

1. Project Overview

Team name: DAWdle, a digital audio workstation with a built-in node-based scripting language.

Github Repo Link: [ChicoState/DAWdle \(github.com\)](https://github.com/ChicoState/DAWdle)

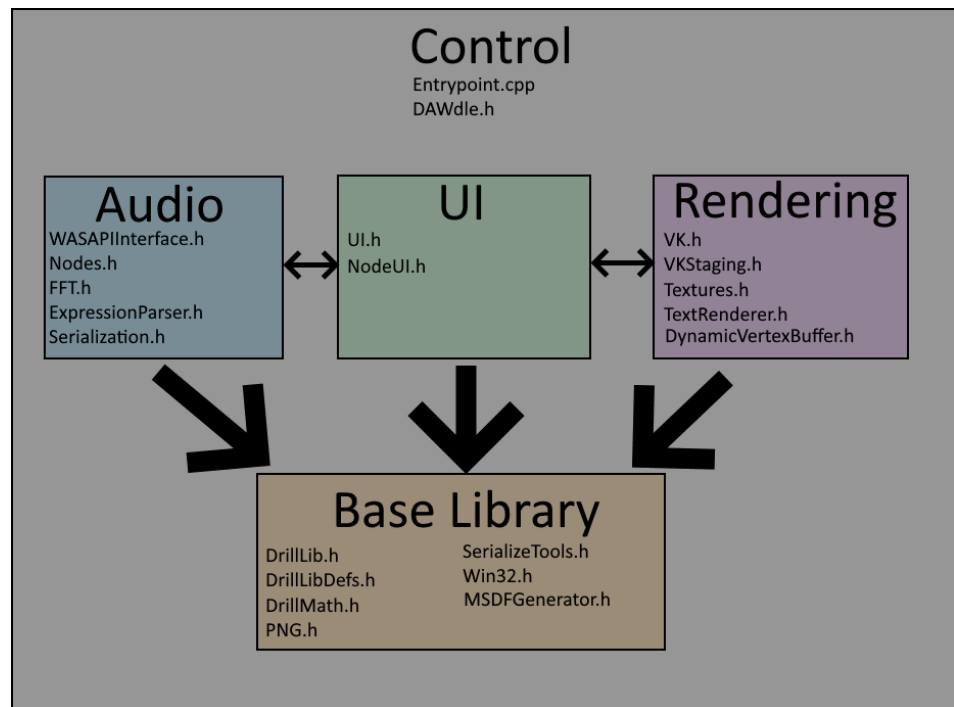
Project Kanban board: [Kanban board with Backlog · DAWdle Kanban \(github.com\)](https://github.com/ChicoState/DAWdle/Kanban)

Tech stack: C++, using Vulkan for rendering (previously C++ with Qt)

2. Software Design

- I. Software Architecture. *Illustrate* and provide a brief explanation of the software's architecture. There is not a formal UML format for architecture, but for example, you can draw which modules belong to the data model separated from those that belong to the view, etc. If you follow a specific architecture (e.g. MVC, MVVM, etc.) name it and explain how your design exemplifies that model.

At a high level, this project is split into three major components, Audio, UI, and rendering, with a shared base library layer. The top level file, DAWdle.h, ties the components together and manages the program main loop.



The project uses a single compilation unit in order to reduce build friction and lessen compilation time (theoretically having multiple compilation

units helps with compile time, but in practice that often isn't the case), so every file is a header that's included into Entrypoint.cpp for compilation. Normally you would include cpp files and run the compiler on only one cpp file, but we used headers in order to avoid Visual Studio automatically compiling every cpp file. So, FileName.h is what you'd normally consider a cpp file, and FileName_decl.h is what you'd normally consider a header file (though in this project it's only for resolving circular file dependency issues).

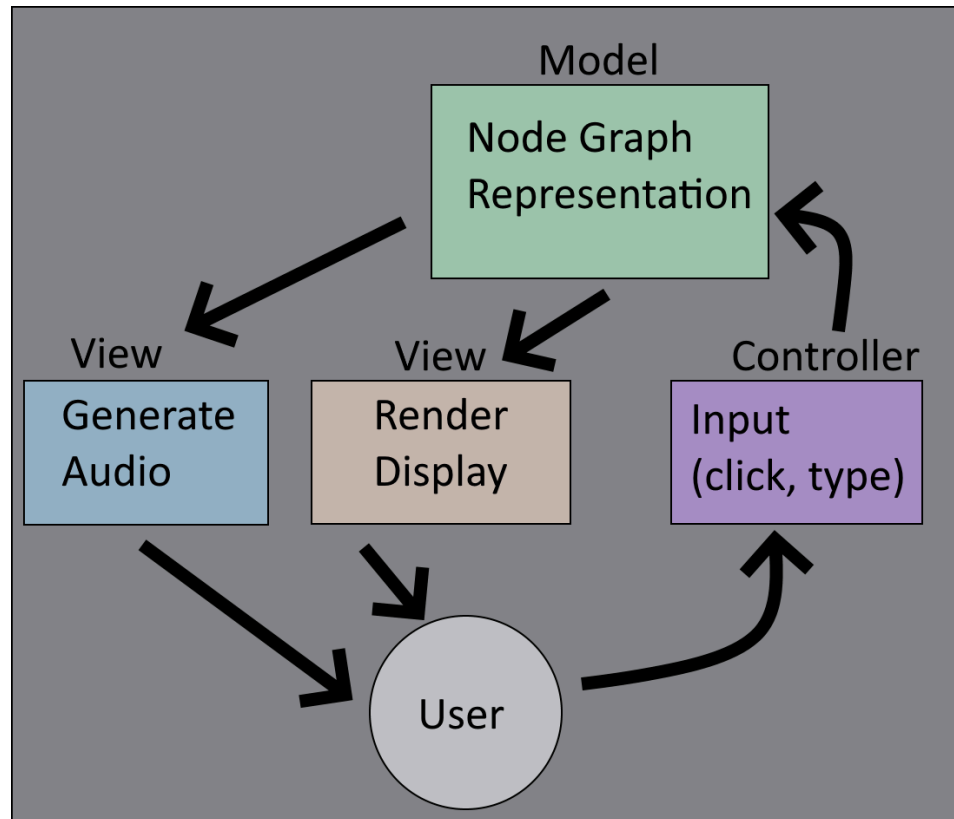
We use a simple two threaded model, one thread for UI and rendering and the other for audio. The only reason audio is on a separate thread at all is so that rendering doesn't block it and cause issues if the frame rate is too low or VSync is enabled.

Audio Component:

This project exclusively uses WASAPI on Windows to output audio. The audio thread wakes up when half the audio buffer should be empty and calls into the node graph to generate however many audio samples are needed to fill it up to full. For performance reasons, the audio graph always processes audio buffers in blocks of 1024 samples, saving any leftover samples for the next time the audio thread wakes up and needs more data.

The audio graph was originally designed to be a function of time, as in "for x time, what is the audio output value?". In hindsight, while this did make processing very vectorizable, it also puts a lot of limitations on the user, since they have to do everything as a function of time now instead of sequentially, which is much easier to think about for certain effects. Effects such as reverb also don't tend to work well as a pure function of time, since they require information from the past.

The audio graph itself uses an MVC architecture. The NodeGraph class acts as the model, holding all data related to the graph. That model is then viewed by both the audio thread for generating output and the render component for drawing the nodes on screen. The UI system acts as the controller, providing user input for changing the node graph.



UI Component:

The UI system is a retained mode GUI based around boxes. Each box represents a single UI element, such as a text string, or a button. Every box has the necessary fields for everything any box could do. While this increases memory usage, it also provides a lot of flexibility, since if you want a clickable box with an outline and a specific hover cursor and a particular tooltip, you don't need to make a new class for that, you just set the necessary fields in the box to enable each bit of functionality.

To provide custom behavior, boxes can define an action callback which gets called whenever something interesting happens to the box (such as the box rendering, or the user clicking on it). These action callbacks take a box and a `UserCommunication` struct as input. The `UserCommunication` struct contains all the information being communicated through the callback, such as whether the box was left clicked, or what key was pressed while the box was selected, or what the current mouse position is. While we could have had different callbacks for each possible action, I (Evan) personally found it to be much simpler to have only one callback that handles everything.

Each frame, the UI system computes a dynamic layout for each box, which automatically resizes when necessary. This means the programmer

does not have to manually specify coordinates for each box.

Render Component:

The renderer is designed using the Vulkan graphics API. Every frame, the UI system puts triangle data into a CPU side vertex buffer, which is then read by the GPU to draw the UI. A bindless architecture is used for rendering, with every texture stored in a big array. The whole UI is drawn in a single draw call.

Vulkan may not have been the optimal API to use for this project, since it's geared toward extremely high performance large scale renderers, not flat color UIs that maybe draw a couple textures. Vulkan tries to do as much as it can at load time so that performance during rendering is very fast, which meant that startup time took significantly (around 0.2 seconds) longer, and memory usage was significantly higher (several hundred megabytes of overhead for initializing Vulkan). Power usage was also worse since the UI is redrawn each frame at a high frame rate. While I (Evan) personally like Vulkan quite a lot, it wasn't the right choice for a project that does zero complex rendering. A basic CPU side software rasterizer with no graphics API may have been a better choice.

Base Library:

This project originally compiled without using the C or C++ standard library at all, which meant a good deal of functionality had to be rewritten from scratch. After more people started using it, the standard library was added back in for their convenience, making much of this base library unnecessary. However, I (Evan) actually like many of my standard systems more than the C/C++ standard ones (they tend to be simpler, or more performant, or have a nicer interface than their counterparts), so I continued to use the base library for most of my work.

The base library (DrillLib.h, DrillLibDefs.h) includes things like common constants, conversion macros, data structures (array lists and strings), serialization helpers, functions for printing to the console, reading and writing files, running external programs, and memory allocators. There is also a math library (DrillMath.h) containing things like small vectors, matrices, and quaternions suitable for computer graphics, trig functions (sin/cos/atan2 etc. Note that angles in this project are defined in terms of turns, not radians or degrees. I (Evan) firmly believe that turns are a much nicer unit for angles in practice, and tend to remove a lot of multiplies by 2π), floating point helpers (fract, round, ceil, etc), variadic min/max functions, bezier curve functions, and more.

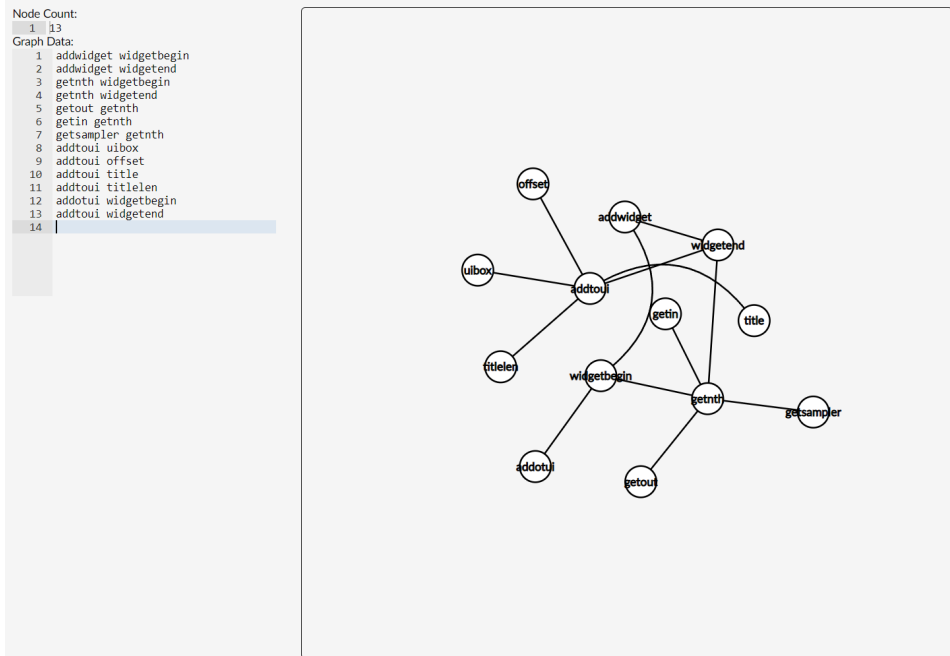
The project is based heavily around arena allocation, which is a type of very fast memory allocator that makes it easy to manage large groups of objects with the same lifetime (you don't have to track object lifetimes individually, once you're done with a set of objects with the same lifetime you can free them all at once just by resetting a single allocation offset). The audio system uses arena allocators heavily for evaluating audio buffers. The arena allocators in this particular project reserve a gigabyte of memory, but importantly do not actually commit this memory until it is used (we use a page fault handler to commit more memory if the program faults on access to an arena). This means that we can have a big linear block of memory that only consumes as much actual memory as we need. There are several smaller components to the base library, including a PNG reader for loading textures, an MSDF generator for generating signed distance fields for icons and fonts, an OS layer for communicating with Windows and providing some degree of separation to make it easier to refactor into OS independence, and a selection of serialization tools for doing things like parsing text and converting floating point numbers to ASCII.

