CV2 Audio Player, Media Chips, and Samples

We need a CV2 sampler for parity with CV1! Let's take this opportunity to answer some questions about how we want to unify the architecture of "Media Players" in UGC, and break ground on a paradigm we can extend to other media types.

Goals

1. Recreate the function of the Sampler for CV2; players should be able to accomplish all the things that the V1 sampler is intended to do
    a. Record a clip
    b. Trim the ends of this clip
    c. Clear a clip
    d. Choose an audio channel
    e. Toggle whether or not a clip interrupts another clip that's still playing
    f. Choose between 2d and 3d audio, and where that 3d position is
    g. Set speed
    h. Set volume
    i. Play, pause, and stop the clip
2. Distill shared media functions into common media chips wherever possible
    a. "Player" and "Media file" should be separate atomic units
3. Consolidate all UGC audio players into the same system
4. Unlock some future power and flexibility in audio design for our creators
    a. Easier and less ink-intensive to toggle between different audio files
    b. Audio Player should have Synced and Unsynced options like SFX chip
    c. Investigate breaking away from unity's default (and limiting) audio tools
5. Establish a "media editing" UX that can expand to other types of media
6. Interfacing with the audio recorder is as intuitive to Tracy as the V1 sampler
7. Audio clips can be looped as seamlessly as V1
8. Audio clips can be sequenced easily
9. Audio settings should be exposed in units and ranges that make sense for audio design
    a. I.e. volume should be a percentage, even if this is an abstraction of unity's default settings
10. We can distinguish sanely and automatically between audio that should be pre-loaded and audio that should stream
11. Release our first party UGC audio files as media files compatible with this system
    a. Do so without flooding the palette with noise
    b. Adding new audio files in the future should be easy for a non-developer to do (documentation)
12. Anticipate new maker pen menu when designing controls; tools belong in the tools menu

13. Audio is no more difficult to moderate than it was previously
    a. Belief statement: longer clips don't necessarily create moderation problems! If anything, it's easier to take down one file than many.
14. Sunset the old SFX gadget to this new system without breaking old content
15. Stretch goal: Audio clips can be triggered with sufficiently low latency that they can be used to make musical instruments

Non-Goals

1. It's not a goal that the Sampler V2 work in exactly the same way that the CV1 sampler did. As long as the same outcomes are achievable, the process doesn't need to look the same, i.e. what was a config setting may now be a circuit input.
2. We're not attempting to address any exploits or edge case uses of the V1 sampler, though we should catalog them for posterity
3. Calling this one out specifically because it's an extremely common request - it's not a goal to facilitate the easy uploading of complete MP3s from Rec.net to reference in-game.
4. Any sophisticated audio or music fx will have to wait for an audio engine upgrade. It's not a goal to try to force this in right now in a suboptimal way, but let's make sure we don't block off those avenues later

The gist

There is one unified audio player, and it can be used in conjunction with media chips that encompass all types of audio in the game. The player is a component that handles synchronization and playback, and the media chips are a new category of chip that contain a reference to a file.

Media-related settings are on the chip, playback-related settings are on the player, and all editing tools are in the Maker Pen menu and accessed via the Edit tool. This should be considered a blueprint for other types of media, such as holotars, when we expand into those systems.

Why do this?

CV2 is all about atomic, modular parts, and consistency is important for combinatorial power! Looking at our four audio gadgets from V1 (the SFX chip, the sampler, the ambient radio and the music radio) we see four self-contained, one-off systems which all behave according to different rules. This is further true for each other type of media interface that our UGC tools offer. This translates to our creators as a bunch of inconsistent tribal knowledge that simply has to be memorized.

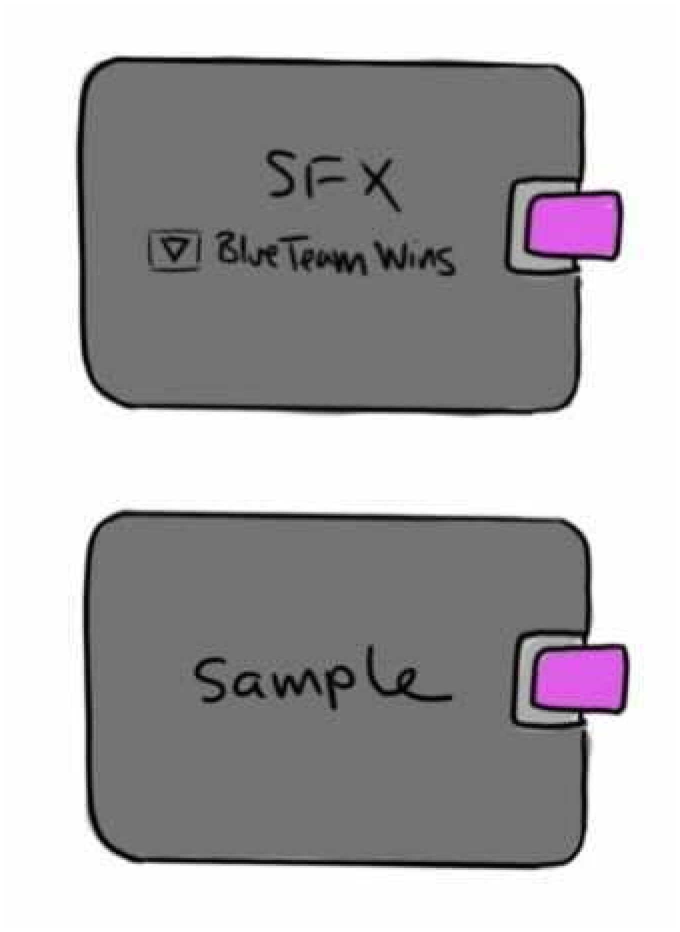
There are also a number of things that are very difficult or expensive with the current tools, including using circuits to choose between a large number of different sounds, or sequencing many Samplers to continue playback seamlessly.

