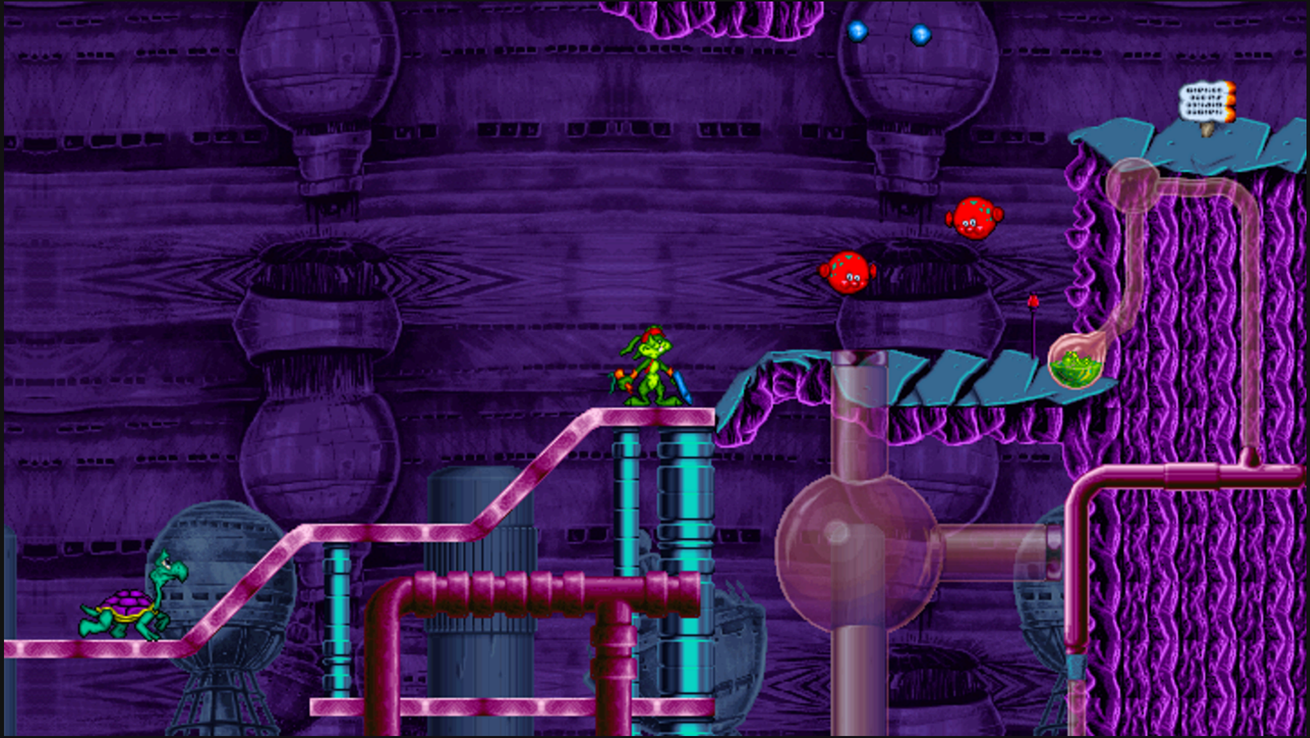
The `CMakeLists.txt` script from the ***ncTemplate*** *project*

📅 2022 - Split Screen and Post-processing

- Most of this year was spent on making advanced 2D graphics possible
  - Think of custom engine indie games like *Pathway* from Robotality or *Eastward* by Pixpil
- Merged the `viewports` branch (-o- 2fb00a58, 📅 Nov 2021 - Jan)
  - Added dirty bits to skip transformation and AABB regeneration (using `nctl::BitSet`)
- Merged the `custom_shaders` branch (-o- 4c306c80, 📰 Dev Update 19, 📅 Feb - Aug)
- DeathKiller ported his C# Jazz Jackrabbit 2 reimplementation to C++ with nCine (📅 Jul)
- Merged the `hidpi` branch (-o- 1874b56e, 📅 Sep - Dec)
  - Support for multiple monitors querying and windows scaling
  - Each desktop backend implements its own version of this new API