The project was originally designed with C style C++ in mind, so no constructors/destructors, no RAII, no new/delete, no move semantics. I (Evan) consider modern C++ memory features to be a real pain to program with, so nearly everything is a simple POD struct. This means no special constructors have to be defined, and copying or assigning a struct always behaves predictably. It also means memory can be moved around as a raw memory block without concern for the C++ object model. Lifetimes of objects are also typically managed either in bulk with arenas or with block allocators, no smart pointers to individually manage object lifetimes.

- II. Module/Class Design. *Illustrate* (using Class Diagrams) your classes. If you have more classes that can legibly fit onto one page, you can choose to only illustrate the most important classes. Analyze their *cohesion* and interpret the result of those analyses.

Below is a UML class diagram showing the relationships between the most important classes in the project. These classes include UI widgets (which define how a node should be rendered), the nodes themselves (including the actual logic for how a node should process audio), and the node graph, which is in charge of storing/managing the nodes and running the data through the nodes and sending the result to WASAPI. We were

unable to find any static analysis tool for calculating the LCOM4 automatically so we had to do it by hand, and thus decided to only do so for the largest and most important classes (NodeGraph and NodeHeader) since it matters more if they have poor cohesion compared to a smaller and less central class. We calculated $LCOM4 = 1$ for NodeHeader and $LCOM4 = 1$ for NodeGraph (note that these calculations did not take init or destroy into account for the same reason we do not typically consider constructors in LCOM4). These graphs can be seen below. Both of these results are ideal and indicate healthy and cohesive class design.

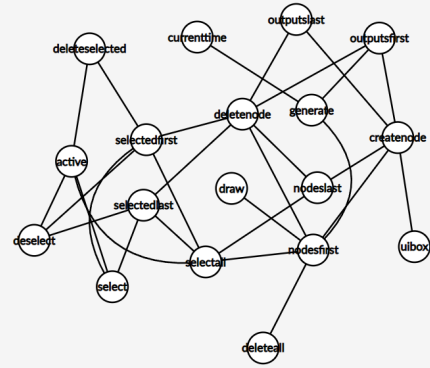


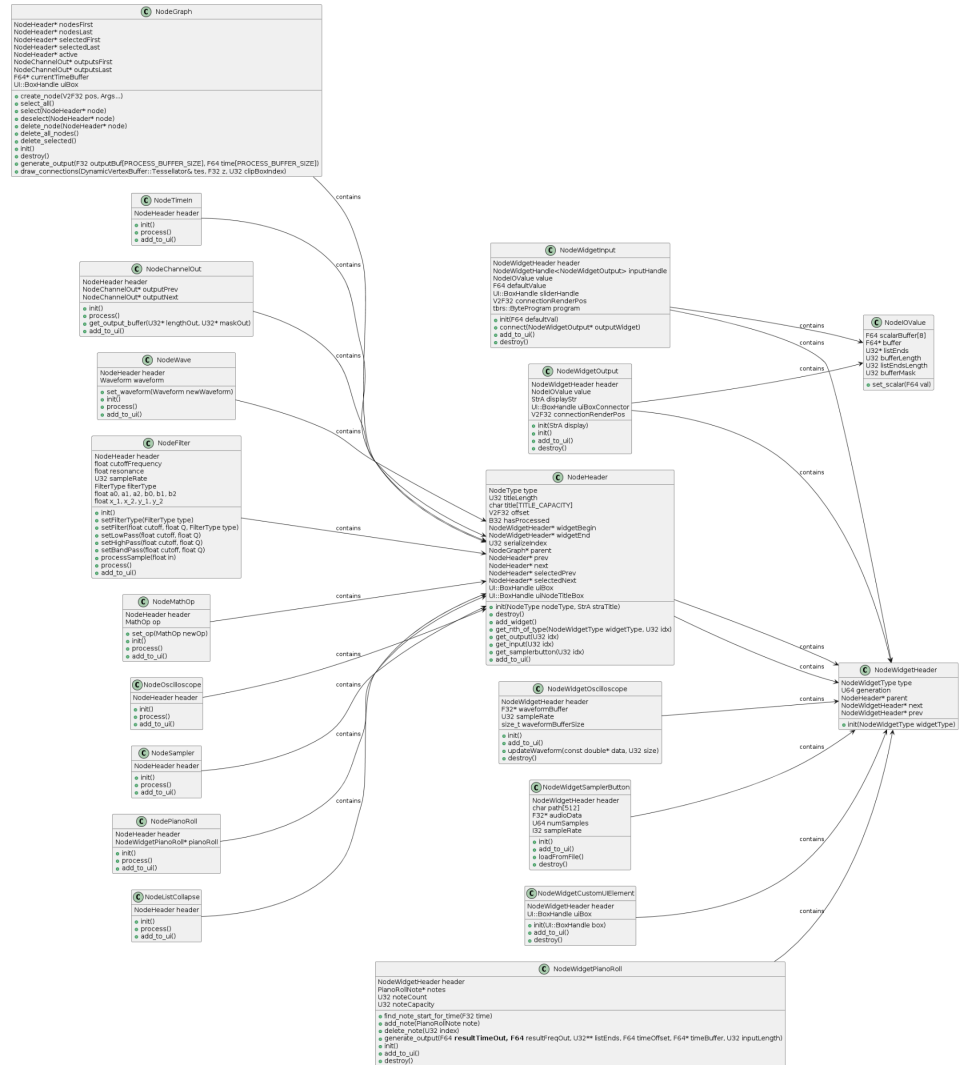
Node Count:

1 18

Graph Data:

12 select selectedlast
13 select active
14 deselect selectedfirst
15 deselect selectedlast
16 deselect active
17 deletenode nodesfirst
18 deletenode nodeslast
19 deletenode selectedfirst
20 deletenode selectedlast
21 deletenode outputsfirst
22 deletenode outputslast
23 deleteall nodesfirst
24 deleteselected selectedfirst
25 deleteselected active
26 generate currenttime
27 generate nodesfirst
28 generate outputsfirst
29 draw nodesfirst
30





III. Function Design. Summarize the *complexity* of all of your functions and interpret the result of that analysis. If you use a tool to calculate the complexity, take a snapshot of the results and include it in your report; otherwise, show your work in manually calculating the complexity of the most complex function and provide a table summarizing the complexity of the rest of the functions.

The actual screenshots of pmccabe running will be below but there is a lot so the interpretation will come first. Looking at the results of pmccabe, the application seems to mostly adhere to the low complexity design philosophy, with the overwhelming majority of functions having a complexity of less than five. However, there are a few very notable exceptions. Some of these exceptions could probably be refactored into less complex functions but several of them genuinely cannot. Something like the expression parser inherently has to switch over many possible

inputs, same with the UI input handler and the MSDF parser. In these cases we believe a high cyclomatic complexity is acceptable because it is actually much simpler to have all of the cases that need to be handled in a line than doing something like only evaluating nine and then calling into another subroutine arbitrarily because the complexity limit has been hit.

1. DrillLib.h:

```

aaulmann@Adam-Inspiron:~/cinsu30/dawdle_complexity$ gcc -c DrillLib.h
1 1 1 15 5 DrillLib.h(17): zero_memory
1 1 4 21 20 DrillLib.h(32): memcpy
1 1 3 43 8 DrillLib.h(43): memset
3 3 8 53 12 DrillLib.h(53): strcmp
3 3 8 67 9 DrillLib.h(67): strlen
3 3 8 79 12 DrillLib.h(79): memcmp
1 1 3 95 9 DrillLib.h(98): swap
1 1 1 105 3 DrillLib.h(105): bswap32
1 1 1 109 3 DrillLib.h(109): bswap16
1 1 1 113 3 DrillLib.h(113): lzcnt32
1 1 1 117 3 DrillLib.h(117): lzcnt64
1 1 1 121 4 DrillLib.h(122): bitcast
2 2 7 136 10 DrillLib.h(136): MemoryArena::init
1 1 1 147 3 DrillLib.h(147): MemoryArena::destroy
1 1 1 151 3 DrillLib.h(151): MemoryArena::reset
5 5 11 155 19 DrillLib.h(156): MemoryArena::realloc
1 1 1 175 4 DrillLib.h(176): MemoryArena::realloc
1 1 4 180 7 DrillLib.h(181): MemoryArena::alloc
1 1 5 182 8 DrillLib.h(189): MemoryArena::zalloc
1 1 4 197 7 DrillLib.h(198): MemoryArena::alloc_bytes
1 1 5 205 8 DrillLib.h(206): MemoryArena::alloc_aligned_with_slack
2 2 9 214 12 DrillLib.h(215): MemoryArena::alloc_and_commit
1 1 1 246 3 DrillLib.h(246): get_scratch_arena
2 2 2 249 3 DrillLib.h(249): get_scratch_arena_excluding
3 3 4 260 6 DrillLib.h(260): ArenaArrayList::reserve
3 3 5 267 9 DrillLib.h(267): ArenaArrayList::resize
2 2 3 277 6 DrillLib.h(277): ArenaArrayList::push_back
2 2 3 284 6 DrillLib.h(284): ArenaArrayList::push_back
1 1 1 291 3 DrillLib.h(291): ArenaArrayList::pop_back
4 4 11 295 16 DrillLib.h(295): ArenaArrayList::push_back_n
1 1 1 312 3 DrillLib.h(312): ArenaArrayList::pop_back_n
3 3 9 316 13 DrillLib.h(316): ArenaArrayList::contains
4 4 11 330 17 DrillLib.h(330): ArenaArrayList::subrange_contains
1 1 1 348 3 DrillLib.h(348): ArenaArrayList::last
1 1 1 352 3 DrillLib.h(352): ArenaArrayList::begin
1 1 1 356 3 DrillLib.h(356): ArenaArrayList::end
1 1 1 360 3 DrillLib.h(360): ArenaArrayList::back
1 1 1 364 3 DrillLib.h(364): ArenaArrayList::clear
1 1 2 368 4 DrillLib.h(368): ArenaArrayList::reset
1 1 1 373 3 DrillLib.h(373): ArenaArrayList::empty
3 3 6 383 11 DrillLib.h(383): StrA::c_str
2 2 1 394 3 DrillLib.h(394): StrA::operator==
2 2 2 397 3 DrillLib.h(397): StrA::operator[]
1 1 1 400 3 DrillLib.h(400): StrA::is_empty
4 4 8 403 11 DrillLib.h(403): StrA::find
3 3 6 414 8 DrillLib.h(414): StrA::find
4 4 8 422 11 DrillLib.h(422): StrA::rfind
3 3 6 433 8 DrillLib.h(433): StrA::rfind
2 2 1 441 3 DrillLib.h(441): StrA::starts_with
2 2 1 444 3 DrillLib.h(444): StrA::ends_with
4 4 9 447 14 DrillLib.h(447): StrA::slice
1 1 1 461 3 DrillLib.h(461): StrA::prefix
2 2 2 464 3 DrillLib.h(464): StrA::suffix
1 1 1 467 3 DrillLib.h(467): StrA::skip
1 1 2 470 4 DrillLib.h(470): StrA::substr
1 1 1 474 3 DrillLib.h(474): StrA::begin
1 1 1 477 3 DrillLib.h(477): StrA::end
1 1 1 480 3 DrillLib.h(480): StrA::front
1 1 1 483 3 DrillLib.h(483): StrA::back
1 1 1 491 3 DrillLib.h(491): operatorSTRINGLITERALsa
1 1 1 495 3 DrillLib.h(495): total_stralen
1 1 1 498 4 DrillLib.h(498): total_stralen
1 1 1 502 3 DrillLib.h(502): copy_strings_to_buffer
1 1 2 505 5 DrillLib.h(506): copy_strings_to_buffer
1 1 4 510 7 DrillLib.h(511): strncat
5 5 6 521 9 DrillLib.h(521): RWSpinLock::lock_read
4 4 7 530 10 DrillLib.h(530): RWSpinLock::lock_write
1 1 1 540 4 DrillLib.h(540): RWSpinLock::unlock_read
1 1 2 544 5 DrillLib.h(544): RWSpinLock::unlock_write
1 1 4 557 6 DrillLib.h(557): ByteBuf::wrap
1 1 1 564 3 DrillLib.h(564): ByteBuf::has_data_left
2 2 3 568 8 DrillLib.h(568): ByteBuf::skip
1 1 1 732 4 DrillLib.h(732): ByteBuf::print
1 1 1 736 3 DrillLib.h(736): ByteBuf::println
1 1 2 739 4 DrillLib.h(739): ByteBuf::println
1 1 1 743 3 DrillLib.h(743): ByteBuf::print
1 1 2 746 4 DrillLib.h(746): ByteBuf::println
3 3 9 750 15 DrillLib.h(750): ByteBuf::print_integer
3 3 10 765 14 DrillLib.h(765): ByteBuf::print_integer_pad
1 1 4 782 6 DrillLib.h(782): ByteBuf::print_float
1 1 2 788 4 DrillLib.h(788): ByteBuf::println_integer
1 1 2 792 4 DrillLib.h(792): ByteBuf::println_float
1 1 3 797 5 DrillLib.h(797): ByteBuf::abort
3 3 18 803 25 DrillLib.h(804): ByteBuf::read_full_file_to_arena
2 2 11 829 16 DrillLib.h(829): ByteBuf::write_data_to_file
3 3 20 846 30 DrillLib.h(846): ByteBuf::run_program_and_wait
2 2 4 877 7 DrillLib.h(877): ByteBuf::current_time_seconds
15 15 9 888 23 DrillLib.h(888): ByteBuf::page_fault_handler
8 8 19 912 29 DrillLib.h(912): ByteBuf::drill_lib_init

