Parenthetical Asides

Nested Scope in Conversation, Games, Stories, and Music

Flyxion
Independent Researcher

July 2026

Abstract

Punctuation marks such as the parenthesis and the em dash are usually treated as stylistic choices, a matter of register rather than substance. This essay argues that they are something else: coarse, discrete notations for a continuous operation that ordinary speech performs constantly and mostly without notice, the opening, holding, and closing of nested informational contexts. The same operation recurs, independently and without coordination, across video game level design, nested shell environments and modal text editors, conversational hypnosis, musical form, and the split-audience logic of forced subtitles. Tracing this recurrence across five unrelated domains motivates a distinction between two independent dimensions of scope: temporal scope, in which a context is opened and later closed, and recipient scope, in which information is partitioned simultaneously between different audiences. Punctuation turns out to notate only the temporal dimension, and only a fragment of it. The essay closes by turning the argument on itself, noting that essays are themselves guided tours that open and close scopes, before gesturing, in three appendices, toward the category-theoretic and sheaf-theoretic language in which both dimensions of scope can be stated more precisely.

1 Introduction: The Punctuation That Isn't a Style Choice

Compare two ways of saying the same thing.

The experiment (which had already been delayed twice) finally succeeded.

The experiment—which had already been delayed twice—finally succeeded.

Most style guides treat the choice between a parenthesis and a dash as a matter of register: parentheses for something quieter and more clinical, dashes for something looser and more conversational. That is true as far as it goes, but it is not the interesting fact about the sentences. The interesting fact is that both are doing the same job with two different tools, and neither tool does it especially well.

Say the sentence aloud instead of writing it. The voice interrupts itself, changes, perhaps drops, perhaps speeds up, and then returns to where it left off. There is a pause, and the pause has a specific length that carries meaning: too short and it reads as a stumble, too long and it reads as a second thought the speaker is not sure should be shared at all. None of this survives onto the page. What survives is a mark, drawn from a very short list, standing in for all of it at once.

This essay is about that gap. Not because punctuation is an urgent problem, but because the gap is a small, visible instance of something much larger and mostly invisible: the way ordinary talk constantly opens small side-contexts, does something inside them, and closes them again, in a system with far more resolution than anything ever written down. Punctuation is the residue. The interesting part is the system that left it behind.

The claim, stated plainly and without any of the machinery it will eventually connect to, is this. Human language continually manages nested scope. Speech has a rich, continuous way of marking this. Writing compresses that system into a handful of punctuation marks, and loses almost all of the resolution in the process. Once this is visible in conversation, the same shape begins showing up everywhere else: in games, in terminals, in code, in stories, in music, in the split attention of a subtitled film. This essay's real interest is in that recurrence, not in the punctuation that happened to notice it first.

2 What a Parenthetical Actually Does

Return to the two versions of the experiment sentence and look at them more closely, because the difference between them is not decorative.

A parenthesis tells the reader something before they have read it: whatever comes next is secondary. It is stage direction inserted in advance. By the time the closing parenthesis arrives, the reader has already been told how to weigh what they just read, which is why parenthetical asides are so easy to skim past, and why reading them aloud tends to produce a quieter voice, a faster pace, sometimes an unconscious skip. The parenthesis downgrades its contents before the sentence even gets there.

The dash does not do this. It opens a clause at full volume and holds it open. There is no advance warning that what follows is minor, because often it is not; the dash is just as happy marking the most important part of a sentence as the least. *There was only one solution, repair the system,* spoken instead with a dash where a colon might otherwise sit, and nothing about that clause reads as secondary. The dash can mark an interruption, an amplification, a correction, a sudden change of direction, and it does all of this without telling the reader in advance which one is coming. The reader finds out by reading, the way a listener finds out by listening. That is what makes it feel spoken. It preserves the sentence's forward motion instead of stepping outside it.

Neither mark actually describes what happens in speech, though. Both are approximations, chosen because written language ran out of better tools around the time punctuation conventions solidified, not because either captures the thing itself. What varies in speech is continuous: how long the pause is, whether the voice rises or falls going into it, whether the pace quickens or slows, whether the register shifts. A short pause with a level tone reads as a passing thought. A longer pause with a dropped voice reads as something the speaker is setting aside, unsure whether to include at all. A pause with a rising, quickening voice reads as urgency, the speaker rushing to insert a clarification before the listener loses the thread. Written punctuation offers, at most, four or five discrete symbols for this. Speech has as much resolution as a voice can produce, which is to say, effectively unlimited.

The choice between a parenthesis and a dash, then, is not really a choice between two styles. It is a choice between two coarse buckets, each capturing part of a signal that

neither can represent in full. Writers reach for the dash more often than grammar books recommend partly because it is the bucket that loses the least: it preserves momentum, preserves ambiguity of emphasis, and leaves the reader doing the interpretive work a listener would ordinarily do by ear. That is a real advantage. It is also, on its own, a fairly narrow fact about two punctuation marks. The rest of this essay is about how narrow it actually is not.

3 Nested Scope Is Everywhere in Ordinary Talk

Stop looking at punctuation and start listening to actual conversation, and the pattern that produced it turns out to be the least surprising thing in the world, because it is constant, and almost none of it gets written down.