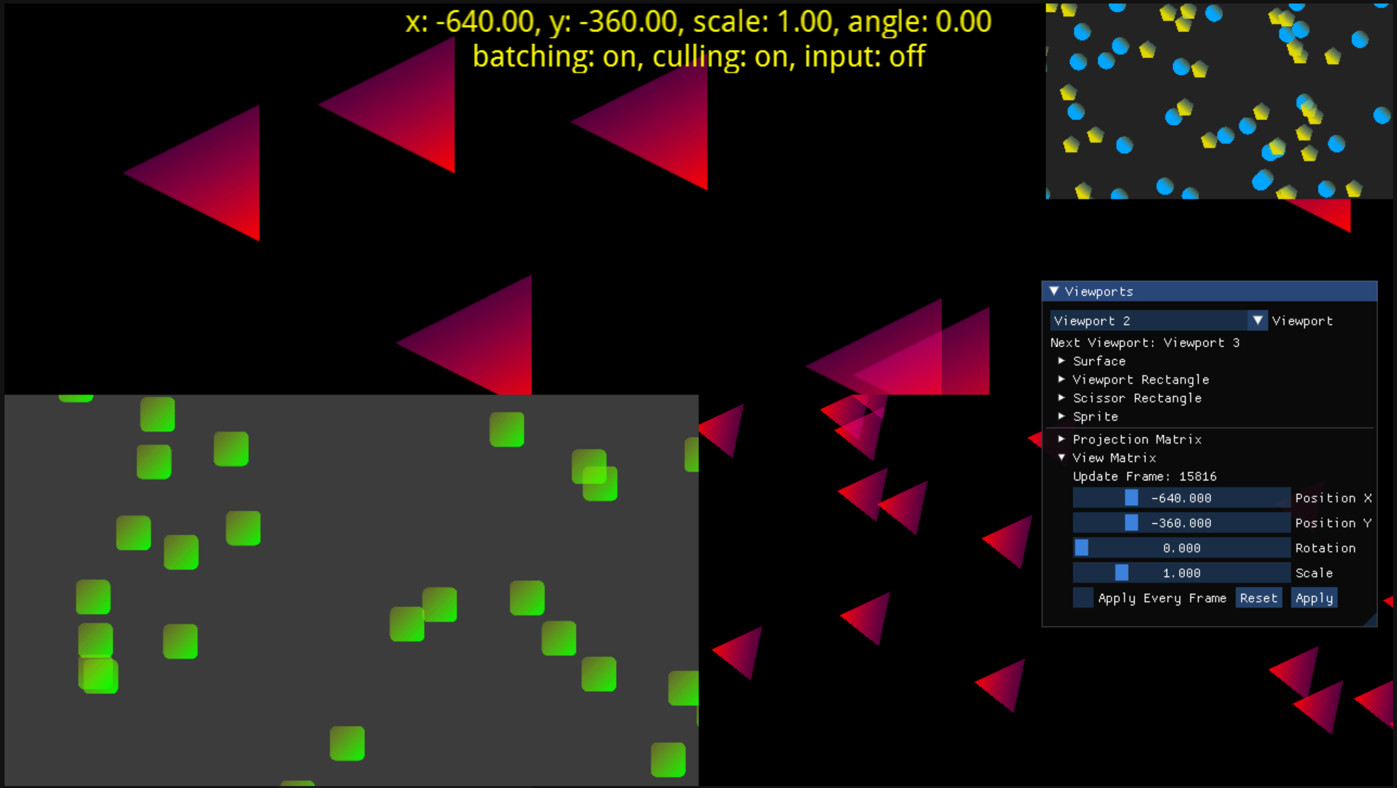
📸 2022 - Jazz² Resurrection

- Based on a custom fork of the nCine
  - The author provides me with feedback about the general nCine architecture



***Jazz² Resurrection**, an open-source reimplementation of Jazz Jackrabbit 2, the 1998 game by Epic MegaGames*

📸 2022 - Viewports Application Test



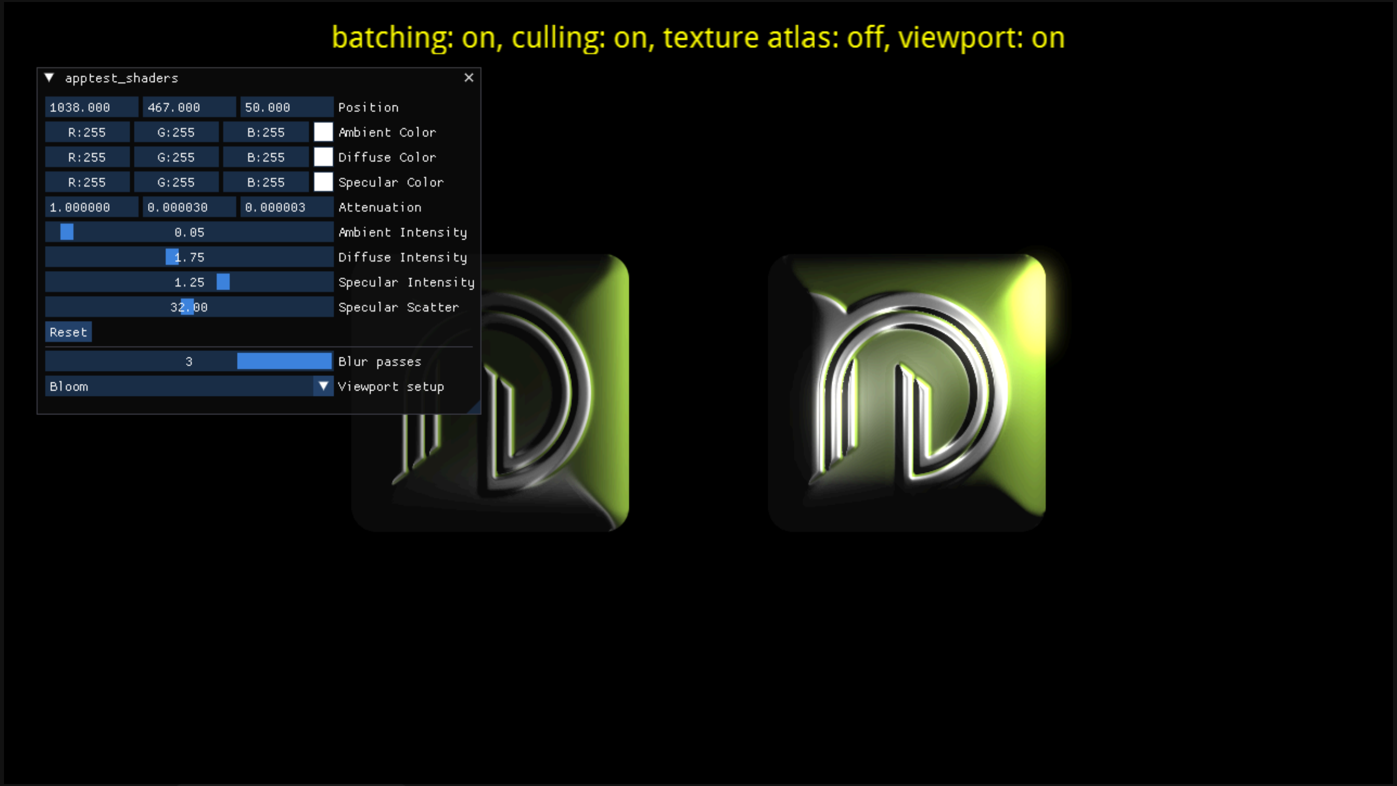
x: -640.00, y: -360.00, scale: 1.00, angle: 0.00
batching: on, culling: on, input: off

▼ Viewports

Viewport 2 ▼ Viewport
Next Viewport: Viewport 3
▸ Surface
▸ Viewport Rectangle
▸ Scissor Rectangle
▸ Sprite
▸ Projection Matrix
▼ View Matrix
Update Frame: 15816
-640.000 Position X
-360.000 Position Y
0.000 Rotation
1.000 Scale
☐ Apply Every Frame

`apptest_viewports` *showing different viewport, scene, and camera setups*

📸 2022 - Custom Shaders Application Test



batching: on, culling: on, texture atlas: off, viewport: on

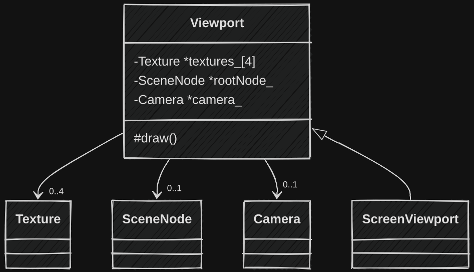
apptest_shaders

```
1038.000   467.000   50.000     Position
R:255      G:255     B:255      Ambient Color
R:255      G:255     B:255      Diffuse Color
R:255      G:255     B:255      Specular Color
1.000000   0.000030  0.000003   Attenuation
0.05                            Ambient Intensity
1.75                            Diffuse Intensity
1.25                            Specular Intensity
32.00                           Specular Scatter
Reset
3                               Blur passes
Bloom                           Viewport setup
```