```

2. DAWdle.h:

```
aaullmann@Adam-Inspiron:/mnt/c/Users/adamu/source/repos/newest/src$ pmccabe DAWdle.h
7      7      27      27      29      DAWdle.h(27): DAWdle::fill_audio_buffer
2      2      5      57      8      DAWdle.h(57): DAWdle::audio_thread_func
1      1      2      66      4      DAWdle.h(66): DAWdle::keyboard_callback
1      1      2      70      4      DAWdle.h(70): DAWdle::mouse_callback
5      5      51      75      65      DAWdle.h(75): DAWdle::do_frame
6      6      37      141      77      DAWdle.h(141): DAWdle::run_dawdle
```

3. DrillLibDefs.h:

```
1      1      3      127      7      DrillLibDefs.h(129): load_le16
1      1      3      134      5      DrillLibDefs.h(134): load_le32
1      1      3      139      5      DrillLibDefs.h(139): load_le64
1      1      1      144      3      DrillLibDefs.h(144): store_le16
1      1      1      147      3      DrillLibDefs.h(147): store_le32
1      1      1      150      3      DrillLibDefs.h(150): store_le64
```

4. DrillMath.h:

```

aaulmann@Adam-Inspiron:/mnt/c/Users/adamu/source/repos/newest/src$ pmccabe
1 1 1 14 6 25 DrillMath.h(15): cosf32x4
1 1 1 1 32 3 DrillMath.h(32): sinf32x4
1 1 1 1 36 3 DrillMath.h(36): tanf32x4
1 1 1 8 45 10 DrillMath.h(45): acosf32x4
1 1 1 1 55 3 DrillMath.h(55): asinf32x4
1 1 1 12 61 14 DrillMath.h(61): atanf32x4
1 1 1 4 75 8 DrillMath.h(75): atan2f32x4
1 1 1 14 84 16 DrillMath.h(84): cosf32x8
1 1 1 1 101 3 DrillMath.h(101): sinf32x8
1 1 1 1 105 3 DrillMath.h(105): tanf32x8
1 1 1 1 109 20 DrillMath.h(109): cosf32
1 1 1 1 129 4 DrillMath.h(129): sinf32
1 1 1 1 133 3 DrillMath.h(133): tanf32
1 1 1 1 136 3 DrillMath.h(136): acosf32
1 1 1 1 139 3 DrillMath.h(139): asinf32
1 1 1 1 142 3 DrillMath.h(142): atanf32
1 1 1 1 145 3 DrillMath.h(145): atan2f32
1 1 1 1 148 3 DrillMath.h(148): sqrtf32
1 1 1 3 151 5 DrillMath.h(151): sincosf32
1 1 1 1 156 3 DrillMath.h(156): fractf32
1 1 1 1 159 3 DrillMath.h(159): truncf32
1 1 1 1 162 3 DrillMath.h(162): floorf32
1 1 1 1 165 3 DrillMath.h(165): ceilf32
1 1 1 1 168 3 DrillMath.h(168): absf32
1 1 1 1 171 3 DrillMath.h(171): signumf32
1 1 1 1 175 3 DrillMath.h(175): floorf64
1 1 1 1 178 3 DrillMath.h(178): truncf64
1 1 1 1 181 4 DrillMath.h(181): roundf64
1 1 1 1 185 3 DrillMath.h(185): fractf64
2 2 2 2 189 4 DrillMath.h(190): max
2 2 2 2 193 4 DrillMath.h(194): max
2 2 2 2 197 4 DrillMath.h(198): min
2 2 2 2 201 4 DrillMath.h(202): min
1 1 1 1 206 4 DrillMath.h(207): clamp
1 1 1 1 210 4 DrillMath.h(211): clamp01
2 2 2 2 215 4 DrillMath.h(216): abs
1 1 1 1 220 3 DrillMath.h(220): epsilon_eq
1 1 1 1 224 3 DrillMath.h(224): next_power_of_two
1 1 1 1 227 3 DrillMath.h(227): next_power_of_two
1 1 1 1 230 3 DrillMath.h(230): next_power_of_two
1 1 1 1 243 3 DrillMath.h(243): operator
1 1 1 1 246 3 DrillMath.h(246): operator
1 1 1 1 249 3 DrillMath.h(249): operator
1 1 1 3 252 5 DrillMath.h(252): operator
1 1 1 3 257 5 DrillMath.h(257): operator
1 1 1 1 263 3 DrillMath.h(263): operator
1 1 1 1 266 3 DrillMath.h(266): operator
1 1 1 1 269 3 DrillMath.h(269): operator
1 1 1 1 272 3 DrillMath.h(272): operator
1 1 1 3 275 5 DrillMath.h(275): operator
1 1 1 3 280 5 DrillMath.h(280): operator
1 1 1 1 286 3 DrillMath.h(286): operator
1 1 1 1 289 3 DrillMath.h(289): operator
1 1 1 1 292 3 DrillMath.h(292): operator
1 1 1 3 295 5 DrillMath.h(295): operator
1 1 1 3 300 5 DrillMath.h(300): operator
1 1 1 1 306 3 DrillMath.h(306): operator
1 1 1 2 309 4 DrillMath.h(309): operator
1 1 1 1 313 3 DrillMath.h(313): operator
1 1 1 3 316 5 DrillMath.h(316): operator
1 1 1 4 321 6 DrillMath.h(321): operator
1 1 1 1 328 3 DrillMath.h(328): dot
1 1 1 1 331 3 DrillMath.h(331): cross
1 1 1 1 334 3 DrillMath.h(334): length_sq
1 1 1 3 337 5 DrillMath.h(337): distance_sq
1 1 1 1 342 3 DrillMath.h(342): length
1 1 1 3 345 5 DrillMath.h(345): distance
1 1 1 2 350 4 DrillMath.h(350): normalize
2 2 2 1 354 3 DrillMath.h(354): epsilon_eq
1 1 1 1 357 3 DrillMath.h(357): get_orthogonal
1 1 1 5 361 7 DrillMath.h(361): println_v2f32
1 1 1 1 376 3 DrillMath.h(376): V3F32::length_sq
1 1 1 1 380 3 DrillMath.h(380): V3F32::length
1 1 1 5 384 7 DrillMath.h(384): V3F32::normalize
1 1 1 1 392 3 DrillMath.h(392): V3F32::xy
1 1 1 1 395 3 DrillMath.h(395): V3F32::xz
1 1 1 1 398 3 DrillMath.h(398): V3F32::yz