Someone describing their weekend stops mid-sentence to explain what happened the week before, then returns and finishes the original sentence, a word like "anyway" doing the work of marking the return. That is a scope opened inside a scope. The speaker has left the main thread, entered a smaller explanatory thread nested inside it, done what needed doing there, and returned to exactly where they left off. Few listeners find this confusing in the moment. Most track it without much effort. But notice what would be needed to write it down faithfully: not just that a digression happened, but how deep it was, how urgent, whether the speaker expected the listener to still remember the outer sentence by the time they got back to it. Ordinary punctuation does not attempt most of this. It settles for marking that something happened, not what kind, or how far in.

Self-correction performs the same operation from a different angle. A sentence starts down one path, an interruption arrives, and the sentence never returns to its original shape at all, replaced entirely by whatever the interruption turned out to matter more. Here the inner scope is not a digression to be resolved; it is an event that cancels the outer sentence's original plan. Jokes routinely exploit a version of this on purpose: a setup establishes one scope, a digression opens inside it that appears unrelated, and the punchline resolves both at once, the humor arising specifically from discovering that the inner scope was never really separate from the outer one at all.

None of this is exotic. It is the default texture of unscripted speech, present in nearly everything anyone says without a script in front of them. What is notable is not that it happens but how casually it happens, how rarely it causes confusion, and how little of the machinery managing it ever makes it onto a page. A transcript of real conversation, punctuated honestly, would be dense with dashes, false starts, and re-entries, because that is what is actually there. Edited prose smooths almost all of it away, which is one more reason the dash reads as unusually alive when a writer lets it through: it is one of the few marks left that still points at how talk actually works, rather than at how writing has decided talk should be tidied up to look.

The question worth asking next is not really about language at all. It is whether this same open-digress-return shape turns up in places that have nothing to do with sentences. It does, and it turns up in a form considerably easier to see clearly, because the boundaries are marked for the observer instead of left to be inferred.

4 Five Places the Same Pattern Shows Up

4.1 Games

Commander Keen, id Software's platformer series from 1990 to 1991, makes an unusually good test case, because its structure puts the scope boundaries in plain sight instead of leaving them to be inferred the way conversation does.

The game runs on a simple two-layer design. An overworld map lets the player move freely between locations; entering one of those locations drops the player into a self-contained level with its own enemies, hazards, and objectives. Finish the level, and control returns to the map. This is the open-digress-return shape from the previous section, externalized into level geometry. The map is the outer scope: persistent, holding onto whatever the player has accumulated, available the whole time the game is running. Each level is a scope nested inside it: local, temporary, populated with things that exist only for as long as the player is inside that level and vanish, in every sense that matters, the moment the player leaves.

The boundary conditions matter as much as the nesting itself. What happens inside a level generally stays inside it, unless the game explicitly carries something out, a key, an item, a switch thrown. That is not incidental to the level design; it is the entire mechanism by which nested scopes remain useful instead of collapsing into a tangle. A scope that leaked everything into its parent by default would make the map-level structure pointless, since no isolation would remain. A scope that leaked nothing at all would be equally pointless, since nothing gained inside a level could ever matter outside it. The design works because it is selective: almost everything stays local, and a small, explicit set of things gets exported.

Save points sharpen this further. A save does not merely record where the player is; it records the entire accumulated state of the outer scope at that instant, everything exported from every level completed so far, frozen into something that can be returned to later regardless of what happens next. It is the closest thing the game has to a snapshot of history, and it behaves exactly the way a snapshot should: it captures what has been resolved, not what is still in progress inside whatever level the player happens to be standing in when they save.

The point of every level, underneath all of this, is to find the exit. In later levels this means finding a specific key card that opens a specific door standing between the player and the way out. That image, the exit gated behind a key, recurs directly in the next section, in a domain that has nothing to do with games on its surface.

Descent, released a few years after *Commander Keen* by Parallax Software and published by Interplay Productions, a different studio from id Software, though both were fixtures of the same 1990s shareware scene, makes the same structure explicit in a way *Keen* never had the dimensionality to attempt. Where Keen's exit and key cards were things the player found by exploring, Descent gave the player a rotatable, three-dimensional cutaway map of the entire level, tunnels rendered as a navigable wireframe the player could turn and inspect from outside their own position inside it, showing exactly where each color-coded key card sat and which correspondingly colored door it opened. The color coding matches Keen's own convention closely enough to read as a shared genre grammar rather than coincidence, two unrelated teams independently landing on the same logic: rather than discovering the scope structure by wandering through it, the player could see the whole nested structure

laid out at once, from above, before deciding how to move through it. It is the difference between inferring a sentence's scope from context while reading it and being handed a diagram of every open parenthesis in advance. Both are real ways of tracking the same thing. One simply costs less attention to use, at the price of removing the discovery.

4.2 The Terminal

Many people who have spent real time at a command line already have physical, muscle-level experience of the same structure, usually without ever describing it that way.