`apptest_shaders` showing normal mapping and bloom while preserving automatic batching

💊 2022 - Viewports

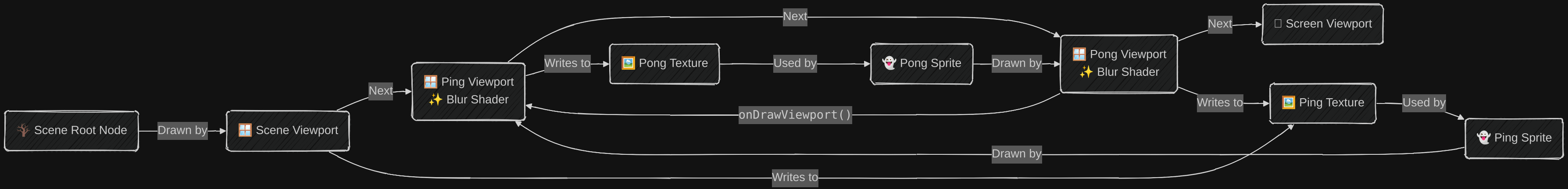
- Can change camera matrices and OpenGL scissor/viewport rectangles (useful for split screen)
- Can write to the screen or to one or more FBOs supporting Multiple Render Targets
- Can be chained together for multi-pass techniques
    - `apptest_shader` shows a blur made with a custom shader and a ping-pong technnique
    - The `onDrawViewport()` callback can be used to change shader parameters on ping-pong passes



Viewport class associations

Pass	Viewport	Input	Shader	Output
#1	Scene	Scene Root Node	Default	Ping Texture
#2	Ping	Ping Sprite	Blur V	Pong Texture
#3	Pong	Pong Sprite	Blur H	Ping Texture
#4	Ping	Ping Sprite	Blur V	Pong Texture
#5	Pong	Pong Sprite	Blur H	Ping Texture
#6	Screen	Ping Sprite	Default	Screen

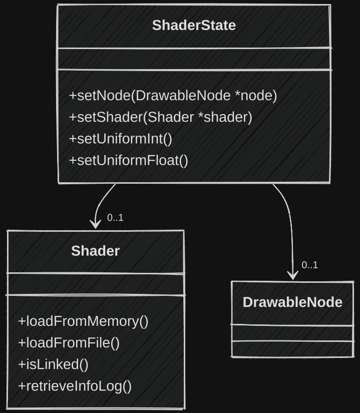
Multi-pass blur input/output table



Multi-pass blur flowchart

💊 2022 - Custom Shaders

- Users can load GLSL shaders from source (including for batching)
- Shaders can be assigned to multiple nodes via shader states
  - Shader states feed node-specific data to the shader during rendering
- They can work in tandem with viewports for post-processing effects
- The `Shader` class wraps an OpenGL shader program
  - Supports GLSL introspection of uniform variables
  - Users can replace just the fragment shader and use a built-in vertex shader, or vice versa



ShaderState

+setNode(DrawableNode *node)
+setShader(Shader *shader)
+setUniformInt()
+setUniformFloat()

0..1

Shader

+loadFromMemory()
+loadFromFile()
+isLinked()
+retrieveInfoLog()

0..1

DrawableNode

Associations of shader related classes

```cpp
void MyEventHandler::onDrawViewport(nc::Viewport &viewport)
{
    // Dirtying the uniform cache value at each blur pass
    if (&viewport == pingViewport_.get())
    {
        vpPingSpriteShaderState_->setUniformFloat(nullptr, "uDirection", 1.0f, 0.0f);
        vpPongSpriteShaderState_->setUniformFloat(nullptr, "uDirection", 0.0f, 1.0f);
    }
}
```

`apptest_shaders` *changing the separable direction uniform at each blur pass*

🗓️ 2023 - Faster Shaders, Smoother Scripting

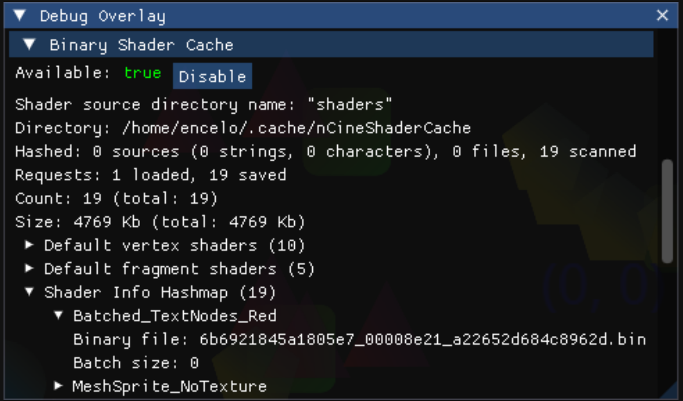
- Added a Binary Shader Cache, requested by DeathKiller to improve Xbox performance
  - UWP uses ANGLE, which is slow at translating GLSL shaders to HLSL
  - Caching precompiled shaders saves time, avoiding ANGLE translation
  - More information in Dev Update 20 📰
- Introduced alongside: double compilation of built-in batched shaders
  - First pass with `BATCH_SIZE = 1` introspects UBO limits
  - Second pass uses optimal size and is saved to the binary cache
  - Ensures compatibility on devices with less than 64kb in UBOs
- Published a Lua tutorial on the website (🗓️ Jan)
  - Yet another way to lower the entry barriers to use the framework
- Easier hot-reloading of Lua scripts (🗓️ Apr)