```

1	1	1	243	3	DrillMath.h(243): operator
1	1	1	246	3	DrillMath.h(246): operator
1	1	1	249	3	DrillMath.h(249): operator
1	1	3	252	5	DrillMath.h(252): operator
1	1	3	257	5	DrillMath.h(257): operator
1	1	1	263	3	DrillMath.h(263): operator
1	1	1	266	3	DrillMath.h(266): operator
1	1	1	269	3	DrillMath.h(269): operator
1	1	1	272	3	DrillMath.h(272): operator
1	1	3	275	5	DrillMath.h(275): operator
1	1	3	280	5	DrillMath.h(280): operator
1	1	1	286	3	DrillMath.h(286): operator
1	1	1	289	3	DrillMath.h(289): operator
1	1	1	292	3	DrillMath.h(292): operator
1	1	3	295	5	DrillMath.h(295): operator
1	1	3	300	5	DrillMath.h(300): operator
1	1	1	306	3	DrillMath.h(306): operator
1	1	2	309	4	DrillMath.h(309): operator
1	1	1	313	3	DrillMath.h(313): operator
1	1	3	316	5	DrillMath.h(316): operator
1	1	4	321	6	DrillMath.h(321): operator
1	1	1	328	3	DrillMath.h(328): dot
1	1	1	331	3	DrillMath.h(331): cross
1	1	1	334	3	DrillMath.h(334): length_sq
1	1	3	337	5	DrillMath.h(337): distance_sq
1	1	1	342	3	DrillMath.h(342): length
1	1	3	345	5	DrillMath.h(345): distance
1	1	2	350	4	DrillMath.h(350): normalize
2	2	1	354	3	DrillMath.h(354): epsilon_eq
1	1	1	357	3	DrillMath.h(357): get_orthogonal
1	1	5	361	7	DrillMath.h(361): println_v2f32
1	1	1	376	3	DrillMath.h(376): V3F32::length_sq
1	1	1	380	3	DrillMath.h(380): V3F32::length
1	1	5	384	7	DrillMath.h(384): V3F32::normalize
1	1	1	392	3	DrillMath.h(392): V3F32::xy
1	1	1	395	3	DrillMath.h(395): V3F32::xz
1	1	1	398	3	DrillMath.h(398): V3F32::yz
1	1	1	411	3	DrillMath.h(411): operator
1	1	1	414	3	DrillMath.h(414): operator
1	1	1	417	3	DrillMath.h(417): operator
1	1	1	421	3	DrillMath.h(421): operator
1	1	1	424	3	DrillMath.h(424): operator
1	1	1	427	3	DrillMath.h(427): operator
1	1	1	430	3	DrillMath.h(430): operator
1	1	1	434	3	DrillMath.h(434): operator
1	1	1	437	3	DrillMath.h(437): operator
1	1	1	440	3	DrillMath.h(440): operator
1	1	1	444	3	DrillMath.h(444): operator
1	1	2	447	4	DrillMath.h(447): operator
1	1	1	451	3	DrillMath.h(451): operator
1	1	1	456	3	DrillMath.h(456): dot
1	1	1	459	3	DrillMath.h(459): cross
1	1	1	462	3	DrillMath.h(462): length_sq
1	1	1	465	3	DrillMath.h(465): distance_sq
1	1	1	468	3	DrillMath.h(468): length
1	1	1	471	3	DrillMath.h(471): distance
1	1	2	474	4	DrillMath.h(474): normalize
1	1	7	479	11	DrillMath.h(481): println_v3f32
1	1	9	503	11	DrillMath.h(503): println_v4f32
1	1	3	516	6	DrillMath.h(517): time_along_line
1	1	2	522	5	DrillMath.h(523): distance_to_line
1	1	2	527	5	DrillMath.h(528): distance_to_ray
1	1	2	532	5	DrillMath.h(533): distance_to_segment
1	1	3	537	5	DrillMath.h(537): signed_distance_to_line
1	1	2	542	4	DrillMath.h(542): signed_distance_to_ray
1	1	3	546	5	DrillMath.h(546): signed_distance_to_segment
1	1	1	552	6	DrillMath.h(555): mix
3	3	5	558	10	DrillMath.h(559): clamped_mix
1	1	2	569	4	DrillMath.h(569): vec2_angle
1	1	7	586	9	DrillMath.h(586): QF32::from_axis_angle
1	1	7	598	13	DrillMath.h(598): QF32::transform
1	1	1	612	3	DrillMath.h(612): QF32::conjugate
1	1	1	616	3	DrillMath.h(616): QF32::magnitude_sq
1	1	1	620	3	DrillMath.h(620): QF32::magnitude
1	1	2	624	4	DrillMath.h(624): QF32::normalize
1	1	2	629	4	DrillMath.h(629): QF32::inverse
1	1	1	635	12	DrillMath.h(635): operator
1	1	3	648	6	DrillMath.h(648): AxisAngleF32::to_quaternion
1	1	5	658	7	DrillMath.h(658): M2F32::set_identity
1	1	5	665	7	DrillMath.h(665): M2F32::set_zero
1	1	5	672	7	DrillMath.h(672): M2F32::set
1	1	11	679	13	DrillMath.h(679): M2F32::rotate
1	1	4	692	6	DrillMath.h(692): M2F32::transpose
1	1	3	698	5	DrillMath.h(698): M2F32::transform
1	1	1	703	3	DrillMath.h(703): M2F32::operator*

```

1 1 1 716 3 DrillMath.h(716): M4x3F32::copy
1 1 13 720 15 DrillMath.h(720): M4x3F32::set_zero
1 1 13 736 15 DrillMath.h(736): M4x3F32::set_identity
1 1 10 752 12 DrillMath.h(752): M4x3F32::transpose_rotation
1 1 19 767 21 DrillMath.h(767): M4x3F32::set_orientation_from_quat
1 1 4 789 6 DrillMath.h(789): M4x3F32::set_offset
1 1 4 796 6 DrillMath.h(796): M4x3F32::translate
1 1 4 803 6 DrillMath.h(803): M4x3F32::add_offset
1 1 13 810 15 DrillMath.h(810): M4x3F32::rotate_quat
1 1 1 826 3 DrillMath.h(826): M4x3F32::rotate_axis_angle
1 1 10 830 12 DrillMath.h(830): M4x3F32::scale
1 1 13 843 15 DrillMath.h(843): M4x3F32::scale_global
1 1 1 859 3 DrillMath.h(859): M4x3F32::determinant_upper_left_3x3_corner
1 1 26 863 36 DrillMath.h(863): M4x3F32::invert
1 1 25 901 33 DrillMath.h(901): M4x3F32::invert_orthogonal
1 1 8 935 11 DrillMath.h(935): M4x3F32::invert_orthonormal
1 1 1 947 7 DrillMath.h(947): M4x3F32::transform
3 3 5 955 9 DrillMath.h(955): M4x3F32::get_row
3 3 12 965 16 DrillMath.h(965): M4x3F32::set_row
1 1 14 984 16 DrillMath.h(984): operator
1 1 1 1001 3 DrillMath.h(1001): operator
1 1 27 1005 31 DrillMath.h(1007): println_m4x3f32
1 1 17 1086 19 DrillMath.h(1086): PerspectiveProjection::project_perspective
1 1 5 1106 7 DrillMath.h(1106): PerspectiveProjection::transform
1 1 12 1122 15 DrillMath.h(1122): ProjectiveTransformMatrix::generate
4 4 1 1145 3 DrillMath.h(1145): RGBA8::operator==
1 1 2 1148 4 DrillMath.h(1148): RGBA8::to_v4f32
1 1 1 1152 3 DrillMath.h(1152): RGBA8::to_rgba8
1 1 1 1156 3 DrillMath.h(1156): V4F32::to_rgba8
3 3 1 1162 3 DrillMath.h(1162): RGB8::operator==
1 1 1 1165 3 DrillMath.h(1165): RGB8::operator+
1 1 1 1168 3 DrillMath.h(1168): RGB8::operator-
1 1 1 1171 3 DrillMath.h(1171): RGB8::to_rgba8
2 2 1 1178 3 DrillMath.h(1178): RG8::operator==
1 1 1 1181 3 DrillMath.h(1181): RG8::to_rgba8
1 1 1 1188 3 DrillMath.h(1188): R8::operator==
1 1 1 1191 3 DrillMath.h(1191): R8::to_rgba8
3 3 5 1230 9 DrillMath.h(1230): axis2_orthogonal
1 1 2 1252 4 DrillMath.h(1252): Rng1F32::init
1 1 1 1256 3 DrillMath.h(1256): Rng1F32::midpoint
1 1 1 1261 3 DrillMath.h(1261): rng_union
2 2 4 1264 4 DrillMath.h(1264): rng_intersect
1 1 1 1268 3 DrillMath.h(1268): rng_area
2 2 1 1271 3 DrillMath.h(1271): rng_contains_point
1 1 1 1279 3 DrillMath.h(1279): Rng2F32::midpoint
1 1 1 1284 3 DrillMath.h(1284): rng_union
3 3 3 1287 4 DrillMath.h(1287): rng_intersect
1 1 1 1291 3 DrillMath.h(1291): rng_area
4 4 1 1294 3 DrillMath.h(1294): rng_contains_point
1 1 1 1302 3 DrillMath.h(1302): Rng3F32::midpoint
1 1 1 1307 3 DrillMath.h(1307): rng_union
4 4 3 1310 4 DrillMath.h(1310): rng_intersect
1 1 1 1314 3 DrillMath.h(1314): rng_area
6 6 1 1317 3 DrillMath.h(1317): rng_contains_point
1 1 1 1322 3 DrillMath.h(1322): pack_unorm4x8
1 1 4 1326 8 DrillMath.h(1327): eval_bezier_quadratic
1 1 6 1334 10 DrillMath.h(1335): eval_bezier_cubic

```

5. DynamicVertexBuffer.h:

```

aaulmann@Adam-Inspiron: /mnt/c/Users/adamu/source/repos/newest/src$ pmccabe DynamicVertexBuffer.h
4 4 14 35 21 DynamicVertexBuffer.h(35): DynamicVertexBuffer::Tessellator::ensure_space_for
1 1 8 57 11 DynamicVertexBuffer.h(57): DynamicVertexBuffer::Tessellator::init
1 1 1 68 3 DynamicVertexBuffer.h(68): DynamicVertexBuffer::Tessellator::destroy
2 2 14 72 17 DynamicVertexBuffer.h(72): DynamicVertexBuffer::Tessellator::begin_draw
1 1 1 89 3 DynamicVertexBuffer.h(89): DynamicVertexBuffer::Tessellator::set_clip_boxes
1 1 1 92 3 DynamicVertexBuffer.h(92): DynamicVertexBuffer::Tessellator::end_draw
3 3 16 95 21 DynamicVertexBuffer.h(95): DynamicVertexBuffer::Tessellator::draw
1 1 5 117 7 DynamicVertexBuffer.h(117): DynamicVertexBuffer::Tessellator::pos3
1 1 4 124 6 DynamicVertexBuffer.h(124): DynamicVertexBuffer::Tessellator::pos2
1 1 4 130 6 DynamicVertexBuffer.h(130): DynamicVertexBuffer::Tessellator::tex
1 1 4 136 6 DynamicVertexBuffer.h(136): DynamicVertexBuffer::Tessellator::color
1 1 1 142 3 DynamicVertexBuffer.h(142): DynamicVertexBuffer::Tessellator::color
1 1 3 145 5 DynamicVertexBuffer.h(145): DynamicVertexBuffer::Tessellator::u32
1 1 1 150 3 DynamicVertexBuffer.h(150): DynamicVertexBuffer::Tessellator::tex_idx
1 1 1 153 3 DynamicVertexBuffer.h(153): DynamicVertexBuffer::Tessellator::flags
1 1 3 156 6 DynamicVertexBuffer.h(157): DynamicVertexBuffer::Tessellator::element
3 3 17 162 22 DynamicVertexBuffer.h(162): DynamicVertexBuffer::Tessellator::end_vert
2 2 13 184 15 DynamicVertexBuffer.h(185): DynamicVertexBuffer::Tessellator::add_geometry
1 1 17 199 28 DynamicVertexBuffer.h(199): DynamicVertexBuffer::Tessellator::ui_rect2d
6 6 69 227 79 DynamicVertexBuffer.h(227): DynamicVertexBuffer::Tessellator::ui_line_strip
2 2 10 306 13 DynamicVertexBuffer.h(306): DynamicVertexBuffer::Tessellator::ui_bezier_curve
2 2 4 321 5 DynamicVertexBuffer.h(321): DynamicVertexBuffer::init
2 2 4 326 5 DynamicVertexBuffer.h(326): DynamicVertexBuffer::destroy
1 1 1 332 3 DynamicVertexBuffer.h(332): DynamicVertexBuffer::get_tessellator

```

6. Entrypoint.cpp:

```

aaulmann@Adam-Inspiron: /mnt/c/Users/adamu/source/repos/newest/src$ pmccabe Entrypoint.cpp
2 2 4 6 7 Entrypoint.cpp(6): main

```

7. ExpressionParser.h:

```
aaullmann@Adam-Inspiron:/mnt/c/Users/adamu/source/repos/newest/src$ pmccabe ExpressionParser.h
5      5      6      19      9      ExpressionParser.h(19): tbrs::parse_op
4      4      5      29      8      ExpressionParser.h(29): tbrs::parse_dollardollar
2      2      5      38      8      ExpressionParser.h(38): tbrs::parse_lparen
2      2      5      47      8      ExpressionParser.h(47): tbrs::parse_rparen
7      7      19      62      28     ExpressionParser.h(62): tbrs::Lex
1      1      1      99      1      ExpressionParser.h(99): tbrs::Node::Node
"ExpressionParser.h", line 100: too many }'s
2      5      4      104     12     ExpressionParser.h(104): tbrs::Node::precedence
10     13     33     119     42     ExpressionParser.h(119): tbrs::Node::parse
3      3      9      162     11     ExpressionParser.h(162): tbrs::Node::parse_helper
1      1      5      190     7      ExpressionParser.h(190): tbrs::Node::ByteProgram::init
1      1      2      198     4      ExpressionParser.h(198): tbrs::Node::ByteProgram::destroy
3      6      17     212     32     ExpressionParser.h(212): tbrs::Node::traverse
3      3      16     245     25     ExpressionParser.h(245): tbrs::Node::compile
3      3      11     271     17     ExpressionParser.h(271): tbrs::Node::parse_program
3      8      36     291     44     ExpressionParser.h(291): tbrs::Node::interpret
```