A terminal session starts as a single shell: one scope, holding environment variables, a working directory, a command history. Running a program that spawns its own shell, entering a subshell, invoking a function that opens its own local scope, all of this nests a new context inside the one already open, the same map-to-level move *Commander Keen* makes explicit with a screen transition. The same rule about what carries over applies here too, almost word for word: a variable set inside a subshell stays inside it unless explicitly exported outward. Do nothing, and it behaves like something picked up inside a level and left behind on exit. Export it, and it behaves like a key carried back out to the map.

Terminal multiplexers such as byobu or tmux make the nesting spatial as well as logical, and they are where the difference between doing this well and doing it badly becomes impossible to miss. The disciplined way to use one is to open it once, at the top, and do everything else inside it: new panes, new windows, new sessions, all spawned from within the multiplexer already running. The undisciplined way, and it is a mistake nearly everyone makes at least once, is to open byobu, forget one is already inside it, and launch byobu again from inside itself. The second version does not fail loudly. It simply stacks quietly, and finding a way back out stops being a matter of retracing one's steps and becomes a matter of guessing how many layers deep one has gone.

There is a reason the disciplined version feels natural and the accidental version feels wrong almost as soon as it has happened, and it is the same reason evaluating a function works the way it does. Reducing a function means substituting an argument into the body of a term already being evaluated, in place, as part of the same ongoing reduction, the operation the author has elsewhere called *abstraction as reduction*: evaluation is not concealment of process but the successful, in-place execution of it [1]. It does not mean handing the argument to a second, unrelated interpreter and waiting to hear back. Spawning a pane inside a running byobu session is the first kind of move, a new context built inside the term already being reduced. Launching byobu again from inside itself is the second kind, a fresh evaluator that happens to be sitting inside another one, with no reduction path connecting them, which is exactly why getting back out feels like guesswork rather than the retracing of a sequence that was never really a single sequence to begin with.

Vim adds one further layer worth including, because it is the one place this logic turns into a joke people actually tell. Vim has two basic modes: normal mode, for moving around a file, and insert mode, for typing into it. Entering insert mode opens a context the same way entering a level does; pressing Escape closes it and returns the user to normal mode, where navigation is the default. New users often get stuck inside Vim, not because they cannot type, but because they do not know the exit condition: quitting requires a specific command, `:wq`, issued from normal mode rather than insert mode, so the first thing anyone has to

learn is not how to edit text but how to leave. Vim is, in a real sense, its own command-line interpreter wearing a text editor's clothes, and the running joke about people trapped inside it is a joke about not yet having found the level's exit, without the key needed to open the door, the same key-card logic *Commander Keen* and *Descent* both end on.

Bash's interrupt keys carry a version of the same lesson, compressed into three gestures. Control-C is not copy here; it is the interrupt that quits whatever is running in the foreground. Control-Z does not kill a process; it suspends it, paused rather than terminated, until `fg` brings it back and resumes it exactly where it stopped. None of these are variations on a single idea of "quit." Each closes a slightly different kind of scope, and knowing which one is needed is most of what separates someone comfortable at a terminal from someone stuck in a level with no visible way out, which is also the practical case for remapping Caps Lock to Escape: in a workflow built this heavily around opening and closing contexts, the exit key ends up doing more work than Enter does, and Caps Lock sits directly beside the home row's A key.

The felt sense of getting out, once a session has run long enough to accumulate real depth, is its own kind of evidence for all of this. A long session ends the way a long game of *Commander Keen* ends: with a string of `exit`s, each popping a single layer and nothing more, until nothing is left to pop. Alt-F4 does the same thing to a stack of windows. A desktop cluttered with folders opened inside folders performs the same operation spatially, each one visibly sitting inside the last, waiting to be closed in an order that is not arbitrary even if nobody stops to think about why.

4.3 Code

Programming languages make this pattern the most explicit of all, because they have to specify it precisely enough for a compiler to enforce, and several of their most familiar constructs are, on inspection, direct answers to the scope-management problem this essay has been circling.

A `try/catch/finally` block is close to a formal specification of open, interrupt, recover, and return: `try` opens a scope, an exception is an abnormal interruption of it, `catch` is the recovery clause handling that interruption instead of letting it propagate further outward, and `finally` guarantees a closing action regardless of which path was taken to get there. An uncaught exception is simply an interruption with nowhere local to resolve, which is why it keeps propagating outward, popping one enclosing scope after another, until something up the chain agrees to handle it or nothing does and the program ends, the code equivalent of typing `exit` until there is nothing left to exit from.

A closure is exported scope, made structural: a function that captures variables from the context it was defined in carries a piece of that outer scope with it even after the outer scope has, by ordinary rules, already closed. It is the key card that leaves the level in the player's inventory. A generator or coroutine that `yield`s is a scope deliberately left open mid-execution, suspended rather than closed, exactly the Control-Z case: paused, not terminated, resumable from precisely where it left off the next time it is asked to continue. A promise or an `async/await` pair is close to the recipient-scope case explored later in this essay rather than the temporal one: the calling code and the awaited operation run on different timelines, each with a partial, asynchronously updated view of what the other has done, reconciled

only at the point where the two are joined back together.