💊 2023 - Binary Shader Cache

- Saves binaries in the `fs::cachePath()/nCineShaderCache` directory by default
  - `%LocalAppData%` on Windows, `~/Library/Caches/` on macOS, `~/.cache` on Linux
- A load request match requires the same platform hash, binary format, and shader hash name
  - The platform hash is calculated from the `GL_RENDERER` and `GL_VERSION` strings
  - Driver updates change the platform hash, invalidating the cache
- The cache can prune outdated shaders automatically

uint64_t	uint32_t	uint64_t
platformHash	binaryFormat	shaderHashName

Key metadata for binary shader cache lookup

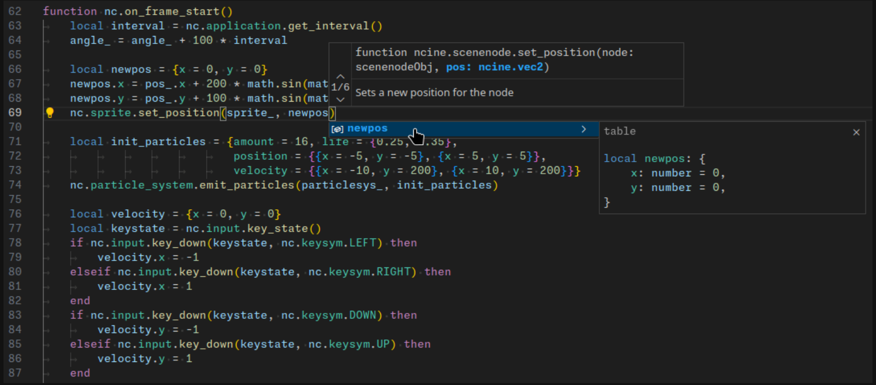


```
Available: true
Shader source directory name: "shaders"
Directory: /home/encelo/.cache/nCineShaderCache
Hashed: 0 sources (0 strings, 0 characters), 0 files, 19 scanned
Requests: 1 loaded, 19 saved
Count: 19 (total: 19)
Size: 4769 Kb (total: 4769 Kb)
 ▶ Default vertex shaders (10)
 ▶ Default fragment shaders (5)
 ▼ Shader Info Hashmap (19)
   ▼ Batched_TextNodes_Red
     Binary file: 6b6921845a1805e7_00008e21_a22652d684c8962d.bin
     Batch size: 0
   ▶ MeshSprite_NoTexture
```

Binary Shader Cache section in the debug overlay interface

📅 2024 - GitHub Recognition

- Presented in Granada about why custom engines matter (📅 Mar)
- Published some "getting started" guides on the GitHub Wiki, to reach more users (📅 Mar)
- Surpassed 1,000 stars on GitHub ⭐ (📅 Jun)
- Merged the `openal_efx` branch (📰 Dev Update 21, 📅 May - Jun)
- Started developing a multi-threaded job system (`job_system`, 📅 May - Jul)
- nCine became an official addon for the LuaLS extension in VS Code (📅 Nov)
  - Offers autocomplete, type checking, and full API documentation in the IDE
- Released a new LDoc documentation



```lua
function nc.on_frame_start()
    local interval = nc.application.get_interval()
    angle_ = angle_ + 100 * interval

    local newpos = {x = 0, y = 0}
    newpos.x = pos_.x + 200 * math.sin(mat
    newpos.y = pos_.y + 100 * math.sin(mat
    nc.sprite.set_position(sprite_, newpos)

    local init_particles = {amount = 16, life = {0.25, .35},
                            position = {{x = -5, y = -5}, {x = 5, y = 5}},
                            velocity = {{x = -10, y = 200}, {x = 10, y = 200}}}
    nc.particle_system.emit_particles(particlesys_, init_particles)

    local velocity = {x = 0, y = 0}
    local keystate = nc.input.key_state()
    if nc.input.key_down(keystate, nc.keysym.LEFT) then
        velocity.x = -1
    elseif nc.input.key_down(keystate, nc.keysym.RIGHT) then
        velocity.x = 1
    end
    if nc.input.key_down(keystate, nc.keysym.DOWN) then
        velocity.y = -1
    elseif nc.input.key_down(keystate, nc.keysym.UP) then
        velocity.y = 1
    end
```

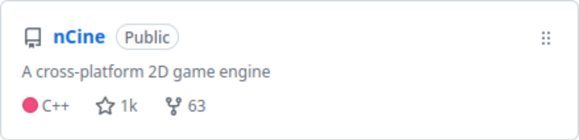
```
function ncine.scenenode.set_position(node: scenenodeObj, pos: ncine.vec2)
```

1/6

Sets a new position for the node

```
table

local newpos: {
    x: number = 0,
    y: number = 0,
}
```

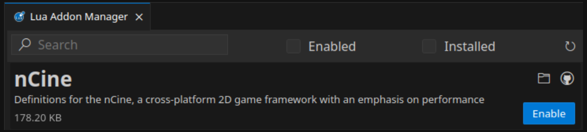
Lua Language Server support in VS Code



nCine Public

A cross-platform 2D game engine

One thousand stars and counting!



nCine

Definitions for the nCine, a cross-platform 2D game framework with an emphasis on performance

178.20 KB

nCine among the addons of the LuaLS

💊 2024 - Job System (🚧 WIP) (1/3)

⚠️ GitHub links don't work as the code has not been pushed yet. ⚠️

- One thread spawned per logic core at program start (thread pool)
- Lock-free work stealing queues per thread for automatic load-balancing
  - Lock-free achieved with Compare-And-Swap operations on atomics
- Parent/children relationship enables waiting on parent jobs
- As soon as a job finishes, all of its *continuations* jobs run immediately
- Built-in `parallelFor(dataArray, 4096, &myDataFunc, nc::CountSplitter(128))`
- 📜 Reference: Molecular Matters blog articles

```
struct Job
{
    JobFunction function;
    Job *parent = nullptr;
    nctl::Atomic32 unfinishedJobs;
    char data[JobDataSize];
    nctl::Atomic32 continuationCount;
    Job *continuations[JobNumContinuations];
};
```

The job system pillar: the `Job` *structure (*`src/include/Job.h` 🔗*)*
To avoid false sharing, it should occupy at least one cache line.

```
while (true)
{
    while (!getJob(jobQueues) && shouldQuit == false)
    {
        queueMutex.lock();
        queueCV.wait(queueMutex);
        queueMutex.unlock();
    }
    if (shouldQuit) break;
    execute(job, jobQueues);
}
```

Main loop of a thread function (`src/threading/JobSystem.cpp` 🔗*)*

💊 2024 - Job System (🚧 WIP) (2/3)

- Jobs can spawn other jobs, queued on the same thread's queue (this is how `parallelFor` works)
- The system is paired with a `LogEntryQueue` class for multi-threading logging

```cpp
/// The thread id for each thread
static inline thread_local unsigned char threadId_;
```