8. MSDFGenerator.h:

```
aaullmann@Adam-Inspiron:/mnt/c/Users/adamu/source/repos/newest/src$ pmccabe MSDFGenerator.h
2      2      3      17      7      MSDFGenerator.h(17): MSDFGenerator::SignedDistance::operator>
2      2      3      25      9      MSDFGenerator.h(25): MSDFGenerator::SignedDistance::operator<
1      1      3      35      5      MSDFGenerator.h(35): MSDFGenerator::SignedDistance::abs
2      2      2      59      3      MSDFGenerator.h(59): MSDFGenerator::ShapeSegmentHeader::color_at
3      3      3      67      6      MSDFGenerator.h(67): MSDFGenerator::ShapeSegmentLinear::eval
1      1      1      73      3      MSDFGenerator.h(73): MSDFGenerator::ShapeSegmentLinear::direction
3      3      8      76      10     MSDFGenerator.h(76): MSDFGenerator::ShapeSegmentLinear::distance
1      1      4      86      6      MSDFGenerator.h(86): MSDFGenerator::ShapeSegmentLinear::pseudo_distance
3      3      7      97      11     MSDFGenerator.h(97): MSDFGenerator::ShapeSegmentQuadratic::eval
1      1      3      108     5      MSDFGenerator.h(108): MSDFGenerator::ShapeSegmentQuadratic::direction
1      1      1      113     3      MSDFGenerator.h(113): MSDFGenerator::ShapeSegmentQuadratic::distance
1      1      1      116     3      MSDFGenerator.h(116): MSDFGenerator::ShapeSegmentQuadratic::pseudo_distance
3      3      10     124     14     MSDFGenerator.h(124): MSDFGenerator::ShapeSegmentCubic::eval
5      5      10     138     15     MSDFGenerator.h(138): MSDFGenerator::ShapeSegmentCubic::direction
1      1      1      153     3      MSDFGenerator.h(153): MSDFGenerator::ShapeSegmentCubic::distance
1      1      1      156     3      MSDFGenerator.h(156): MSDFGenerator::ShapeSegmentCubic::pseudo_distance
2      4      5      166     8      MSDFGenerator.h(166): MSDFGenerator::shape_segment_direction
2      4      5      174     8      MSDFGenerator.h(174): MSDFGenerator::shape_segment_eval
2      4      5      182     8      MSDFGenerator.h(182): MSDFGenerator::shape_segment_distance
2      4      5      190     8      MSDFGenerator.h(190): MSDFGenerator::shape_segment_pseudo_distance
3      3      12     211     19     MSDFGenerator.h(212): MSDFGenerator::bezier_pseudo_distance
7      7      33     231     45     MSDFGenerator.h(232): MSDFGenerator::numerical_distance_solve
7      7      26     279     30     MSDFGenerator.h(279): MSDFGenerator::endpoint_arc_to_center_arc
6      6      34     310     44     MSDFGenerator.h(310): MSDFGenerator::create_arc
40     47     156     356     189     MSDFGenerator.h(356): MSDFGenerator::parse_shape
8      8      19     546     23     MSDFGenerator.h(546): MSDFGenerator::preprocess_shape
1      1      1      570     3      MSDFGenerator.h(570): MSDFGenerator::median
15     15     41     574     52     MSDFGenerator.h(574): MSDFGenerator::render_shape_to_msdf
7      7      12     628     21     MSDFGenerator.h(628): MSDFGenerator::detect_error_pixel_pair
23     23     16     650     25     MSDFGenerator.h(650): MSDFGenerator::error_correct
4      4      28     682     37     MSDFGenerator.h(682): MSDFGenerator::generate_msdf_image
12     12     50     720     61     MSDFGenerator.h(720): MSDFGenerator::generate_msdf_font
```

9. NodeUI.h:

```
aaullmann@Adam-Inspiron:/mnt/c/Users/adamu/source/repos/newest/src$ pmccabe NodeUI.h
28     28     142     31      203     NodeUI.h(31): NodeUI::Panel::build_ui
5      5      52     235     63      NodeUI.h(235): NodeUI::Panel::split
5      5      18     299     23      NodeUI.h(299): NodeUI::Panel::destroy
3      3      3      323     5      NodeUI.h(323): NodeUI::Panel::sibling_ref
3      3      3      328     5      NodeUI.h(328): NodeUI::Panel::child_ref
2      2      7      337     10     NodeUI.h(337): NodeUI::alloc_panel
1      1      1      348     4      NodeUI.h(348): NodeUI::free_panel
1      1      9      353     11     NodeUI.h(353): NodeUI::init
```

10. Nodes.h:


```

saullmann@Adam-Inspiron:/mnt/c/Users/adamu/source/repos/newest/src$ gcc -c Nodes.h
2 2 9 67 10 Nodes.h(67): Nodes::NodeIOValue::set_scalar
29 29 103 78 113 Nodes.h(78): Nodes::make_node_io_consistent
1 1 2 201 4 Nodes.h(201): Nodes::NodeWidgetHeader::init
4 4 2 211 3 Nodes.h(211): Nodes::NodeWidgetHandle::get
1 1 3 224 5 Nodes.h(224): Nodes::NodeWidgetOutput::init
1 1 1 229 3 Nodes.h(229): Nodes::NodeWidgetOutput::init
1 1 0 235 3 Nodes.h(235): Nodes::NodeWidgetOutput::destroy
1 1 4 251 6 Nodes.h(251): Nodes::NodeWidgetInput::init
1 1 1 258 3 Nodes.h(258): Nodes::NodeWidgetInput::connect
9 9 34 262 49 Nodes.h(262): Nodes::NodeWidgetInput::add_to_ui
1 1 1 312 3 Nodes.h(312): Nodes::NodeWidgetInput::destroy
1 1 5 322 7 Nodes.h(322): Nodes::NodeWidgetOscilloscope::init
3 3 25 330 33 Nodes.h(330): Nodes::NodeWidgetOscilloscope::add_to_ui
2 2 5 364 6 Nodes.h(364): Nodes::NodeWidgetOscilloscope::updateWaveform
1 1 1 371 3 Nodes.h(371): Nodes::NodeWidgetOscilloscope::destroy
1 1 1 382 3 Nodes.h(382): Nodes::NodeWidgetSamplerButton::init
1 1 18 386 26 Nodes.h(386): Nodes::NodeWidgetSamplerButton::add_to_ui
3 3 13 413 19 Nodes.h(413): Nodes::NodeWidgetSamplerButton::loadFromFile
1 1 1 433 3 Nodes.h(433): Nodes::NodeWidgetSamplerButton::destroy
1 1 2 440 4 Nodes.h(440): Nodes::NodeWidgetCustomUIElement::init
2 2 3 444 6 Nodes.h(444): Nodes::NodeWidgetCustomUIElement::add_to_ui
1 1 0 450 1 Nodes.h(450): Nodes::NodeWidgetCustomUIElement::destroy
5 5 6 610 10 Nodes.h(610): Nodes::NodeWidgetPianoRoll::find_note_start_for_time
5 5 12 620 17 Nodes.h(620): Nodes::NodeWidgetPianoRoll::add_note
1 1 2 637 4 Nodes.h(637): Nodes::NodeWidgetPianoRoll::delete_note
6 6 27 641 32 Nodes.h(641): Nodes::NodeWidgetPianoRoll::generate_output
1 1 4 674 6 Nodes.h(674): Nodes::NodeWidgetPianoRoll::init
16 16 95 680 112 Nodes.h(680): Nodes::NodeWidgetPianoRoll::add_to_ui
1 1 1 792 3 Nodes.h(792): Nodes::NodeWidgetPianoRoll::destroy
3 3 12 813 14 Nodes.h(813): Nodes::alloc_widget
1 1 3 827 5 Nodes.h(827): Nodes::free_widget
1 1 11 852 13 Nodes.h(852): Nodes::NodeHeader::init
3 3 2 866 15 Nodes.h(866): Nodes::NodeHeader::destroy
2 2 10 882 15 Nodes.h(882): Nodes::NodeHeader::add_widget
4 4 9 898 12 Nodes.h(898): Nodes::NodeHeader::get_nth_of_type
1 1 1 910 3 Nodes.h(910): Nodes::NodeHeader::get_output
1 1 1 913 3 Nodes.h(913): Nodes::NodeHeader::get_input
1 1 1 916 3 Nodes.h(916): Nodes::NodeHeader::get_samplerbutton
1 1 2 924 4 Nodes.h(924): Nodes::NodeTimeIn::init
1 1 2 929 4 Nodes.h(929): Nodes::NodeTimeIn::add_to_ui
1 1 2 940 4 Nodes.h(940): Nodes::NodeChannelOut::init
1 1 0 944 3 Nodes.h(944): Nodes::NodeChannelOut::process
2 2 6 947 10 Nodes.h(947): Nodes::NodeChannelOut::get_output_buffer
1 1 2 957 4 Nodes.h(957): Nodes::NodeChannelOut::add_to_ui
1 1 4 966 6 Nodes.h(966): Nodes::random_f32
2 2 6 981 10 Nodes.h(981): Nodes::waveform_name
2 2 3 998 6 Nodes.h(998): Nodes::NodeWave::set_waveform
3 3 25 1005 35 Nodes.h(1005): Nodes::NodeWave::init
7 11 60 1041 64 Nodes.h(1041): Nodes::NodeWave::process
1 1 2 1105 4 Nodes.h(1105): Nodes::NodeWave::add_to_ui
2 4 5 1117 8 Nodes.h(1117): Nodes::filterTypeName
3 3 33 1137 44 Nodes.h(1137): Nodes::NodeFilter::init
2 2 3 1182 6 Nodes.h(1182): Nodes::NodeFilter::setFilterType
2 4 7 1189 13 Nodes.h(1189): Nodes::NodeFilter::setFilter
1 1 14 1203 18 Nodes.h(1203): Nodes::NodeFilter::setLowPass
1 1 14 1222 18 Nodes.h(1222): Nodes::NodeFilter::setHighPass
1 1 14 1241 18 Nodes.h(1241): Nodes::NodeFilter::setBandPass
1 1 6 1260 11 Nodes.h(1260): Nodes::NodeFilter::processSample
4 4 14 1271 18 Nodes.h(1271): Nodes::NodeFilter::process
1 1 2 1290 4 Nodes.h(1290): Nodes::NodeFilter::add_to_ui
2 21 22 1322 25 Nodes.h(1322): Nodes::math_op_name
2 2 3 1351 7 Nodes.h(1351): Nodes::NodeMathOp::set_op
6 6 41 1359 62 Nodes.h(1359): Nodes::NodeMathOp::init
22 41 141 1421 144 Nodes.h(1421): Nodes::NodeMathOp::process
1 1 2 1565 4 Nodes.h(1565): Nodes::NodeMathOp::add_to_ui
1 1 3 1574 5 Nodes.h(1574): Nodes::NodeOscilloscope::init
2 2 7 1579 12 Nodes.h(1579): Nodes::NodeOscilloscope::process
1 1 2 1591 4 Nodes.h(1591): Nodes::NodeOscilloscope::add_to_ui
1 1 4 1600 6 Nodes.h(1600): Nodes::NodeSampler::init
3 3 45 1606 51 Nodes.h(1606): Nodes::NodeSampler::process
1 1 2 1657 4 Nodes.h(1657): Nodes::NodeSampler::add_to_ui
1 1 6 1666 8 Nodes.h(1666): Nodes::NodePianoRoll::init
1 1 17 1674 21 Nodes.h(1674): Nodes::NodePianoRoll::process
1 1 1 1695 3 Nodes.h(1695): Nodes::NodePianoRoll::add_to_ui
1 1 3 1782 5 Nodes.h(1782): Nodes::NodeListCollapse::init
4 4 20 1787 22 Nodes.h(1787): Nodes::NodeListCollapse::process
1 1 1 1729 3 Nodes.h(1729): Nodes::NodeListCollapse::add_to_ui
"Nodes.h", line 1739: too many ]'s
3 3 12 1743 14 Nodes.h(1743): Nodes::Node::alloc_node
1 1 2 1757 4 Nodes.h(1757): Nodes::Node::free_node
13 12 33 1762 47 Nodes.h(1762): Nodes::Node::process_node
2 1 5 1810 11 Nodes.h(1810): Nodes::Node::add_node_to_ui
2 1 2 1822 8 Nodes.h(1822): Nodes::Node::add_widget_to_ui
2 2 9 1846 13 Nodes.h(1846): Nodes::Node::NodeGraph::create_node
2 2 8 1860 9 Nodes.h(1860): Nodes::Node::NodeGraph::select_all
2 2 3 1869 6 Nodes.h(1869): Nodes::Node::NodeGraph::select
3 3 4 1875 8 Nodes.h(1875): Nodes::Node::NodeGraph::deselect
3 3 6 1883 10 Nodes.h(1883): Nodes::Node::NodeGraph::delete_node
2 2 4 1893 7 Nodes.h(1893): Nodes::Node::NodeGraph::delete_all_nodes
2 2 7 1900 8 Nodes.h(1900): Nodes::Node::NodeGraph::delete_selected
1 1 3 1908 5 Nodes.h(1908): Nodes::Node::NodeGraph::init
2 2 4 1913 5 Nodes.h(1913): Nodes::Node::NodeGraph::destroy
7 7 29 1919 28 Nodes.h(1919): Nodes::Node::NodeGraph::generate_output
5 5 11 1948 13 Nodes.h(1948): Nodes::Node::NodeGraph::draw_connections
5 5 31 1963 42 Nodes.h(1963): Nodes::Node::NodeHeader::add_to_ui
2 2 7 2006 11 Nodes.h(2006): Nodes::Node::NodeTimeIn::process
6 6 26 2018 37 Nodes.h(2018): Nodes::Node::NodeWidgetOutput::add_to_ui