None of these constructs were designed with sentences, or levels, or shells in mind. They were designed to solve a narrower engineering problem: how to let one piece of code depend on another without either losing track of what belongs to whom, or losing track of how to get back out once it is done. That they end up looking exactly like the shape this essay has been tracing everywhere else is not a coincidence worth explaining away. It is the same evidence as the rest of this essay, in a domain built by people who had every reason to think they were solving a problem specific to their own field.

4.4 Stories

Nested storytelling is a technique with a name and a deliberate practitioner, which makes the pattern easier to see precisely because it is used on purpose rather than falling out of ordinary speech by accident.

Igor Ledochowski, working in the tradition of conversational hypnosis, builds his inductions out of stories interrupted by other stories [2]. He begins an account of something and, partway through, digresses into an entirely different story, with no obvious connection to the first. That second story might itself pause for a third. Each digression opens a new scope inside the one before it, and none are resolved until the sequence starts unwinding, in reverse, innermost first, the last-opened story finishing first and the original story finishing last, exactly where it left off.

The effect is not incidental to the technique; it is the entire point of it. A listener tracking several open stories at once is holding several unresolved threads in mind simultaneously, and holding an unresolved thread takes a particular kind of attention, the same attention a sentence interrupted by a dash demands before it resolves. Stacking enough open loops absorbs enough conscious attention that the more critical, evaluative part of listening quietly steps aside. The stories need not be hypnotic in content. The nesting alone does most of the work.

The same shape recurs across storytelling traditions that never coordinated with one another. Frame narratives such as *The Arabian Nights* embed stories within stories, each opened and, eventually, closed, sometimes chapters or nights later. Films built around nested dreams or layered flashbacks ask an audience to hold the same kind of open loop, trusting that the outer layer will still be there when the innermost one resolves. Interactive fiction and branching narrative games build nested choice trees that work the same way structurally, a decision opening a path that eventually folds back into, or permanently diverges from, the path it branched off of. None of this required language specifically, and none of it required hypnosis specifically either. What it required was an audience willing to hold an open context and trust that it would close.

What matters here is not the hypnosis itself. It is that this is the clearest possible demonstration that nested scope is not merely something language falls into by accident under the pressure of real-time speech. It is something language, and story more generally, can be built out of, deliberately, as a structural technique, with the resolution order, innermost out, mattering as much as the opening order did.

4.5 Music

Music performs the same move without any language at all, which is worth sitting with for a moment, since whatever is doing the work here is not specific to sentences or stories.

The clearest version is the oldest one. A theme is introduced, plain and memorable enough to be recognized again. It is then varied, transposed into a different key, sped up, given to a different instrument, fragmented and recombined, each variation a kind of digression from the original statement. Eventually the theme returns, recognizably itself, in a moment a listener feels as resolution without necessarily being able to explain why. Sonata form runs on the same shape at a larger scale: an exposition states the material, a development pulls it apart and wanders through related but unresolved territory, a recapitulation brings it back. The middle section is permitted to leave, sometimes quite far, precisely because the form guarantees a way back.

What makes this worth including alongside stories and games is not just the structural resemblance; it is that music makes the emotional logic of the pattern audible in a way nothing else here quite does. An unresolved variation, still mid-development, creates a specific kind of tension, the musical equivalent of a sentence that has opened a dash and not yet closed it. Resolution back to the theme releases that tension in a way listeners respond to whether or not they know anything about musical form.

Smaller-scale devices carry the same logic down to the level of a single phrase. A suspension holds a note over a chord change that would ordinarily resolve it, deliberately opening a small dissonant scope and delaying its closure by a beat or two before letting the expected note arrive, the harmonic equivalent of a sentence that pauses right before its punchline. A deceptive cadence sets up every expectation of a clean return, the harmonic signal that says the theme is about to close, and then resolves somewhere else instead, a scope the listener was certain was about to shut that opens onto a different room entirely. Question-and-answer phrasing, a musical figure stated and then echoed with a variation, works like a short call-and-response digression nested inside the larger form, opened by the question, closed by the answer, before the larger phrase continues. A fugue pushes the nesting further still, introducing a subject, then layering additional voices that enter with the same material while the first voice continues on, several scopes open and running concurrently rather than sequentially, each voice's entrance and eventual resolution tracked independently by a listener without any single voice ever needing to pause for the others.

A moment from *Hamilton* pushes this one step further and is worth pausing on directly. At one point in the show, a line is delivered as an aside, marked by a shift in the performer's delivery, held briefly, then closed by a return to the main line, the ordinary kind of scope this essay opened with. When the show is subtitled, the caption track renders that same aside in parentheses, a written parenthetical marking a spoken one. Two independent systems, a voice and a caption, arrive at the same underlying scope boundary and reach for the same notation, without coordinating, because a parenthesis happens to be the closest thing either has to a general-purpose scope marker.

That moment sits at a seam worth naming on its own, because what follows is not quite the same shape as anything covered so far. Consider forced subtitles: captions for dialogue in a language a story's own characters do not understand, an alien tongue, an untranslated phone call, that appear on screen even when subtitles are otherwise switched off. The

translation is a genuine aside, but it is not nested inside the scene the way a digression is nested inside a sentence. It runs alongside the scene, at the same time, addressed past the characters directly to the audience, who need content the characters themselves are never given. Nothing opens or closes here in sequence. Two interpretations of the same moment coexist, simultaneously, addressed to two different recipients, and the boundary between them is a boundary of access rather than a boundary of time.