Recreating those abilities with atomic parts, and adhering to the paradigm of tools belonging in the maker pen menu, we can distill these into two modular pieces that allow our creators more power and flexibility.

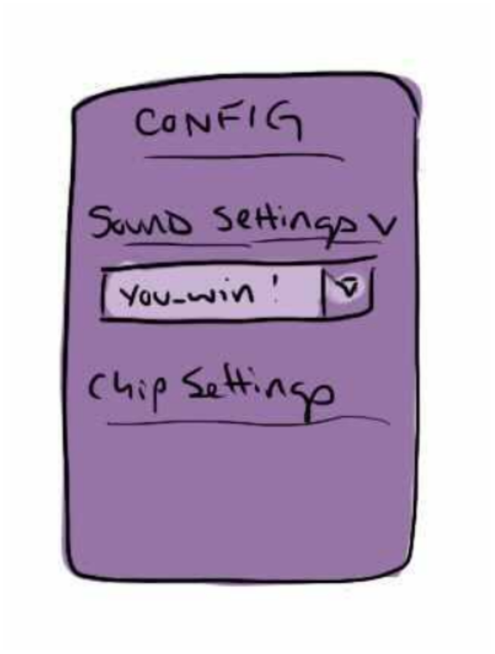
Media Chips

Media is stored on a new category of chip that, while it most resembles Variables from among our existing chips, can be thought of more like a Constant, referencing an unchangeable piece of data. The Sampler and the new Doors will use a similar chip! Unlike most current chips, these chips can be saved with state, i.e. they maintain the reference to the file that they're referencing across room saves and in inventions.

We should have one type of media chip for each Type of media we require, with new rules mandating a new type. I see three for audio: **SFX, Audio Tracks, and Sample.** SFX includes our list of first party sound effects - anything that currently lives in the SFX gadget. Tracks includes our first part music and ambient tracks - longer clips such as those in the radio props. Sample is for user-recorded audio.



Within the config menu for the SFX and Tracks chips, there's a dropdown containing all of our available sfx and music tracks, similar to the current SFX component. Most of the other settings that currently exist in the SFX component will be moved to the Media Player.



Within the config menu for the Sampler, there's a button that shortcuts to the new page in the maker pen menu which will expose the controls. Using the Edit tool on the Sampler will achieve the same goal. (I've drawn it above with a button on the chip face similar to the circuit buses, as another option.)

Why this instead of media variables? Referencing a media clip in a variable and saving that reference into inventions is - essentially - setting the home value of that variable to that reference. We're some time away from implementing functions, properties, and actual home values, so attempting to do something like home values now has the potential to back us into a corner later. Using these "constants" offers us a migration path for the future, where these media chips become delivery mechanisms for first party media but can be used to set media variables that behave in the same way as ordinary variables.

These "constants" have another advantage, in that this is a pattern extensible to other things like color, object references, and other list-selectable, non-chip data that we may want to expose to a CV2 graph.