Using C++11 `thread_local` keyword for Thread Local Storage (TLS) (`include/ncine/IJobSystem.h` 🔗)

```cpp
void finish(Job *job, JobQueue *jobQueues)
{
    const int32_t unfinishedJobs = --job->unfinishedJobs; // atomic decrement
    if (unfinishedJobs == 0)
    {
        // Releasing the job back to the pool.
        job->function = nullptr;
        if (job->parent)
            finish(job->parent, jobQueues);
        // run follow-up jobs
        for (int i = 0; i < job->continuationCount; i++)
            jobQueues[JobSystem::threadId()].push(job->continuations[i]);
    }
}
```

Finishing a job, signalling the parent, and running continuations (`src/threading/JobSystem.cpp` 🔗)

💊 2024 - Job System (🚧 WIP) (3/3)

```cpp
// If set to 1, the `JobId` will be a 16bit integer that encodes the thread id (5 bits) and the pool index (11 bits)
// A packed id will limit the number of threads to 32, and the pool size to 2048.
// Set it to 0 to use a 64bit pointer and overcome those limits (less space in the `Job` struct for remaining fields).
#define PACKED_JOBID (0) // Experimental!

#include <cstdint>
#include "common_defines.h"

namespace ncine {

#if PACKED_JOBID
using JobId = uint16_t;
#else
using JobId = uintptr_t;
#endif
using JobFunction = void (*)(JobId, const void*);
```

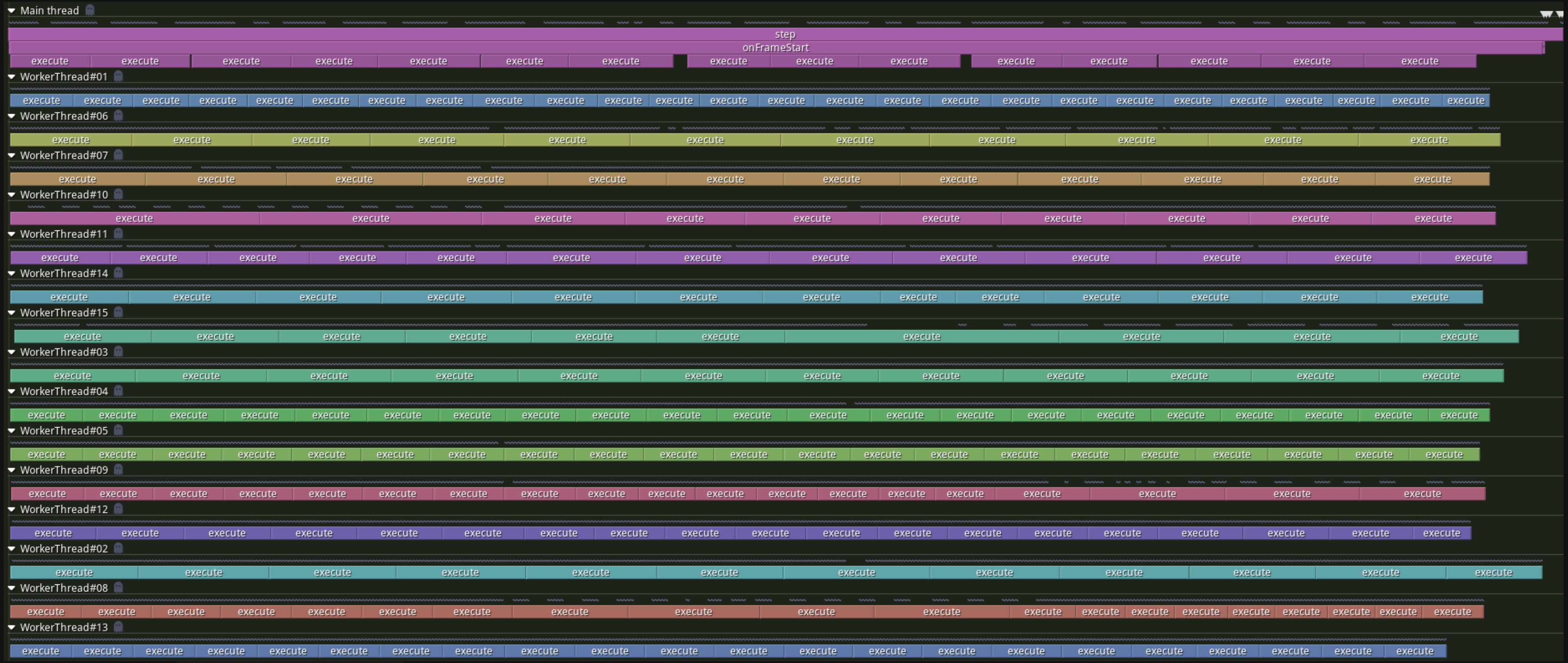
Users are not exposed to raw pointers. `JobId` *is a numeric, opaque identifier.*

When packed, it functions as a real handle encoding both the queue index (thread id) and the element index.

From: `include/ncine/IJobSystem.h` 🔗

📸 2024 - Threads Captured in Tracy

- The main thread and all worker threads are actively processing jobs in the `execute()` function
- Jobs are initially queued on the main thread, then distributed across workers via work stealing

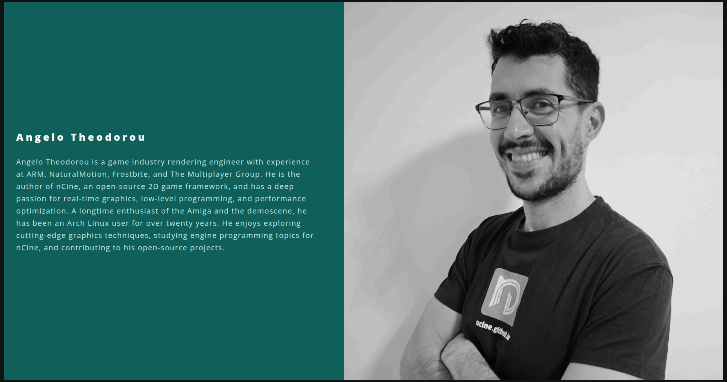


Tracy capture of `apptest_jobsystem`*, showing jobs execution distributed among worker threads*

🗓️ 2025 - Let’s Try Some Different Things

- Industry layoffs finally hit me, currently job-free but time-rich ⏳ (🗓️ Mar)
- I made an nCine game at the Global Game Jam (🗓️ Jan)
    - I also sponsored the event and gave away an nCine mug as a prize 🍺
- Tried Google and Reddit advertisement with a very small budget
- ChatGPT suggested I contact Valve for a collaboration (no reply 😄)
- Updated GitHub `README.md` with documentation links and screenshots
- Applied for conferences (Guadalindie in Malaga 👎, and /dev/games in Rome 👍)
- Switched to introsort for `RenderCommand` sorting (🗓️ Jan)

The nCine mug prize



Angelo Theodorou

Angelo Theodorou is a game industry rendering engineer with experience at ARM, NaturalMotion, Frostbite, and The Multiplayer Group. He is the author of nCine, an open-source 2D game framework, and has a deep passion for real-time graphics, low-level programming, and performance optimization. A longtime enthusiast of the Amiga and the demoscene, he has been an Arch Linux user for over twenty years. He enjoys exploring cutting-edge graphics techniques, studying engine programming topics for nCine, and contributing to his open-source projects.