5 Two Kinds of Scope

Five domains, one shape, up to a point. The Hamilton subtitle is where the shape needs a second dimension.

Everything up to now has treated scope as a matter of time. A speaker leaves the main thread, enters a smaller context nested inside it, and returns. A level is entered and exited. A shell is opened and closed. A digression is opened and, eventually, resolved. In every case the defining question has been *when*: when does this context open, when does it close, what state survives the transition back out. Parentheses, dashes, subordinate clauses, quotations, musical development sections, games, terminals, and nested stories all fit this description without strain.

Forced subtitles do not fit it, or rather, they fit a different description entirely. The aside is not separated from the main scene by time; it runs at the same time, in the same moment, alongside everything else on screen. What separates it is *who it is for*. The characters occupy one informational scope, hearing a language they do not understand as noise. The audience occupies a second scope, simultaneously, in which that same dialogue is legible. Nothing opens or closes. Two interpretations of the same moment coexist, addressed to two different recipients, and the boundary between them is a boundary of access rather than a boundary of sequence.

Once that distinction is visible, it turns out to be everywhere too, running alongside the temporal cases rather than replacing them. Dramatic irony is recipient scope: the audience knows something a character does not, for the entire length of a scene, not for a bounded interruption inside it. A stage aside, spoken directly to the audience while the other characters are understood not to hear it, is the same split made explicit by theatrical convention. A whispered confidence, an internal monologue, a compiler warning visible only to the programmer and invisible to the program's eventual users, all of these partition a single moment into simultaneous, differently accessible layers, rather than a single sequence into nested, sequentially resolved ones.

The Hamilton example sits exactly at the seam between the two kinds of scope, which is what makes it worth lingering on. The performer opens a temporal aside, marked by delivery, held for a few words, closed by a return to the main line. The subtitle track, encountering that same moment, solves a different problem entirely: not when to mark a boundary, but how to represent, typographically, a boundary one audience needs to see and another never notices is there. Two independent representational systems arrive at the same underlying scope boundary and choose the same notation for it, without coordinating, because a parenthesis happens to be the closest thing either has to a general-purpose scope marker.

That convergence is the strongest evidence here for a conclusion stronger than the one this essay started aiming at. The subject was never really punctuation, or games, or terminals, or stories, or music, taken as separate topics that happen to resemble one another. The subject is scope management: the general operation of partitioning information into distinct interpretive domains, whether those domains are separated by time, by intended recipient, by modality, or by which representational system happens to be doing the encoding. Punctuation is one notation for one corner of this, the temporal corner, and a fairly lossy notation at that. It is not the phenomenon. It is the fragment of the phenomenon that happened to get written down.

6 Punctuation as One Visible Projection

Return, now, to the sentence this essay opened with: an experiment, delayed twice, finally succeeding, with a dash doing the work of marking an aside the reader is meant to weigh at full volume.

Nothing about that sentence has changed. What has changed is what it looks like after watching the same operation play out across games, terminals, stories, music, and the split-audience case of forced subtitles. The dash is no longer a stylistic flourish. It is a small, compressed instance of the same thing a level exit is, the same thing typing `exit` is, the same thing a story's innermost digression closing is: one particular notation for the temporal half of a much larger operation that also runs on a second axis entirely, the axis of who gets to see what, running underneath dramatic irony, forced subtitles, whispered asides, and a compiler warning meant for one reader and nobody else.

What makes punctuation different from the other examples in this essay is not the operation but the bandwidth. A game can show a level transition. A terminal can show a prompt returning. A story can convey, out loud, exactly how long to hold a pause and what tone to hold it in. Writing can do none of that. It has a small, fixed set of marks, and must make that small set stand in for a signal that in speech has no real upper limit on resolution, pause length, pitch, pace, register, all continuously variable, all collapsed onto a handful of symbols the moment anyone tries to write the sentence down. The dash is not a good encoding of that signal. It is simply a better encoding than the alternatives, closer to the shape of what speech actually does, which is presumably why writers keep reaching for it more often than the style guides think they should.

It is worth stating the underlying claim once, plainly, rather than leaving it to be assembled from examples. Speech is a high-bandwidth signaling system in which scope is represented continuously, through timing, pitch, stress, tempo, and gesture. Writing projects that continuous signal onto a finite alphabet of punctuation marks. Parentheses, commas, quotation marks, and dashes are therefore not the phenomenon itself but a lossy encoding of it, chosen from a small fixed set because a page cannot hold a waveform.

Underneath that encoding problem sits a reason scope exists at all, and it is worth stating just as plainly. A scope is fundamentally a mechanism for limiting interaction. Every system that scales eventually invents one. Languages call it punctuation and subordination; operating systems call it processes; programming languages call it lexical scope; music calls it form; mathematics calls it locality. The names differ because the domains differ. The

constraint does not. Without it, every operation would affect everything else in reach, a sentence's aside bleeding into its main clause, a level's enemies persisting into the next one, a suspended process interfering with whatever runs after it. Scope creates temporary locality, a boundary inside which something can happen without immediately mattering everywhere. That is why games have levels, programs have local variables, conversations have digressions, music has developmental sections, and essays have chapters: not by convention, but because a system that could not locally contain change would have no way to make change tractable at all. Scope, in this light, is not decoration on top of communication. It is the constraint that makes communication, or computation, or composition of any kind, possible in the first place.