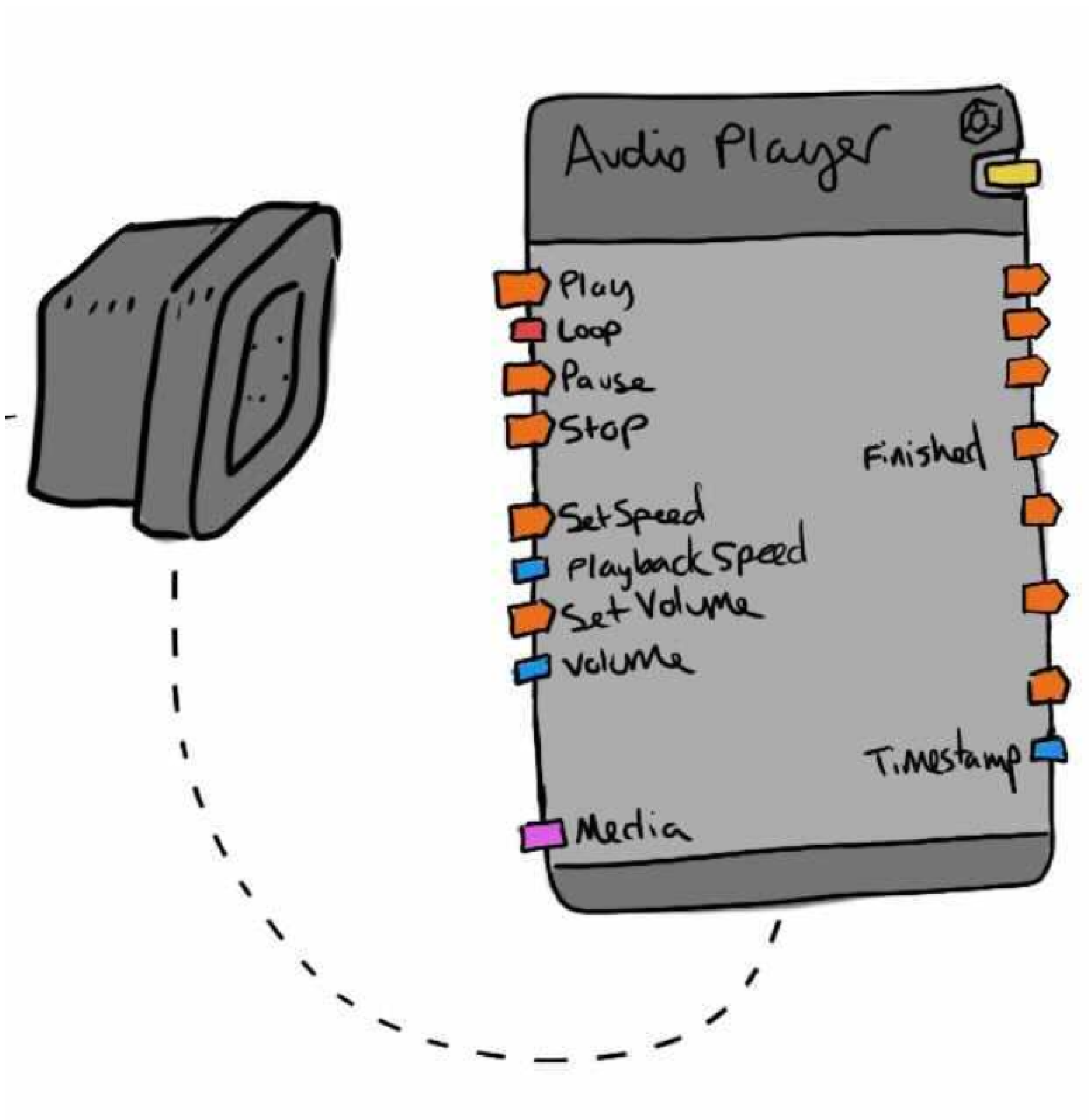
Why are tracks and sfx different types? It's possible that they won't need to be! I'm assuming that there may be a difference between them because sfx are short, pre-loaded clips and music is longer, streamed-in audio.

The config menu experience of picking sfx isn't great! True! That probably isn't within scope to address in this project, but it will be no worse on a media chip than it is in the sfx gizmo. We could take it on here if we wanted to, though it would likely affect scope.

You've drawn these in pink, do we need a different color for these types? Maybe! I drew it that way for clarity, and don't have super strong feelings, but we should discuss it. Media will be a category and not a one-off, and yellow is pretty overloaded, so it's worth considering.

Playing Audio

Playing all types of audio is done via the Audio Player, which is a component. The visual mesh is constructed out of our modular component set, and is gadget-visible like all others. This component should be a valid target for both Clamps and Set Position, though it's possible (depending on tech constraints) that the position will only matter at the moment when Play is executed.



All properties related to playback are found on the player, either as a port or as a config setting.