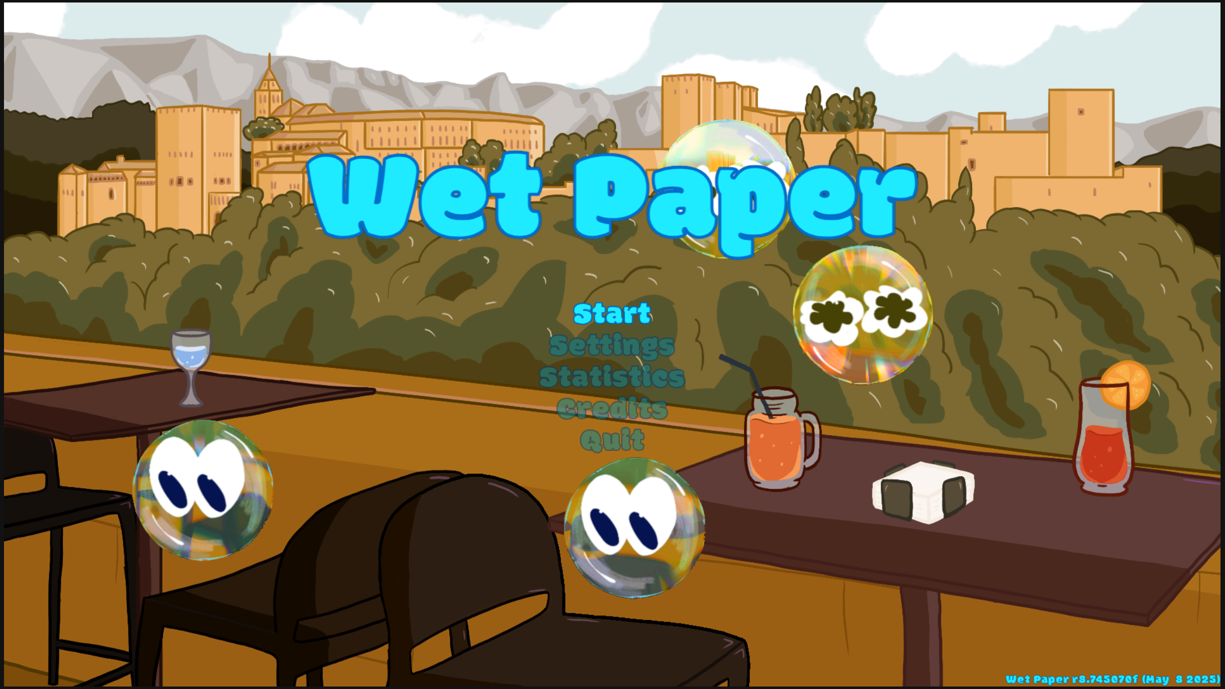
My presentation card for /dev/games 2025

📸 2025 - The GGJ 2025 in Granada

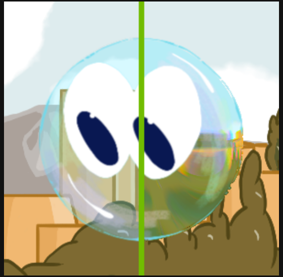
Invited on stage to award a prize, nCine is an official sponsor

📸 2025 - Wet Paper

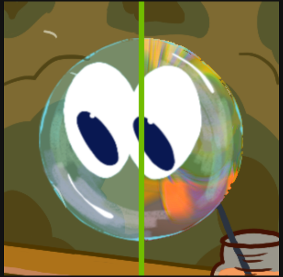
- The GGJ game is still in development and will be a new showcase for users
  - With custom shaders, statistics, load/save settings in TOML format, joystick vibration
  - A dogfooding experience to make nCine better (remember the Blender Open Movies?) 🐶



Wet Paper with the custom refraction shader for bubbles



Refraction off/on (#1)



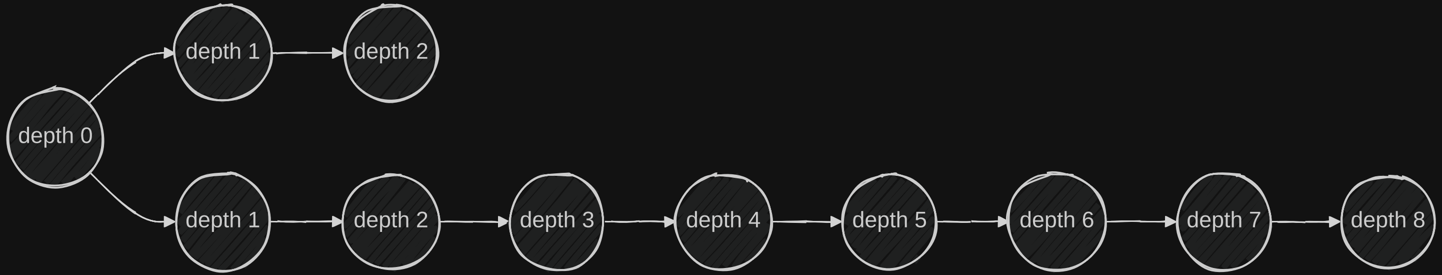
Refraction off/on (#2)

💊 2025 - Introspective Sort (1/2)

- Introsort is a **hybrid** sorting algorithm (also used by `std::sort`)
  - Fast with very few elements (uses *insertion sort* below a threshold)
  - Fast on average, as it uses *quicksort* partitioning
  - Optimal worst-case performance (switches to *heapsort* when recursion is too deep)

```cpp
template <class Iterator, class Compare>
inline void sort(Iterator first, Iterator last, Compare comp)
{
    const unsigned int maxDepth = log(distance(first, last)) * 2;
    introsort(first, last, comp, maxDepth);
}
```