7 One Scope Left Unmentioned: This Essay

There is a version of scope reentry this essay has not yet touched, and it is worth naming precisely because it is the one closest to how this piece itself came to exist. Many long pieces of writing get set down and picked back up, sometimes hours later, sometimes, for the pieces that matter most, a year or more. Coming back to an old draft is scope reentry stretched to a timescale none of the earlier examples covered. There is no exit command to type, no save point loaded automatically. The writer has to reconstruct, by rereading, what the draft was doing, what it had already resolved, what remained open when it was set down. It is the same operation as `fg` bringing a suspended process back to the foreground, except the suspension might last a year, and nothing preserved the state automatically. The writer alone does that work, and doing it badly, forgetting what an earlier passage was for, why a section existed, what problem a paragraph was quietly trying to solve, is exactly the failure mode of resuming a scope without correctly restoring what it contained.

That gives away something about why essays are shaped the way they are. An academic essay is, structurally, a guided tour: it opens a scope in its introduction, promising a question or a claim, descends into whatever sections, subsections, and digressions the argument requires, and closes that scope in its conclusion, returning the reader to the question the introduction opened with, now answered or at least reframed. The familiar shape, introduction, middle, conclusion, is not a genre convention floating free of content. It is the same open-work-return pattern as everything else in this essay, applied to the act of writing and reading itself, at the scale of an entire document rather than a sentence or a scene. A reader who finishes an essay's conclusion and feels a sense of return, of having been brought back to where they started but changed by the trip, is feeling something close to what a listener feels when a sonata resolves to its tonic, or what a player feels stepping back onto the overworld map with a key in hand. This document has been performing that operation the entire way through, opening a claim about punctuation in Section 1 and only now, here, closing it.

8 Closing: What This Does Not Solve

This essay has not formalized any of it, and it is not going to, not in the main text. Naming the operation precisely, giving it the kind of structure that could be checked or built upon, is

different work, sketched briefly in the appendices for anyone who wants a fuller vocabulary, and developed at greater length elsewhere. What is offered here is smaller and, it is hoped, more useful as a starting point: once this pattern is visible clearly in one place, it does not stop showing up. A conversation, a level transition, a terminal prompt, a story's digression, a chord's return to the tonic, a caption meant for one audience and invisible to another. They stop looking like separate things that happen to resemble each other, and start looking like one operation, worn many different ways: contexts opened, held, and closed, or opened and run in parallel, with continuity, or at least legibility, preserved on the way back out.

A A Categorical Treatment of Temporal Scope

This appendix gives a formal vocabulary for the temporal notion of scope used throughout the essay, developed a step further than a bare glossary of terms. Nothing here is required to follow the main text; it is offered for readers who want the pattern stated with enough precision to be checked, without cluttering the essay itself with derivations.

Definition 1 (Scope)**.** *A* scope *is a context C together with a set of visible bindings, the information accessible while C is active. A scope may contain other scopes.*

Definition 2 (Scope category)**.** *Let* **Scope** *be the category whose objects are scopes and whose morphisms $C \to C'$ record admissible transitions between them. Two morphisms are of particular interest:* open $: C \to C_{\text{inner}}$*, which introduces a new scope nested inside C, and* close $: C_{\text{inner}} \to C$*, which discharges C_{inner} and returns to C, carrying forward only whatever bindings were explicitly exported. A* close *morphism is defined only when C_{inner} is* ready*: every computation the interior depends on has already resolved. Readiness, not mere bracket-balance, is what licenses a closure; a scope with unresolved internal dependencies has no admissible* close *available to it yet, regardless of how long it has been open.*

A sequence of nested openings, $C \xrightarrow{\text{open}} C_1 \xrightarrow{\text{open}} C_2 \xrightarrow{\text{open}} \cdots$, composed with the corresponding closes in reverse order, is precisely the shape traced by a game's map-to-level-to-map transition, a nested subshell returning through a sequence of `exit` calls, or a story's innermost digression resolving before the one that contains it. That composition is well-defined and associative is what licenses treating the essay's many examples, games, shells, code, stories, sentences, as instances of a single pattern rather than as merely analogous.

Remark 1. *Two scopes that share no dependency, neither reads from nor exports into the other, may be ready simultaneously, and their* close *morphisms may fire in either order, or concurrently, without affecting the resulting object of* **Scope***. This is the formal content behind Section 4.5's observation that a fugue's voices, each pursuing its own subject, can be tracked independently and resolved without any one voice needing to wait on the others: concurrent readiness, not merely sequential nesting, is already available within the category as defined.*

Definition 3 (Execution history)**.** *Let* **Hist** *be the category whose objects are execution histories, finite sequences of events, and whose morphisms are history extensions, one history embedding into another as a prefix.*