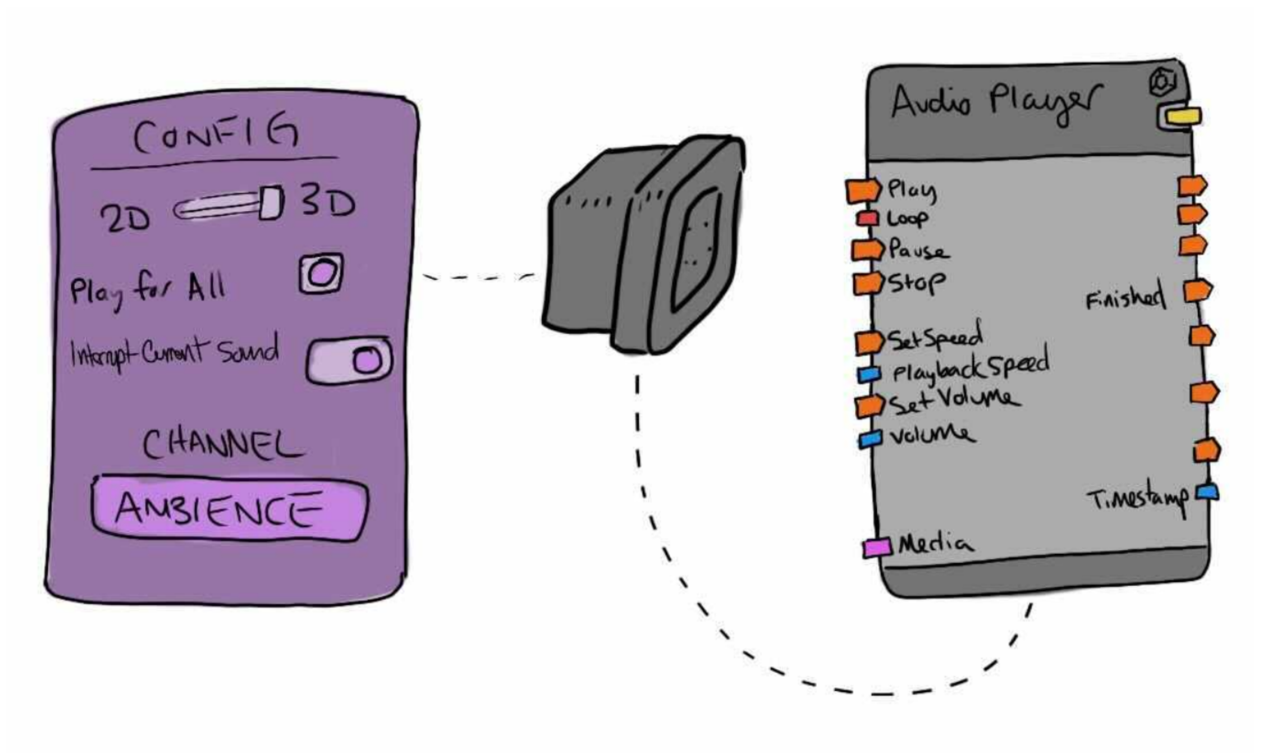
2d/3d toggle: Config setting, similar to the existing SFX. This toggle controls whether sound is 2d (has no origin) or 3d (originates at the position of the Audio Player's visual mesh).

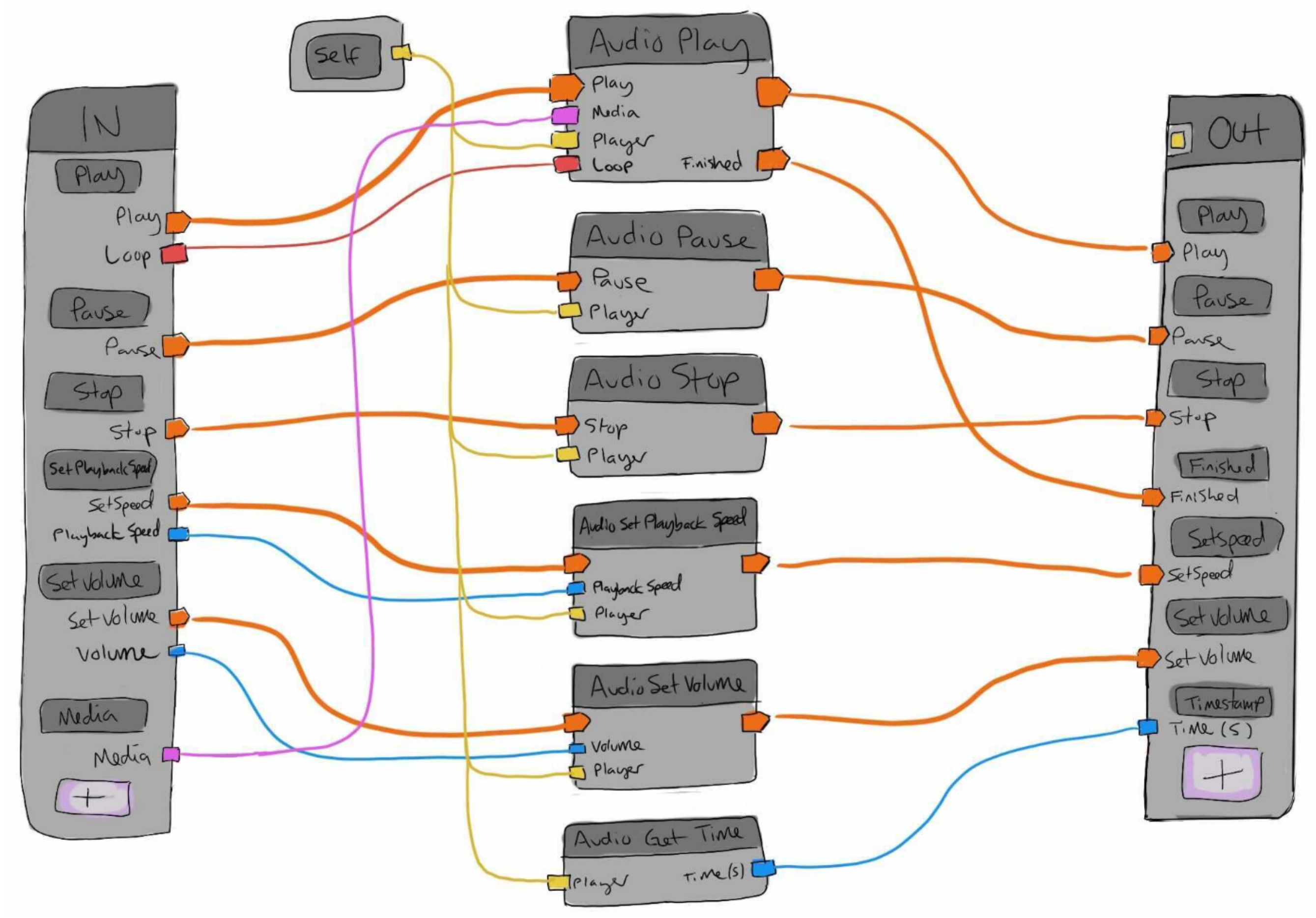
Synced: Also similar to the existing SFX V2 component, this controls whether or not the audio playback is local or synced.