```

11. PNG.h:


```

aaulmann@Adam-Inspiron:/mnt/c/Users/adamu/source/repos/newest/src$ pmccabe PNG.h
2      2      11      11      12      PNG.h(11): PNG::BitReader::init
2      2      3      24      11      PNG.h(24): PNG::BitReader::try_read_next
1      1      9      37      13      PNG.h(37): PNG::BitReader::increase_byte_pos
1      1      4      51      7      PNG.h(51): PNG::BitReader::align_to_byte
1      1      2      59      4      PNG.h(59): PNG::BitReader::get_current_pointer
1      1      5      64      7      PNG.h(64): PNG::BitReader::read_bits
1      1      4      72      6      PNG.h(72): PNG::BitReader::read_bits_no_check
1      1      2      79      4      PNG.h(79): PNG::BitReader::put_back_bits
1      1      5      84      7      PNG.h(84): PNG::BitReader::read_uint8
1      1      5      92      7      PNG.h(92): PNG::BitReader::read_uint16
1      1      1      100      3      PNG.h(100): PNG::BitReader::read_uint32
3      3      21      112      49      PNG.h(112): PNG::HuffmanTree::read_next
1      1      6      162      9      PNG.h(162): PNG::HuffmanTree::reverse_bits
5      5      21      172      33      PNG.h(172): PNG::HuffmanTree::build
4      4      18      232      22      PNG.h(232): PNG::adler32
4      4      11      258      13      PNG.h(258): PNG::compute_crc_table
2      2      6      272      7      PNG.h(272): PNG::crc32
3      3      6      280      10      PNG.h(280): PNG::resize_buffer
1      1      9      291      12      PNG.h(291): PNG::generate_fixed_tree
9      9      29      304      30      PNG.h(304): PNG::read_tree_lengths
2      2      23      335      32      PNG.h(335): PNG::decompress_trees
11     11     61      368      90      PNG.h(368): PNG::inflate
4      8      34      482      35      PNG.h(482): PNG::read_ihdr
4      4      9      518      13      PNG.h(518): PNG::paeth_predictor
3      3      3      538      7      PNG.h(538): PNG::sample_line
4      8      33      546      45      PNG.h(546): PNG::translate_pass
12     19     64      593      94      PNG.h(593): PNG::rescale_bit_depth
11     11     26      689      29      PNG.h(690): PNG::rescale_components
16     16     34      719      49      PNG.h(719): PNG::translate_png_data
16     16     97      773      149     PNG.h(773): PNG::read_image
4      4      52      923      72      PNG.h(923): PNG::write_image
1      1      2      996      4      PNG.h(996): PNG::init_loader

```

12. Serialization.h:

```

aaulmann@Adam-Inspiron:/mnt/c/Users/adamu/source/repos/newest/src$ pmccabe Serialization.h
2      1      3      8      10      Serialization.h(8): Nodes::createNodeByType
24     24     109     21      125     Serialization.h(21): Serialization::SaveNodeGraph
19     19     86      146     106     Serialization.h(146): Serialization::LoadNodeGraph

```

13. SerializeTools.h:

```

aaulmann@Adam-Inspiron:/mnt/c/Users/adamu/source/repos/newest/src$ pmccabe SerializeTools.h
6      6      1      7      3      SerializeTools.h(7): SerializeTools::is_whitespace
2      2      1      10     3      SerializeTools.h(10): SerializeTools::is_digit
6      6      1      13     3      SerializeTools.h(13): SerializeTools::is_hex_digit
4      4      1      16     3      SerializeTools.h(16): SerializeTools::is_alpha
2      2      1      19     3      SerializeTools.h(19): SerializeTools::is_upper_alpha
2      2      1      22     3      SerializeTools.h(22): SerializeTools::is_lower_alpha
3      3      3      26     6      SerializeTools.h(26): SerializeTools::skip_whitespace
39     39     90      42      179     SerializeTools.h(42): SerializeTools::parse_f64
1      1      4      221     6      SerializeTools.h(221): SerializeTools::parse_f32
44     44     174     228     300     SerializeTools.h(228): SerializeTools::serialize_f64
1      1      1      528     3      SerializeTools.h(528): SerializeTools::serialize_f32

```

14. TextRenderer.h:

```

aaulmann@Adam-Inspiron:/mnt/c/Users/adamu/source/repos/newest/src$ pmccabe TextRenderer.h
1      1      4      8      6      TextRenderer.h(8): TextRenderer::string_size_x
3      3      40     15      49      TextRenderer.h(15): TextRenderer::draw_string_batched
1      1      4      65     6      TextRenderer.h(65): TextRenderer::draw_string
1      1      1      72     3      TextRenderer.h(72): TextRenderer::draw_string

```

15. Textures.h:

```

aaulmann@Adam-Inspiron:/mnt/c/Users/adamu/source/repos/newest/src$ pmccabe Textures.h
4      4      14      49      19      Textures.h(49): Textures::alloc_texture_memory
4      4      69      69      83      Textures.h(69): Textures::create_texture
2      2      7      153     13      Textures.h(153): Textures::load_png
2      2      8      167     15      Textures.h(167): Textures::load_msdf
4      4      23      183     24      Textures.h(183): Textures::load_all
3      3      7      208     11      Textures.h(208): Textures::destroy_all

```

16. UI.h:

```

aaulmann@Adam-Inspiron: /mnt/c/Users/adamu/source/repos/newest/src$ pmtccabe UI.h
4 4 2 151 3 UI.h(151): UI::BoxHandle::get
2 2 22 176 26 UI.h(176): UI::make_default_settings
2 2 6 203 9 UI.h(203): UI::alloc_text_input
1 1 2 212 4 UI.h(212): UI::free_text_input
3 3 15 216 17 UI.h(216): UI::alloc_box
6 6 17 233 22 UI.h(233): UI::free_box
2 2 3 256 6 UI.h(256): UI::move_box_to_front
8 8 22 263 27 UI.h(263): UI::context_menu
1 1 1 291 3 UI.h(291): UI::clear_context_menu
2 2 24 295 28 UI.h(295): UI::init_ui
2 2 4 324 5 UI.h(324): UI::destroy_ui
19 19 52 330 64 UI.h(330): UI::layout_boxes_recurse
25 25 56 395 71 UI.h(395): UI::solve_layout_conflicts_and_position_boxes_recurse
5 5 31 467 38 UI.h(467): UI::layout_boxes
3 3 13 506 20 UI.h(506): UI::compute_final_render_area_and_offset_for_children
19 19 46 526 58 UI.h(526): UI::draw_box
4 4 20 585 25 UI.h(585): UI::draw
20 20 46 611 60 UI.h(611): UI::mouse_input_for_box_recurse
9 9 18 672 22 UI.h(672): UI::handle_mouse_action
10 10 25 694 31 UI.h(694): UI::mouse_update_for_box_recurse
11 11 25 726 32 UI.h(726): UI::handle_mouse_update
10 10 20 758 26 UI.h(758): UI::keyboard_input_for_box_recurse
7 7 21 784 28 UI.h(784): UI::handle_keyboard_action
1 1 4 815 6 UI.h(815): UI::generic_box
2 2 2 821 5 UI.h(821): UI::pop_box
1 1 5 826 7 UI.h(826): UI::layout_box
1 1 1 833 3 UI.h(833): UI::ubox
1 1 1 836 3 UI.h(836): UI::dbox
1 1 1 839 3 UI.h(839): UI::lbox
1 1 1 842 3 UI.h(842): UI::rbox
3 3 8 849 9 UI.h(849): UI::spacer
1 1 1 858 3 UI.h(858): UI::spacer
1 1 5 861 7 UI.h(861): UI::str_a
8 8 22 868 28 UI.h(868): UI::text_input
3 3 12 896 17 UI.h(896): UI::button
3 3 12 913 17 UI.h(913): UI::text_button
1 1 3 930 4 UI.h(930): UI::expandable_down
1 1 0 935 7 UI.h(935): UI::dropdown
1 1 4 949 6 UI.h(949): UI::dropdown_button
1 1 2 956 4 UI.h(956): UI::toggle
19 19 63 960 89 UI.h(960): UI::slider_number
3 3 27 1050 39 UI.h(1050): UI::scroll_bar
1 1 9 1090 11 UI.h(1090): UI::context_menu_begin_helper
1 1 4 1101 6 UI.h(1101): UI::context_menu_end_helper

```

IV. Smells and Anti-patterns. Identify any code smells or anti-patterns in your software, and give recommendations on how they should be addressed. If you use a static analysis tool(s), provide a snapshot of its results and specify which tool(s) you used.

Most of the classes we wrote for the application do not follow the RAII pattern common to C++. Instead, each class has an initialization function that must be explicitly called on object creation. This is a form of sequence coupling as any further member functions that rely on the initialized values will not work as intended until the initialization function is called. The obvious solution would be to refactor every initialization function into a constructor, but this would also necessitate rewriting a ton of other logic as memory is copied, swapped, allocated, and constantly deallocated in a generic way that does not respect RAII or copy or move semantics, making it impossible for us to ensure that such automatically invoked functions will actually be called.

DAWdle also suffers from vendor lock-in as all of the logic that necessitates system calls into the OS directly calls into the Win32 library. Since this code is written specifically for the way the Windows OS API is structured it may prove difficult to port to other OSs that do not operate in the same way. The fix for this would be to either port all OS-facing code to using an agnostic library that handles the OS-specific dispatching for

us, or to manually write code paths for every operating system we wish to support.