Definition 4 (Stack functor)**.** *Define* $\Sigma : \mathbf{Hist} \to \mathbf{Scope}$ *sending a history* h *to the scope* $\Sigma(h)$ *currently in force after executing* h*, and sending a history extension* $h \hookrightarrow h'$ *to the corresponding composite of* open *and* close *morphisms connecting* $\Sigma(h)$ *to* $\Sigma(h')$*. Functoriality,* $\Sigma(h' \circ h'') = \Sigma(h') \circ \Sigma(h'')$ *under history concatenation, is exactly the claim that the current scope depends only on the sequence of opens and closes so far, not on any information outside that sequence, which is why two systems with identical histories of scope transitions are always in the same scope regardless of what else has happened around them.*

Remark 2. *Function evaluation via beta reduction, described informally in Section 4.2, is the special case of* **Scope** *in which* open *is substitution of an argument into a term already under evaluation, and there is no morphism corresponding to spawning an unrelated second evaluator; any such spawning falls outside* **Scope** *as defined here, which is the formal way of stating why relaunching a terminal multiplexer from inside itself does not compose cleanly with the session already running. The exception construct discussed in Section 4.3 is the case where a morphism* $\text{close}_k : C_{\text{inner}} \to C$ *is applied not to the immediately enclosing scope but to some* C *several opens back, skipping the intervening scopes entirely, a controlled violation of strict stack order that a language's exception-handling semantics must define explicitly rather than leaving implicit.*

Definition 5 (Export as pushout)**.** *Let* C_{inner} *be a scope nested inside* C*, and let* $B \subseteq C_{\text{inner}}$ *be the bindings explicitly marked for export. The closed scope* C' *resulting from* close *is the pushout of* $C \leftarrow B \rightarrow C_{\text{inner}}$ *in* **Scope***: the smallest scope containing both* C*'s original bindings and the exported bindings* B*, identified along their shared names, with everything in* $C_{\text{inner}} \setminus B$ *discarded rather than carried forward.*

Remark 3. *This is the precise sense in which export is a controlled leak rather than an accident. An ordinary variable assignment inside a subshell, with no export, corresponds to the degenerate pushout along* $B = \varnothing$*, in which* C' *is simply* C *unchanged; exporting everything corresponds to the other degenerate case, the pushout along* $B = C_{\text{inner}}$*, in which no isolation survives the closure at all. Every level of a game, every closure in a program, and every subshell in a terminal sits somewhere between these two extremes, and the essay's repeated observation that the interesting cases are the selective ones is the claim that most useful scopes correspond to a proper, nontrivial choice of* B*.*

Proposition 1 (Well-formedness)**.** *A sequence of scope transitions is well-formed if and only if every* close *is preceded, without an intervening unmatched* close*, by a corresponding* open*. Equivalently, the sequence of transitions, read as a word in* $\{\text{open}, \text{close}\}$*, is a balanced sequence in the usual sense of matched parentheses.*

This is stated without proof, since it amounts to the observation that **Scope**'s morphisms compose along a stack discipline: the most recently opened scope is the first eligible to close. The proposition is the formal content behind the essay's repeated observation that nested scopes resolve innermost-first, whether the domain is a sentence, a level, a shell, or a story, and the exception case of the preceding remark is precisely the controlled exception to it that a language's semantics has to name.

B A Sheaf-Theoretic Treatment of Recipient Scope

The temporal picture of Appendix A does not capture recipient scope, the axis introduced in Section 5 to account for forced subtitles, dramatic irony, and similar cases in which

information is partitioned simultaneously rather than sequentially. This appendix develops the language in which that axis is more naturally stated, treating temporal and recipient scope as genuinely different mathematical objects rather than two dialects of the same one: the first is naturally a tree or stack, governing sequencing; the second is naturally a system of overlapping local sections, governing locality.

Definition 6 (Audience poset)**.** *Let $\mathcal{U}$ be a set of audiences, partially ordered by informational access, so that $u \leq u'$ means every recipient with access level u also has access to everything a recipient at level u' has access to. Concretely, in the subtitle example, a bilingual viewer and a monolingual viewer occupy different, non-nested positions in $\mathcal{U}$, since neither's access is strictly contained in the other's.*

Definition 7 (Content presheaf)**.** *Let F assign to each audience $u \in \mathcal{U}$ the content $F(u)$ accessible to that audience at a given moment, with restriction maps $F(u') \to F(u)$ whenever $u \leq u'$, recording that content visible to a more broadly informed audience restricts consistently to what a less informed audience sees.*

Remark 4. *Forced subtitles are, in this language, a section of F defined only over the sub-poset of audiences requiring translation, glued onto the underlying dialogue track visible to every audience regardless of language. The essay's observation that the subtitle track and the performer's vocal delivery independently arrive at the same notation, a parenthesis, for the same underlying boundary, is the informal content of a gluing condition: two local sections of F, one acoustic and one typographic, agree on their overlap, namely, the fact that a bounded aside is occurring, even though the sections were constructed by entirely different processes.*

Proposition 2 (Locality of dramatic irony)**.** *Dramatic irony, as used in Section 5, is the case $F(u_{\text{audience}}) \supsetneq F(u_{\text{character}})$ sustained over an extended interval rather than a single bounded moment; it differs from a forced-subtitle aside only in duration, not in kind.*

Again, no proof is offered or needed; the proposition simply records, in the vocabulary of this appendix, the essay's claim that recipient scope admits both brief, aside-like instances and extended, scene-length instances, in the same way temporal scope admits both a single interrupting dash and an entire nested level.