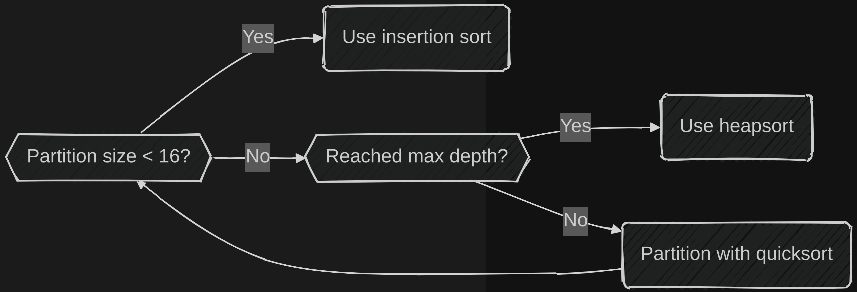
The sort function calculates the maximum depth for quicksort



Some pivot selections lead to unbalanced partitions, deep recursion, and a worst-case time complexity of O(n^2)

💊 2025 - Introspective Sort (2/2)

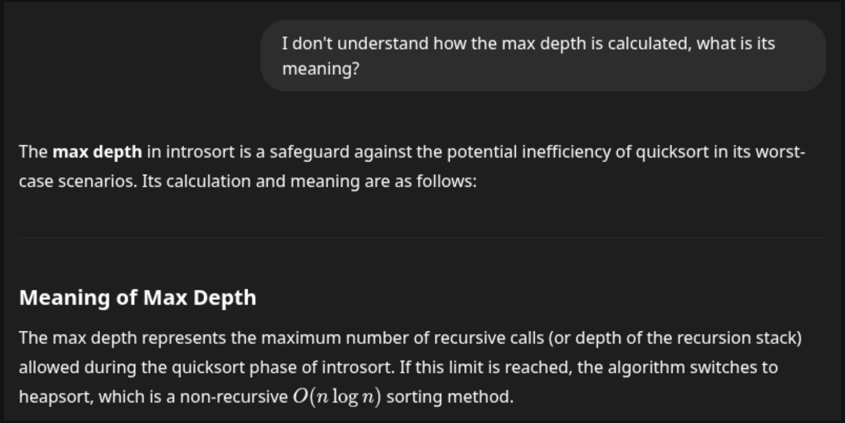
```
/// Introspective sort implementation with iterators and custom compare function
template <class Iterator, class Compare>
inline void introsort(Iterator first, Iterator last, Compare comp, unsigned int maxDepth)
{
    const int size = distance(first, last);
    if (size < 16)
        insertionsort(first, last, comp);
    else if (maxDepth == 0)
        heapsort(first, last, comp);
    else
    {
        Iterator p = prev(last);
        swap(*next(first, size / 2), *p);
        Iterator q = partition(first, p, comp);
        swap(*q, *p);
        introsort(first, q, comp, maxDepth - 1);
        introsort(next(q), last, comp, maxDepth - 1);
    }
}
```


Partition size < 16?
Yes
Use insertion sort
No
Reached max depth?
Yes
Use heapsort
No
Partition with quicksort

Note that introsort is called recursively, switch conditions are checked per each quicksort partition

🗓️ 2025 - What About AI? 🤖

- Brainstormed game, business, and collaboration ideas
- Assisted me in writing a quicksort unit test for nCine
- Helped me proofread this presentation 👨‍🏫
  - Wrote bash and Python scripts to create charts about Git activity
  - Explained how to write custom CSS code for Slidev
- Helped setup the refraction shader in 2D for Wet Paper



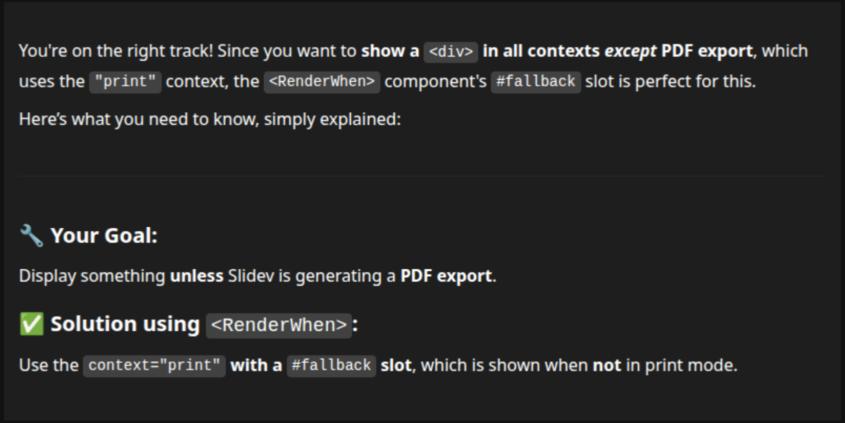
I don't understand how the max depth is calculated, what is its meaning?

The **max depth** in introsort is a safeguard against the potential inefficiency of quicksort in its worst-case scenarios. Its calculation and meaning are as follows:

Meaning of Max Depth

The max depth represents the maximum number of recursive calls (or depth of the recursion stack) allowed during the quicksort phase of introsort. If this limit is reached, the algorithm switches to heapsort, which is a non-recursive $O(n \log n)$ sorting method.

ChatGPT helping me understand introsort



You're on the right track! Since you want to **show a `<div>` in all contexts *except* PDF export**, which uses the `"print"` context, the `<RenderWhen>` component's `#fallback` slot is perfect for this.

Here's what you need to know, simply explained:

🔧 **Your Goal:**

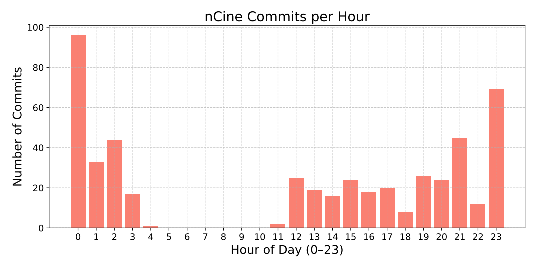
Display something **unless** Slidev is generating a **PDF export**.