Interrupt Current Sound: If set to Yes, new Play executions will interrupt other clips currently being played on the same audio player. If set to No, new play executions will fail while another clip is playing. (Do we need a Failed output for this?) This setting has no effect on other audio players.

Channel: A direct port from our existing SFX settings. All media played on this audio player will be tied to the volume slider of Ambience, Voices, Sounds, or Music.

Ports: See diagram and chip list below

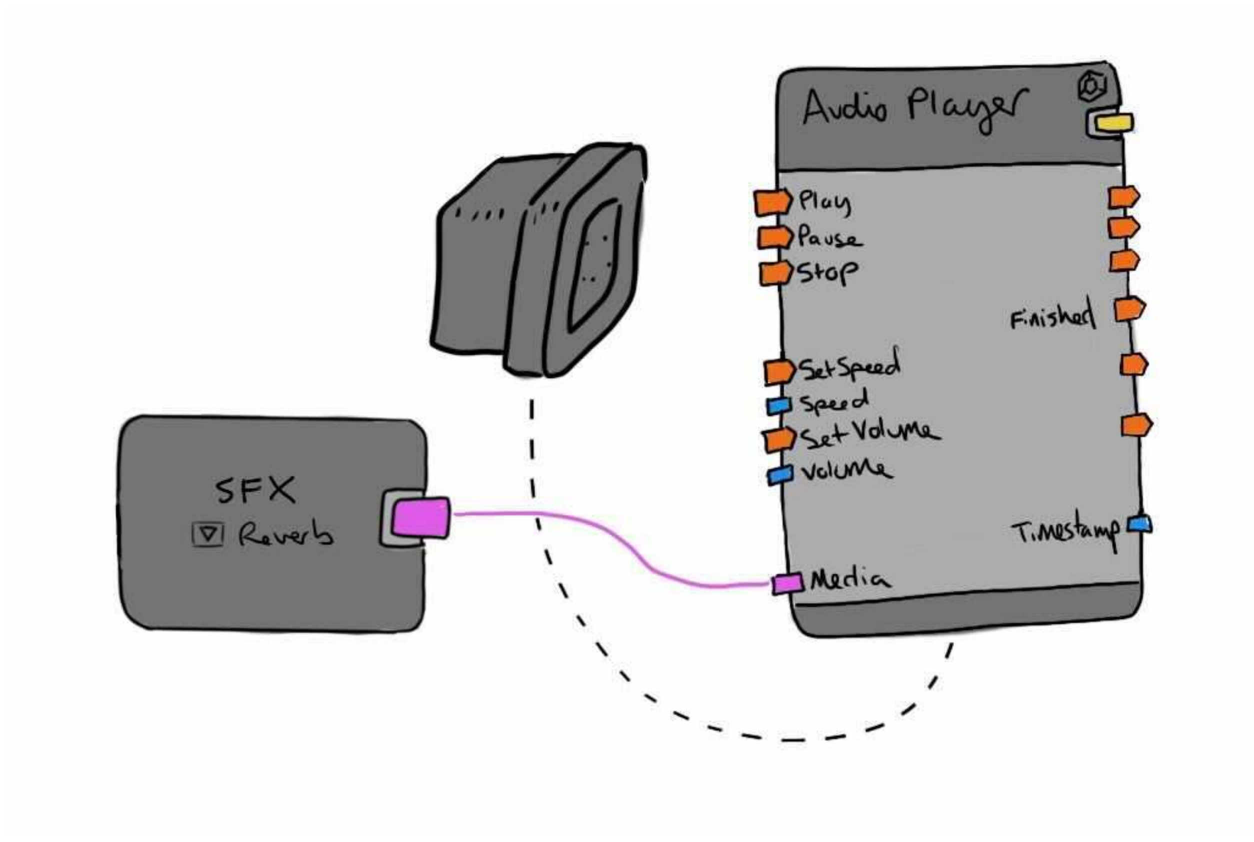