Yet another anti-pattern that can be found in DAWdle is busy waiting. DAWdle relies heavily on multiple threads to achieve the highest performance possible, and where there is complex multithreading code there is the need for synchronization. As with many other aspects of the codebase, DAWdle does not rely on any existing library for synchronization primitives and instead implements its own lock class. However, this lock is but a simple spinlock that will continually check a flag to see if a resource is available. This is a textbook example of Busy Waiting. This could be addressed by using a built in synchronization primitive such as a mutex which unschedules the thread when it sees the resource it needs is unavailable.

When defining a new node, the node definition is separate from the UI system for adding it into a graph. This makes inconsistencies easy, as updating the node's name in the node system does not update the node's name in the UI system. This is an example of shotgun surgery, and can be prevented by having names defined along with the node definitions, then having the UI system use the node definition macro to generate menus.

In many places, the project has inconsistent style. Since we never agreed on a proper programming style for the project, sometimes names have inconsistent capitalization rules, different names for the same types are used, or functions are written with entirely different styles of C++. These coding style disparities can be quite drastic as well. For example, the PNG parser loads images using a custom built file reading function while audio samples and DAWdle projects are loaded using standard `fstream`. There is no technical reason why this needs to be this way, it just came down to the preferences of whoever wrote that part of the project. Even worse, DAWdle largely relies on custom allocators, but in some specific places (such as when allocating the buffer for storing a sample or allocating the memory to hold the bytecode for interpreting an expression) standard `new` and `delete` are used instead. It would not prove too difficult to refactor the code to be more consistent, but it would be very tedious and time consuming. Ultimately, we should have agreed on a concrete style guide beforehand and used some kind of static analysis tools such as linters and auto formatters to enforce uniformity throughout the program.

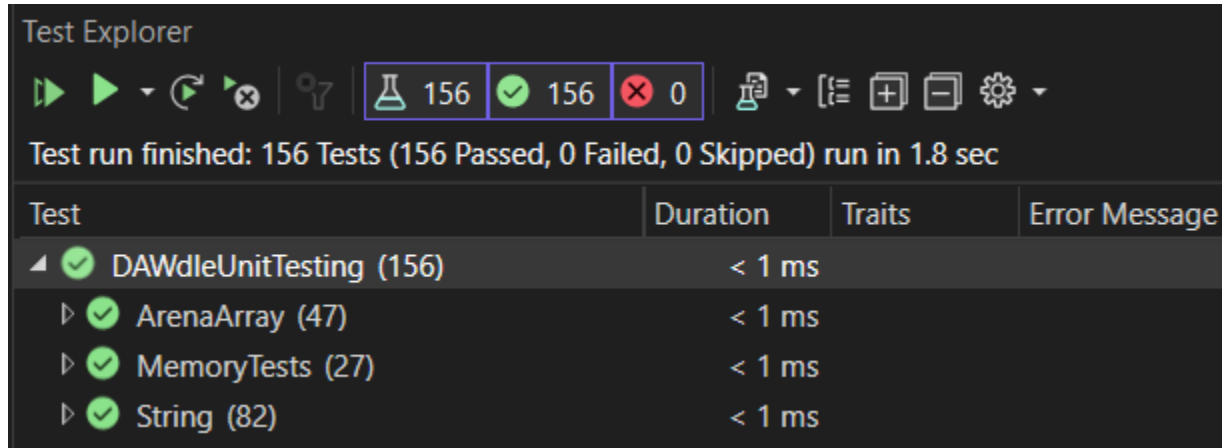
3. Verification

- I. Test Infrastructure. Describe your team's test plan, along with the tools used, and explain your choices along with recommendations if other testing tools/techniques should be adopted in the future.

Since we have such a large code base, most of it largely not unit-testable, we chose a two-pronged approach to our testing. Firstly, we did employ unit testing where applicable. Prioritizing quality over quantity, we concentrated our efforts on writing the most exhaustive test suites we could for the most important and foundational classes and functions in the project. We managed to cover two vital classes (a string class and a dynamic array class used in place of `std::string` and `std::vector` respectively) and some vital memory-related functions (mostly drop-in replacements for classic C standard library functions) in the development time we allocated for unit testing. In making these tests we used Visual Studio's built in unit test functionality with the Google gtest testing library, and used OpenCPPOverage to track code coverage. For the future we recommend implementing exhaustive unit tests for other functions and classes in the code as we were far from able to cover all the unit-testable functionality in the project. Additionally, the rigor of these tests is currently only supported by our own intuition and simple code coverage metrics, so getting some additional confirmation on that via something like mutation testing is also recommended. The second aspect to our testing approach was extensive subjective testing. Our subjective testing was feature-based, meaning that each subjective test measured the quality, performance, and auditory/visual characteristics for an isolated feature or limit. The tests are as follows: Sine Wave, Audio playback, User Interface Layout, Real-time parameter adjustment, Oscilloscope, Math Node, Performance for large graphs, Audio processing, Error Handling, Limits, Sliders, Dropdown menus, Operations, Expression Parsing, Serialization, Single Sequence, Latency, Dynamic Modification, Pause Button, Play Button, Piano Roll, List Collapse, Clear Button. The purpose of this subjective testing was to cover the important parts of our application that were not easily unit-testable (eg. the user interface, the actual node system, and the audio synthesis and playback) These areas comprise a large amount of our program, so we knew zero testing would be unacceptable. Through subjective testing we were able to effectively explore scenarios that a user might actually find themselves in while using our application at the cost of a lack

of automation for these tests. For the future we recommend even more subjective testing be done as even what we have is not exhaustive, as well as perhaps some more rigorous system for generating the scenarios to be tested.

- II. Test Results. Show the results of your tests running. If there are failing tests, create Issues in GitHub that document what functionality should be fixed to pass the tests.



Test	Duration	Traits	Error Message
DAWdleUnitTesting (156)	< 1 ms		
ArenaArray (47)	< 1 ms		
MemoryTests (27)	< 1 ms		
String (82)	< 1 ms		

- III. Test Quality. Describe how you are evaluating the quality/thoroughness of your tests and interpret those analyses. Include snapshots of the results of any tools you are using for this analysis (identifying the tools and interpreting the results).

1. Unit Testing: With thousands of lines of code to analyze, we targeted the core class used throughout all our applications. This was the DrillLib.h, a library of standard functionality Evan likes to have in his projects. We use this library in practically everything in DAWdle. The major groups of functions are the StrA class (A string class, used everywhere instead of std::string), the ArenaArrayList class (A critical data structure class, used by the node graph), and a suite of memory-related free functions that serve as drop-in replacements for classic C standard library algorithms (eg. memcpy, memset, strcmp). For these three areas alone we created more than 150 combined unit tests, consisting of both standard use cases and intentional attempts to exploit potential edge cases. Using the code coverage tool OpenCPPCoverage (results posted below), we analyzed the test coverage to ensure that every possible path of every function tested was executed in at least one test. Even when the tool reported 100% coverage we often continued adding tests for a particular

function until we felt every single reasonably unique case was covered.

```

17. INLINE void zero_memory(void* mem, U64 bytes) {
18.     __stosb(reinterpret_cast<Byte*>(mem), 0, bytes);
19. }
20.
21. DEBUG_OPTIMIZE_OFF
22.
23. #pragma warning(disable:28251) // Inconsistent annotation for ''
24. #pragma warning(disable:6001) // Using uninitialized memory. There appears to be a false positive for some functions here
25.
26. #pragma intrinsic(memcpy, memset, strcmp, strlen, memcmp)
27.
28. #pragma function(memcpy)
29. void* __cdecl memcpy(void* dst, const void* src, size_t count) {
30.     const Byte* bSrc = reinterpret_cast<const Byte*>(src);
31.     Byte* bDst = reinterpret_cast<Byte*>(dst);
32.     /*while (count-- > 0) {
33.         *bDst++ = *bSrc++;
34.     }*/
35.     __movsb(bDst, bSrc, count);
36.     return dst;
37. }
38.
39. #pragma function(memset)
40. void* __cdecl memset(void* dst, int val, size_t count) {
41.     Byte* bDst = reinterpret_cast<Byte*>(dst);
42.     /*while (count-- > 0) {
43.         *bDst++ = val;
44.     }*/
45.     __stosb(bDst, Byte(val), count);
46.     return dst;
47. }
48.
49. #pragma function(strcmp)
50. int __cdecl strcmp(const char* s1, const char* s2) {
51.     const Byte* bS1 = reinterpret_cast<const Byte*>(s1);
52.     const Byte* bS2 = reinterpret_cast<const Byte*>(s2);
53.     Byte b1 = 0;
54.     Byte b2 = 0;
55.     while ((b1 = *bS1++) == (b2 = *bS2++)) {
56.         if (b1 == '\0') {
57.             break;
58.         }
59.     }
60.     return b1 - b2;
61. }
62.
63. #pragma function(strlen)
64. size_t __cdecl strlen(const char* str) {
65.     size_t result = 0;
66.     if (str != nullptr) {
67.         while (*str++) {
68.             result++;
69.         }
70.     }
71.     return result;
72. }
73.
74. #pragma function(memcmp)
75. int __cdecl memcmp(const void* m1, const void* m2, size_t n) {
76.     int diff = 0;
77.     const Byte* bM1 = reinterpret_cast<const Byte*>(m1);
78.     const Byte* bM2 = reinterpret_cast<const Byte*>(m2);
79.     while (n-- > 0) {
80.         if (*bM1++ != *bM2++) {
81.             diff = *bM1 - *bM2;
82.             break;
83.         }
84.     }
85.     return diff;
86. }

```

```

375. struct StrA {
376.     const char* str;
377.     U64 length;
378.
379.     FINLINE const char* c_str(MemoryArena& arena) const {
380.         if (length && str[length - 1] == '\0') {
381.             return str;
382.         }
383.         else {
384.             char* result = arena.alloc<char>(length + 1);
385.             memcpy(result, str, length);
386.             result[length] = '\0';
387.             return result;
388.         }
389.     }
390.     FINLINE B32 operator==(const StrA& other) const {
391.         return length == other.length && memcmp(str, other.str, length) == 0;
392.     }
393.     FINLINE char operator[](I64 pos) const {
394.         return str[pos < 0 ? length + pos : pos];
395.     }
396.     FINLINE B32 is_empty() const {
397.         return length == 0;
398.     }
399.     FINLINE I64 find(StrA other) const {
400.         if (other.length > length) {
401.             return -1;
402.         }
403.         for (U64 i = 0; i <= length - other.length; i++) {
404.             if (memcmp(str + i, other.str, other.length) == 0) {
405.                 return I64(i);
406.             }
407.         }
408.         return -1;
409.     }
410.     FINLINE I64 find(char c) const {
411.         for (U64 i = 0; i < length; i++) {
412.             if (str[i] == c) {
413.                 return I64(i);
414.             }
415.         }
416.         return -1;
417.     }
418.     FINLINE I64 rfind(StrA other) const {
419.         if (other.length > length) {
420.             return -1;
421.         }
422.         for (I64 i = I64(length - other.length); i >= 0; i--) {
423.             if (memcmp(str + i, other.str, other.length) == 0) {
424.                 return i;
425.             }
426.         }
427.         return -1;
428.     }
429.     FINLINE I64 rfind(char c) const {
430.         for (I64 i = I64(length - 1); i >= 0; i--) {
431.             if (str[i] == c) {
432.                 return i;
433.             }
434.         }
435.         return -1;
436.     }
437.     FINLINE B32 starts_with(StrA other) const {
438.         return other.length <= length && memcmp(str, other.str, other.length) == 0;
439.     }
440.     FINLINE B32 ends_with(StrA other) const {
441.         return other.length <= length && memcmp(str + (length - other.length), other.str, other.length) == 0;
442.     }
443.     FINLINE StrA slice(I64 begin, I64 end) const {
444.         if (begin < 0) {
445.             begin = max(I64(length) + begin, 0LL);
446.         }
447.         if (end < 0) {
448.             end = max(I64(length) + end, 0LL);
449.         }
450.         if (end < begin) {
451.             return StrA();
452.         }
453.         U64 first = min(U64(begin), length);
454.         U64 last = min(U64(end), length);
455.         return StrA( str + first, U64(last - first) );
456.     }
457.     FINLINE StrA prefix(I64 amount) const {
458.         return slice(0, amount);
459.     }
460.     FINLINE StrA suffix(I64 amount) const {
461.         return slice(amount < 0 ? -amount : length - amount, I64_MAX);
462.     }
463.     FINLINE StrA skip(I64 amount) const {
464.         return slice(amount, I64_MAX);
465.     }
466.     FINLINE StrA substr(U64 begin, U64 newLength) const {
467.         U64 start = min(begin, length);
468.         return StrA( str + start, min(newLength, length - start) );
469.     }
470.     const char* begin() const {
471.         return str;
472.     }
473.     const char* end() const {
474.         return str + length;
475.     }
476.     char front() const {
477.         return str[0];
478.     }
479.     char back() const {
480.         return str[length - 1];
481.     }
482. };
483.
484. // Here's one of the C++ features I'll be using, should make string literals much nicer
485. // Technically user defined literal suffixes that don't start with an underscore are reserved, but I don't really care.
486. // If it becomes an issue, I'll change it then
487. FINLINE constexpr StrA operator""sa(const char* lit, U64 len) {
488.     return StrA( lit, len );
489. }
490.
491. FINLINE U64 total_stralen(StrA str) {
492.     return str.length;
493. }
494. template<typename... Others>
495. FINLINE U64 total_stralen(StrA str, Others... others) {
496.     return str.length + total_stralen(others...);
497. }
498. FINLINE void copy_strings_to_buffer(char* out, StrA str) {
499.     memcpy(out, str.str, str.length);
500. }
501. template<typename... Others>
502. FINLINE void copy_strings_to_buffer(char* out, StrA str, Others... others) {
503.     memcpy(out, str.str, str.length);
504.     copy_strings_to_buffer(out + str.length, others...);
505. }
506. template<typename... Others>
507. StrA strconcat(MemoryArena& arena, StrA strA, Others... others) {
508.     U64 totalLength = total_stralen(strA, others...);
509.     char* result = arena.alloc<char>(totalLength);
510.     copy_strings_to_buffer(result, strA, others...);
511.     return StrA( result, totalLength );
512. }