✅ **Solution using `<RenderWhen>`:**

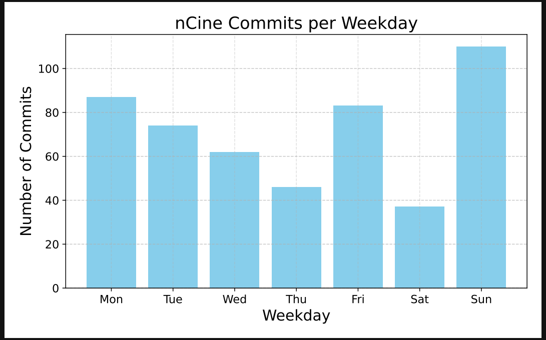
Use the `context="print"` **with a `#fallback` slot**, which is shown when **not** in print mode.

ChatGPT helping me with Slidev

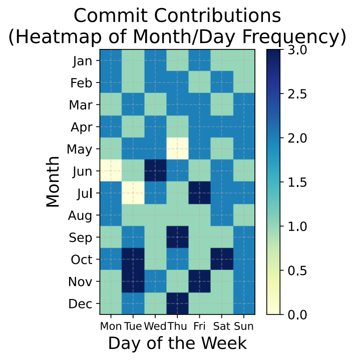
📊 Some Git Statistics



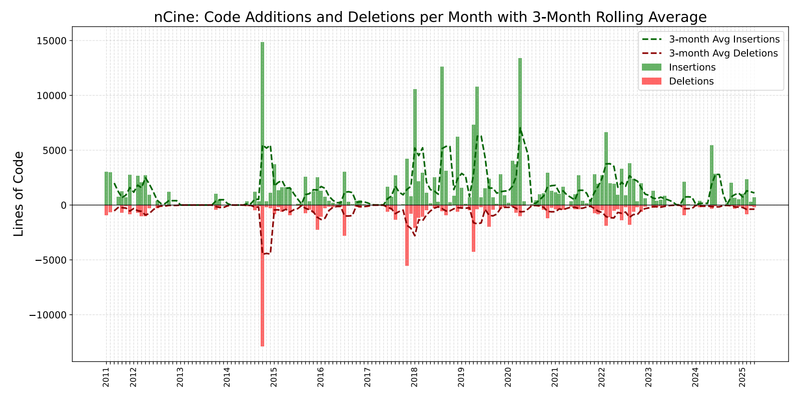
Commits per Hour (yes, I'm a 🧛)



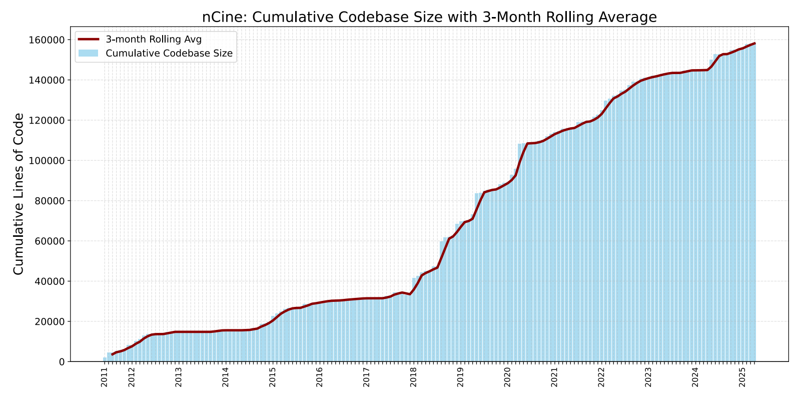
Commits per Weekday



Commit Contributions Heatmap



Additions/Deletions Over Time



Cumulative Codebase Histogram

🚢 A Classic Sunk Cost Fallacy?

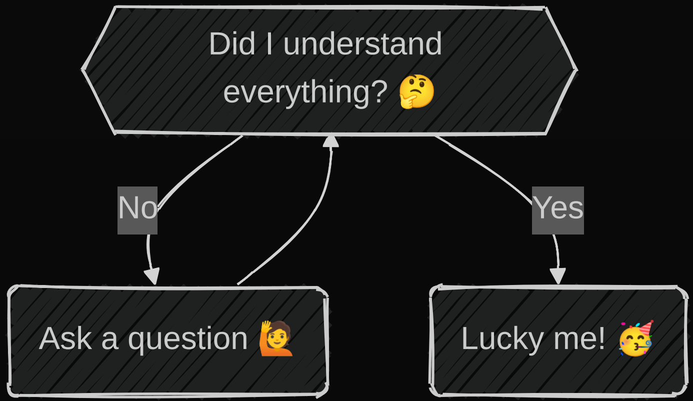
- It might seem like I’m just refusing to let the ship sink…
    - I’ve spent so many years on this, I *have* to see it through!
    - If I stop now, it will all have been for nothing! 😭
- But fortunately, that’s not the reality
    - I have poured in a lot of time, but I’ve also learned immensely and enjoyed the journey
    - It might never become my full-time job, and that’s fine
    - Just having more users and watching the project grow is already rewarding

🔮 Future Work

- Finish *incomplete* tasks:
  - Complete the job system, then parallelize engine parts with Data Oriented Design
  - Test and finalize the CrashPad integration
- Support more *technologies*:
  - Add a unified graphics layer supporting OpenGL, Vulkan, Metal, and WebGPU backends
  - Switch to SDL3 as the new default desktop backend
  - Port to iOS (requires Metal support first)
- Explore *new* and ambitious projects:
  - Develop a C++ neural network library for games and experiments (NEAT and genetic algorithms)
  - Implement a raylib-compatible API on top of nCine to attract new users 🤯
  - Build a fully-fledged ImGui editor with a runtime scene "player" (like Unity/Godot)
- Return to existing projects and *to-do notes*:
  - Revisit ncTracer for continuous learning and to stay sharp in graphics
  - Add new features to SpookyGhost: particles, timeline, batch processing

Any Questions?

Feel free to try it out and have fun tinkering! ⚗️
For any questions, reach me at: encelo@gmail.com 📧



Did I understand everything? 🤔

No

Yes

Ask a question 🙋

Lucky me! 🥳

💊 List of Technical Pills

- Monotonic Clocks
- OpenGL 2 Renderer
- Atomic Counters
- C++ 11
- OpenGL 3.3 Renderer
- nCine Template Library
- Small String Optimization
- Leapfrog Probing
- Continuous Integration
- Custom Allocators
- CMake Scripts
- Viewports
- Custom Shaders
- Binary Shader Cache
- Job System
- Introspective Sort