How do you loop audio? Cycles! Wire from the Finished port back to Play. Under the hood, this has to set a property on the audio source and is probably searching back for the first loaded sample (i.e. how we currently do it)

How do you sequence audio? The old way still works, but you could also increment a counter on "finished" and pipe the value of the counter into a "data switch" (see below) on the way back to play.

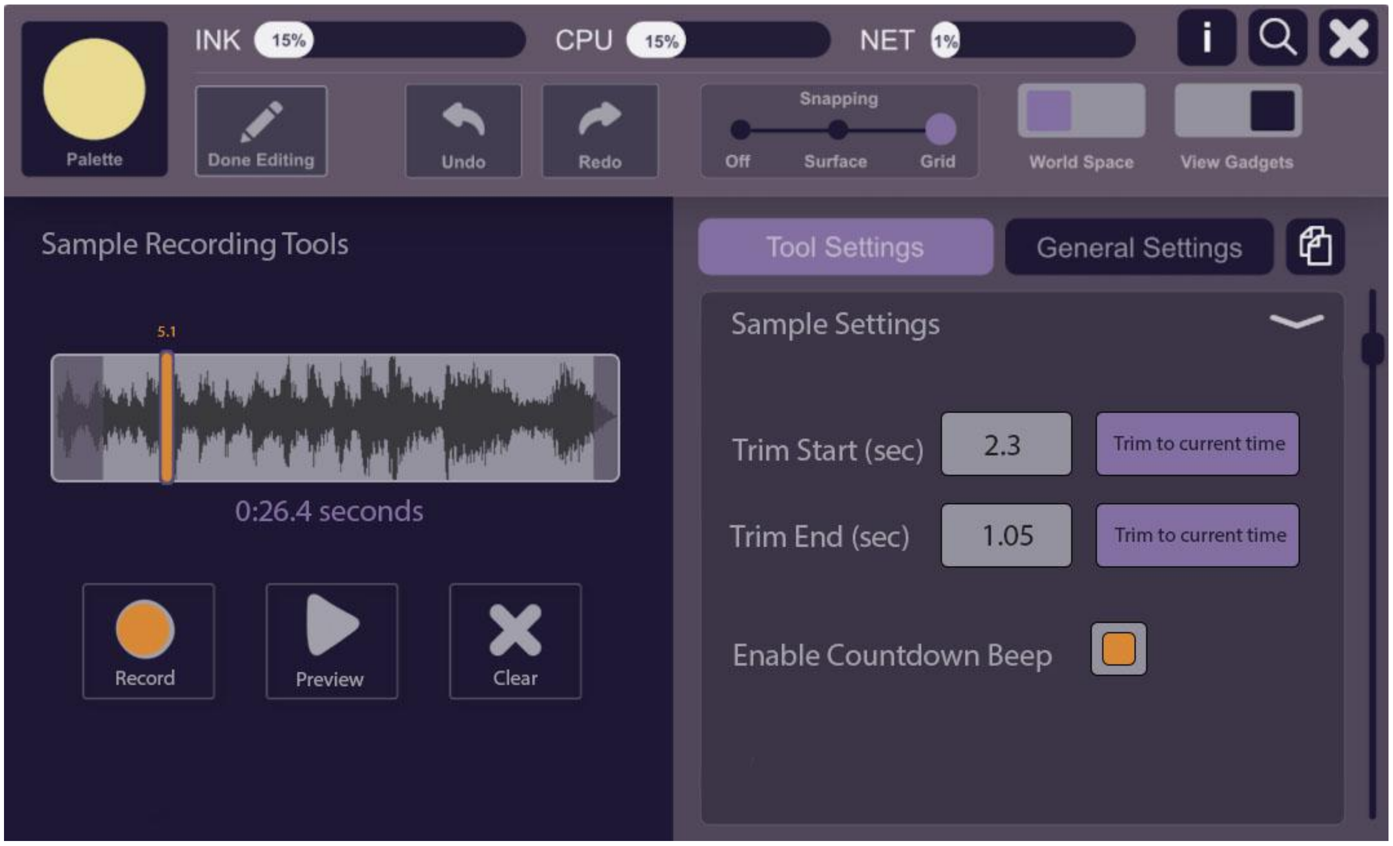
So I have to spawn two different things to get audio working? Yes! But we can spawn them as a set if we choose to, i.e. I spawn a SFX player from the Palette, but it's really a combination of SFX chip and Audio Player..