```



```

249. template<typename T>
250. struct ArenaArrayList {
251.     MemoryArena* allocator;
252.     T* data;
253.     U32 size;
254.     U32 capacity;
255.
256.     FINLINE void reserve(U32 newCapacity) {
257.         if (newCapacity > capacity) {
258.             data = (allocator ? allocator : &globalArena)->realloc(data, capacity, newCapacity);
259.             capacity = newCapacity;
260.         }
261.     }
262.
263.     FINLINE void resize(U32 newSize) {
264.         if (newSize < size) {
265.             size = newSize;
266.         }
267.         else if (newSize > size) {
268.             reserve(newSize);
269.             zero_memory(data + size, (newSize - size) * sizeof(T));
270.         }
271.     }
272.
273.     FINLINE void push_back(const T value) {
274.         if (size == capacity) {
275.             reserve(max(U32)(capacity * 2, 8));
276.         }
277.         data[size++] = value;
278.     }
279.
280.     FINLINE T& push_back() {
281.         if (size == capacity) {
282.             reserve(max(U32)(capacity * 2, 8));
283.         }
284.         return data[size++];
285.     }
286.
287.     FINLINE T& pop_back() {
288.         return data[--size];
289.     }
290.
291.     void push_back_n(const T* values, U32 numValues) {
292.         U32 newCapacity = capacity;
293.         if (newCapacity == 0) {
294.             newCapacity = 8;
295.         }
296.         U32 newSize = size + numValues;
297.         while (newCapacity < newSize) {
298.             newCapacity *= 2;
299.         }
300.         reserve(newCapacity);
301.         T* dst = data + size;
302.         while (numValues--) {
303.             *dst++ = *values++;
304.         }
305.         size = newSize;
306.     }
307.
308.     FINLINE void pop_back_n(U32 numValues) {
309.         size -= numValues;
310.     }
311.
312.     B32 contains(const T& value) {
313.         T* begin = data;
314.         T* end = begin + size;
315.         B32 returnVal = false;
316.         while (begin != end) {
317.             if (*begin == value) {
318.                 returnVal = true;
319.                 break;
320.             }
321.             begin++;
322.         }
323.         return returnVal;
324.     }
325.
326.     B32 subrange_contains(U32 rangeStart, U32 rangeEnd, const T& value) {
327.         rangeStart = min(rangeStart, size);
328.         rangeEnd = min(rangeEnd, size);
329.         if (rangeStart >= rangeEnd) {
330.             return false;
331.         }
332.         T* begin = data + rangeStart;
333.         T* end = begin + rangeEnd;
334.
335.         while (begin != end) {
336.             if (*begin == value) {
337.                 return true;
338.             }
339.             begin++;
340.         }
341.         return false;
342.     }
343.
344.     FINLINE T& last() {
345.         return data[size - 1];
346.     }
347.
348.     FINLINE T* begin() {
349.         return &data[0];
350.     }
351.
352.     FINLINE T* end() {
353.         return &data[size];
354.     }
355.
356.     FINLINE T& back() {
357.         return data[size - 1];
358.     }
359.
360.     FINLINE void clear() {
361.         size = 0;
362.     }
363.
364.     FINLINE void reset() {
365.         size = 0;
366.         capacity = 0;
367.     }
368.
369.     FINLINE bool empty() {
370.         return size == 0;
371.     }
372. };

```

2. Subjective Testing: the subjective testing consisted of three key steps: thinking of a situation the user might find themselves in while using the application, setting up such a situation in the application, and analyzing the actual behavior of the program in that situation. This setup would isolate a specific test, allowing for the test to solely target that UI component to ensure the quality and thoroughness of the test. Interpreting the tests would also be documenting anything that would seem off about the UI component, such as possible delays and problems with hardware the application may have. When it came to the audio generation, comparisons were made with verified working audio generations that can be found scattered throughout the web or playing a sample audio file that has an original sound file you can play. The point of the subjective testing was to test the UI component, but it was also to test if there were boundaries on the UI, such as a component potentially crashing the app if there were too many. As such, subjective testing had parts where the sole purpose was to see if something would crash the implementation, which allowed for limit testing of the application.


4. Software Engineering Team

I. Aidan Raymond (Tabris05)

- i. Other responsibilities
 1. Scrum Master
 2. Ran code coverage testing and did the writeup on it
 3. Calculated class cohesion metrics and did the writeup on it
 4. Found code smells and antipatterns and did the writeup on it
- ii. Pull Requests Contributed To:
 1. vkDAWdle (current build)
 - a. [Unit testing](#)
 - b. [Serialization](#)
 - c. [Expression Parsing](#)
 - d. [Sampler Node](#)
 2. QtDAWdle (legacy build)
 - a. [Sampler Node](#)
 - b. [Serialization](#)
 - c. [Refactor Arithmetic Nodes](#)
 - d. [Audio Output](#)
- iii. List of links to other contributions authored (i.e. Issues created, PR Reviews)

1. Approved PRs
 - a. [FFTs](#)
 - b. [Filters](#)
 - c. [Waveforms](#)
 - d. [Bug Fixes](#)
 - e. [Device Selection](#)
 - f. [Arithmetic Nodes](#)
 - g. [Initial Setup](#)

II. Steven Lam (slam210)

- i. Other responsibilities
 1. Subjective Test Maker
 2. Did the writeup for test infrastructure, test results, and subjective test quality
- ii. Pull Requests Contributed To
 1. Initial Construction: <https://github.com/ChicoState/DAWdle/pull/1>
 2. Node Toolbar: <https://github.com/ChicoState/DAWdle/pull/6>
 3. Initial Arithmetic: <https://github.com/ChicoState/DAWdle/pull/8>
 4. Arithmetic: <https://github.com/ChicoState/DAWdle/pull/9>
 5. Code Migration + PianoRoll:
<https://github.com/ChicoState/DAWdle/tree/QtDAWdle>
- iii. List of links to other contributions authored
 1. VkDAWdle Subjective Audio Testing:
 [DAWdle Subjective Audio Testing](#)
 2. VkDAWdle Community Website:
<https://github.com/Slam210/DAWdle>
 3. Tracked and notified the team about issues when doing initial subjective testing of VkDAWdle
 4. PR reviews:
 - a. Library Updates:
<https://github.com/ChicoState/DAWdle/pull/2>
 - b. Audio Output:
<https://github.com/ChicoState/DAWdle/pull/7>
 - c. Library Updates:
<https://github.com/ChicoState/DAWdle/pull/30>
 - d. Issue Fix for Audio:
<https://github.com/ChicoState/DAWdle/pull/39>
 - e. Clear Graph:
<https://github.com/ChicoState/DAWdle/pull/40>

III. **Adam Ullmann** (AdamUllmann)

i. Other Responsibilities:

1. DAWdle Project Author: Implemented sample DAWdle projects and prepared all of the demos on my machine.
2. Created an example repo to help my team with the fundamentals of audio programming using the API that we chose to use for QtDAWdle: [Example Repo](#)
3. Authored the UML diagram for this writeup
4. Evaluated the cyclomatic complexity of functions in the codebase and did the writeup on it

ii. Pull Requests Authored or Contributed To:

1. vkDAWdle (current build)
 - a. [Clear Button](#)
 - b. [Unit Testing](#)
 - c. [Filters](#)
 - d. [Waveforms](#)
 - e. [Serialization](#)
 - f. [Oscilloscope](#)
2. QtDAWdle (legacy build)
 - a. [Fullscreen Fix](#)
 - b. [Bug Fixes \(Dynamic Sample Rate & arithmetic node crash fix\)](#)
 - c. [Basic Tone Generators](#)
 - d. [Basic Input, Output, and Sine tone](#)

iii. Other Contributions:

1. Issues:
 - a. [Blue Screen Problem \(QtDAWdle\)](#)
 - b. [UI Freezes \(QtDAWdle\)](#)
2. Approved PRs:
 - a. [Piano Roll](#)
 - b. [Array Testing](#)
 - c. [UI update](#)
 - d. [Port Sampler Node](#)
 - e. [README](#)
 - f. [Control Sample Speed](#)
 - g. [Pause Button](#)
 - h. [PortAudio Refactor](#)
 - i. [Serialization \(QtDAWdle\)](#)
 - j. [UI Freeze Fix](#)

- k. [Fix Sample Rate Value](#)
- l. [Add QtNodes](#)

IV. **Evan Pottier** (Drillgon200)

- i. Other Responsibilities:
 - 1. Provide documentation for vkDAWdle
 - 2. Author the software architecture portion of this writeup
- ii. PRs authored:
 - 1. VkDAWdle (current build)
 - a. [Base code for VkDAWdle](#) (not a PR, done before we had VkDAWdle protected on main)
 - b. [Fix resize lag](#) (not a PR, done before we had VkDAWdle protected)
 - c. [UI update](#)
 - d. [Piano Roll](#)
 - e. [Fix for audio not playing without a piano roll](#)
 - f. [Fast Fourier Transform](#)
 - 2. QtDAWdle (legacy build)
 - a. [Device Selection](#)
 - b. [Fix samplerate value.](#)
 - c. [UI freeze fix](#)
- iii. Approved PRs
 - 1. [Fix arithmetic audio](#)
 - 2. [Fix white bar in fullscreen](#)
 - 3. [Move vkdawdle to main branch](#)
 - 4. [Unit Testing](#)
 - 5. [Fix blindspots in testing](#)

V. **Ray Perez** (raperez2)

- i. Pull Requests Contributed To:
 - 1. [Array Test Pseudocode](#)
 - 2. [ByteBuff Test Pseudocode](#)

VI. **Caleb Ray** (epicgy12)

- i. Pull Requests Contributed To:
 - 1. [Update test.cpp](#) (later changed to UnitTesting.cpp)