C Combining the Two Axes

Appendices A and B have so far been developed independently, matching the essay's own claim that temporal and recipient scope are orthogonal. They need not stay independent, and combining them gives a sharper picture of what a moment like the Hamilton subtitle actually is.

Definition 8 (Scope fibration)**.** *Define a functor $\mathcal{F} : \mathbf{Scope}^{\text{op}} \to \mathbf{Sh}(\mathcal{U})$ sending each scope $C \in \mathbf{Scope}$ to a sheaf of content over the audience poset $\mathcal{U}$, the audience-indexed view of what is accessible while C is active, and sending each* close $: C_{\text{inner}} \to C$ *to the corresponding restriction of sheaves. The category of pairs (C, s) with $C \in \mathbf{Scope}$ and s a section of $\mathcal{F}(C)$, equipped with the evident projection back to* **Scope***, is the Grothendieck construction of $\mathcal{F}$, a single fibred category in which temporal position and audience-relative content are tracked together rather than in two separate bookkeeping systems.*

Remark 5. *Under this construction, the Hamilton example is a single object (C, s) where C is the temporal scope of the aside, opened and closed by the performer's delivery, and s is a section of $\mathcal{F}(C)$ that happens to restrict differently across $\mathcal{U}$: to viewers who do not need the caption, s restricts to something already legible in the performance itself; to viewers who do, s restricts to the parenthesized caption text. The essay's observation that a voice and a caption independently converge on the same notation is, in this language, the observation that two different constructions of a section of $\mathcal{F}(C)$ over overlapping parts of $\mathcal{U}$ satisfy the sheaf gluing condition without having been designed to. Forced subtitles more generally are the case where $\mathcal{F}(C)$ is genuinely non-constant across $\mathcal{U}$ for the entire duration of C, rather than only at isolated points inside it, which is the fibred-category way of stating that some scenes are built, structurally, to be watched differently by different audiences at once.*

This fibration is offered as a direction rather than a finished theory. Making it precise enough to compute with, choosing a topology on $\mathcal{U}$, verifying $\mathcal{F}$ actually satisfies the sheaf axioms rather than merely resembling them, and working out what the fibration predicts in cases beyond the ones this essay has considered, is left for the larger formal treatment this appendix gestures toward rather than completes.

D Cognitive and Historical Grounding

The main text deliberately avoids leaning on citation-heavy argument, on the view that the pattern it describes should be recognizable before it is footnoted. This appendix collects, briefly, some of the supporting context a reader might reasonably want.

The cost of holding an open context is not merely a metaphor borrowed from computing. Working memory research finds that nested or hierarchical tasks impose a real, measurable load that increases with depth of embedding [3], and that switching between a broad, outer task and a narrower, embedded one carries its own cost distinct from the cost of either task alone [4]. Language processing shows a parallel structure at the neural level, with hierarchical, recursively embedded syntax engaging regions, including Broca's area, not equally engaged by simpler, non-embedded sequences [5]. Developmentally, the capacity to track nested rule structures, of which nested game rules and nested grammatical clauses are both instances, is acquired gradually rather than present from the start [6].

The historical record of computing shows the same open-close discipline being arrived at independently, more than once. Early Unix shells introduced nested subshells specifically to isolate process state, a design choice that shaped the scoping rules described informally in Section 4.2 [7]. The broader shift from flat to hierarchical file systems runs on the identical logic: a directory nested inside another directory isolates its contents by default, and requires an explicit path to be referenced from outside, in exact analogy to the export rules governing subshell variables.

Musical expectation, finally, has its own empirical literature. Listeners form measurable expectations about how a musical phrase will resolve, and the interplay between fulfilled and delayed expectation is a large part of what produces the felt tension-and-release structure discussed in Section 4.5, from the single suspended note to the fugue's concurrent voices [8].

None of this material is required to follow the essay's central claim. It is included because a reader who wants to press further than recognition, into why the pattern feels the way it

does, deserves a starting bibliography rather than a bare assertion that such research exists.

References

[1] Flyxion. *Abstraction as Reduction*. Independent Research.

[2] Ledochowski, I. *The Power of Conversational Hypnosis: A Practical Guide to Advanced Hypnotic Influence*.

[3] Baddeley, A. (2003). Working memory: Looking back and looking forward. *Nature Reviews Neuroscience*, 4(10), 829–839.

[4] Monsell, S. (2003). Task switching. *Trends in Cognitive Sciences*, 7(3), 134–140.

[5] Friederici, A. D. (2011). The brain basis of language processing: From structure to function. *Physiological Reviews*, 91(4), 1357–1392.

[6] Piaget, J. (1962). *Play, Dreams, and Imitation in Childhood*. Norton.

[7] Ritchie, D. M., & Thompson, K. (1974). The UNIX time-sharing system. *Communications of the ACM*, 17(7), 365–375.

[8] Huron, D. (2006). *Sweet Anticipation: Music and the Psychology of Expectation*. MIT Press.