Recording Audio, i.e. the actual “Sampler” part

Recording is done by Editing the Sample chip. This opens a page in the new RRUI maker pen menu that exposes the recording controls. A button on the chip (like the + on the circuit buses) or in the config menu can also be used to shortcut to this editing screen.

Any controls related to creating, editing, or deleting a sample belong in the maker pen menu. This is the same rule we should use for other media going forward.



0:26.4 seconds

Trim Start (sec) 2.3

Trim End (sec) 1.05

Enable Countdown Beep

Features of this screen include:

- A visual representation of the audio waveform
- A visual representation of the current time
    - Stretch goal: The “playhead” is a grabbable slider that snaps to the nearest unity sample
    - Even stretchier goal: The “playhead” snaps to the start/end of audio
- A visual representation of the “offset” or “trim” values, and easy ways to set them. Trimming is non-destructive, but does affect the timing of the “Finished” exec.
- The ability to turn the countdown beep on and off (the visual countdown always remains)
- An indication of the clip length
- Buttons to record, preview (which becomes pause while the preview is playing,) and clear your clip. Record is greyed out if there’s any data present; you must Clear before you can record again. Clear asks you to confirm the operation.

These controls behave much like they did on the old sampler, with the exception of being able to visualize your waveform and offsets. There’s a beep on “Record” to count you down before the actual recording; however, we should try to ensure that this beep doesn’t bleed over into the recording itself, which is a common problem today. If that proves impossible for latency reasons, the ability to turn it off and easily edit it out will suffice.

Can I edit the other audio chips?
Stretch goal: you can trim and preview the other audio chips, but not clear or record. We can use the same screen minus, with inapplicable buttons greyed out. If this implies a lot of technical complexity with the Type or how the chips are storing data, I'm fine to abandon it!

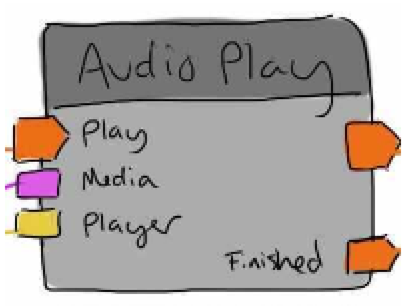
Can I edit samples that are part of inventions? They should play by the other invention rules, so if the invention is set to "Use Only," no, but otherwise yes.

Should there be a limit on length? Yes, but we should allow longer clips than we do right now if it's possible. This limit should be defined by tech requirements and perf - copyright isn't a factor What's a sane length, and what are the factors that limit it?

New Chips

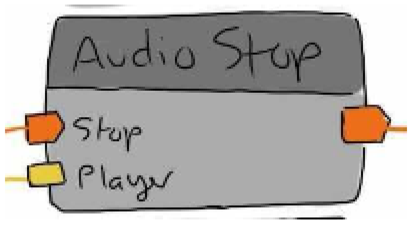
Audio Play

- Plays the clip. Requires an exec signal, a reference to the audio player's Self (to synchronize clip data), and a reference to the media. The play chip controls which media the player is synchronizing across all of its functions.
- Output: A pass through exec and a "finished" exec. The finished exec fires when playback reaches the end of the clip, accounting for any trimming.



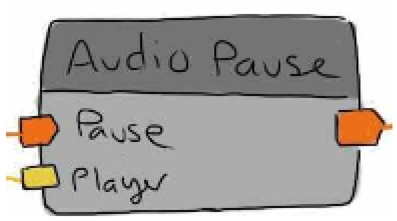
Audio Stop

- Stops the clip. Requires an exec signal, a reference to the audio player's Self (to synchronize clip data.) Outputs a passthrough exec.



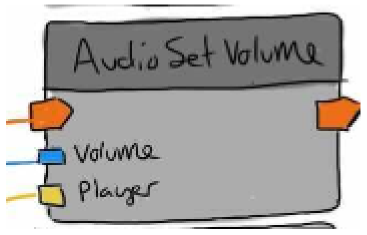
Audio Pause

- Pauses the clip. Requires an exec signal and a reference to the audio player's Self, to synchronize clip data. Outputs a passthrough exec.



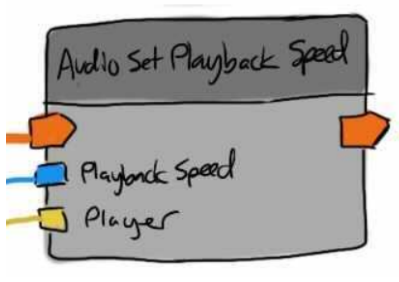
Audio Set Volume

- Sets the volume whenever this is executed. Requires an exec signal, a reference to the player's Self for synchronization, and a Float (for percentage) between 1 and 100. Outputs a passthrough exec.
- If the clip is currently playing when this exec fires, the volume should be adjusted mid playback. A 30hz lerp should result in a reasonably smooth transition, and a ~30hz flip flop between two similar values should result in a vibrato-like effect.
- If the float piped in is below zero or above 100, playback volume should be capped at those bounds.



Audio Set Playback Speed

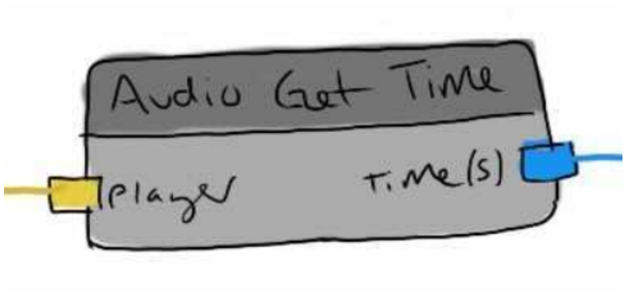
- Sets the speed of playback whenever executed. Requires an exec signal, a reference to the player's Self for synchronization, and a Float (between -2 and 5) for speed. Outputs a passthrough exec.
- If the clip is currently playing when this exec fires, the volume should be adjusted mid playback. A 30hz lerp should result in a reasonably smooth transition.
- A speed of 0 is perceived by the listener as "stopped." A speed of 1 is normal speed. Other increments are multiplicative, i.e. a speed of 2 is 2x, a speed of 5 is 5x, a speed of .5 is half speed, and a speed of -2 is sped up but backwards.
    - Playing clips backwards may be difficult for reasons I don't anticipate. I'm also satisfied with 0 to 5, where 0 is effectively stopped and 5 is 5x.



In a future world where we're able to divorce speed and pitch, Playback Speed would remain untouched, and we'd add new Speed and Pitch execs.

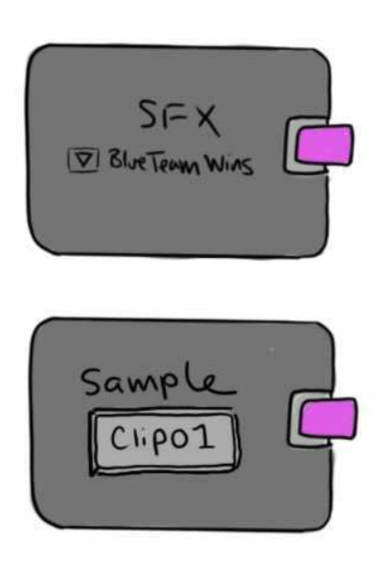
Audio Get Current Time

- Takes a reference to the audio player's Self and gets the timestamp in seconds (down to two decimal places) of the current clip at the moment the request is made.
- (Do we also need a way to skip to a certain time?)
- We get the timestamp of the unadjusted clip when we're interacting with speed
- Playback speed is linked to pitch; we can't circumvent this yet. In the future, we can differentiate between "playback speed" tied to pitch and "speed/pitch" as independent values.



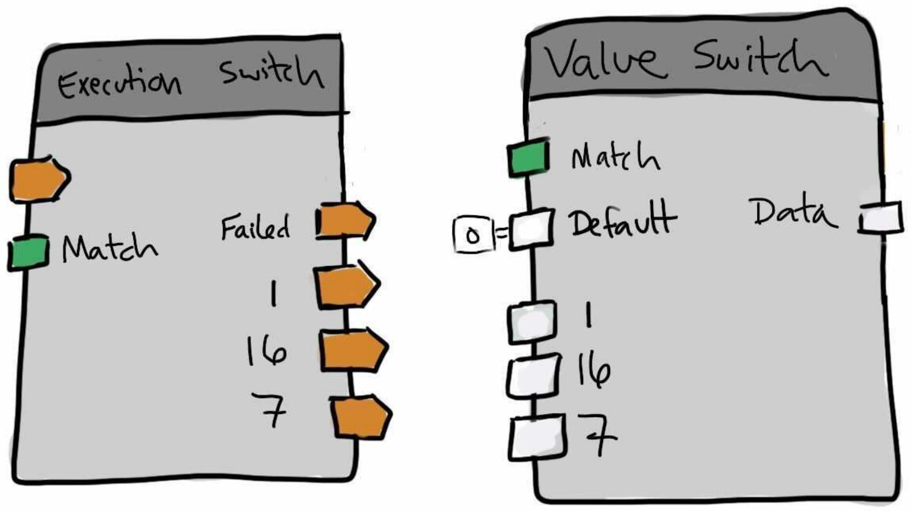
The media chips (and corresponding types)

SFX and Audio Track: choose the media via a dropdown in the config menu
Sample: record your own media via the Edit tool



Data Switch

This will likely already have been done for Garden Party, but here's the doc (REDACTED) if not.


Execution Switch
Match
Failed
1
16
7
Value Switch
Match
0
Default
Data
1
16
7