**Sinecraft Tutorial**

Introduction:

Welcome to the first official Sinecraft tutorial! Provided are a few tasks for you to complete to get you acquainted with the basics of a 3D virtual environment for digital signal processing. All tasks are highly recommended, though feel free to experiment or quit at any point.

Task 1: Basics

Controls:

WASD to move around

Q to open the menu

E to hide spawn menu

1-9 Spawn a block

Click and drag blocks to move them

Shift-Click to delete them

Click on sockets to start connecting them

Click and hold to drag sliders

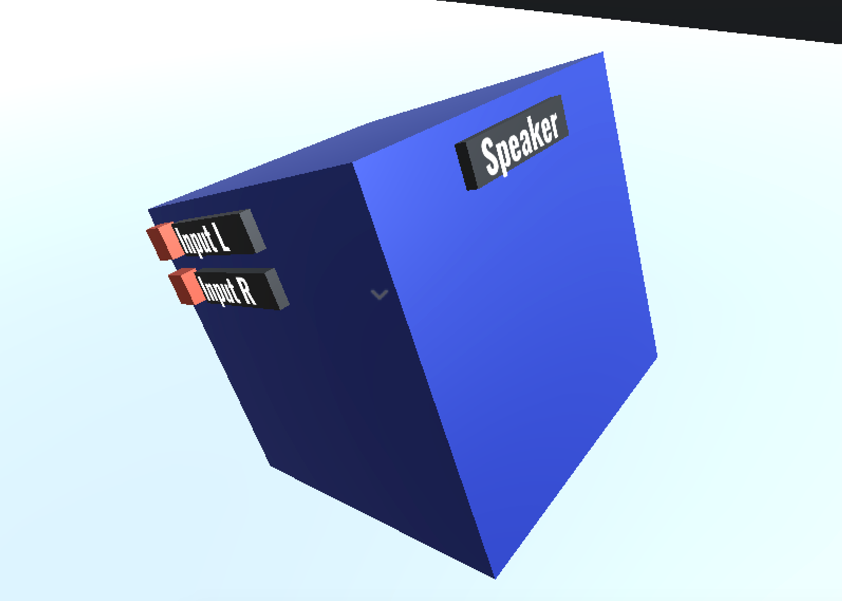
Click buttons to press them

Double click the same socket to disconnect it

For starters we’re going to move around, spawn a block then delete it.

To anyone that isn’t familiar with first person perspective games. WASD keys on your keyboard move your character and you move the mouse to look around.

This arena is looking pretty empty right now, so let’s spawn a speaker block. Look at the spawn menu on the right of the screen - Press the number shown next to the speaker key.



As you can see, it looks a lot like a speaker, but I’ve added a label to make it *really* obvious. This is the block that actually outputs the audio you send it, so you can hear what you’ve made.

Let’s try moving it around – look at any of the casing (the blue bit.) Click and hold to start moving it. It will always face the main control panel towards you while you’re moving it. (Currently the speaker has no controls so it’s just a label on the front.)

Once you’re done, put it down anywhere you’d like and take a look at face to the left of the panel with the label – notice two red blocks, those are audio sockets. Labelled ‘Input L’ and ‘Input R’ for left and right channels.

Audio inputs will always be on the panel to the left of the control panel, audio outputs will always be on the panel to the right of the control panel.

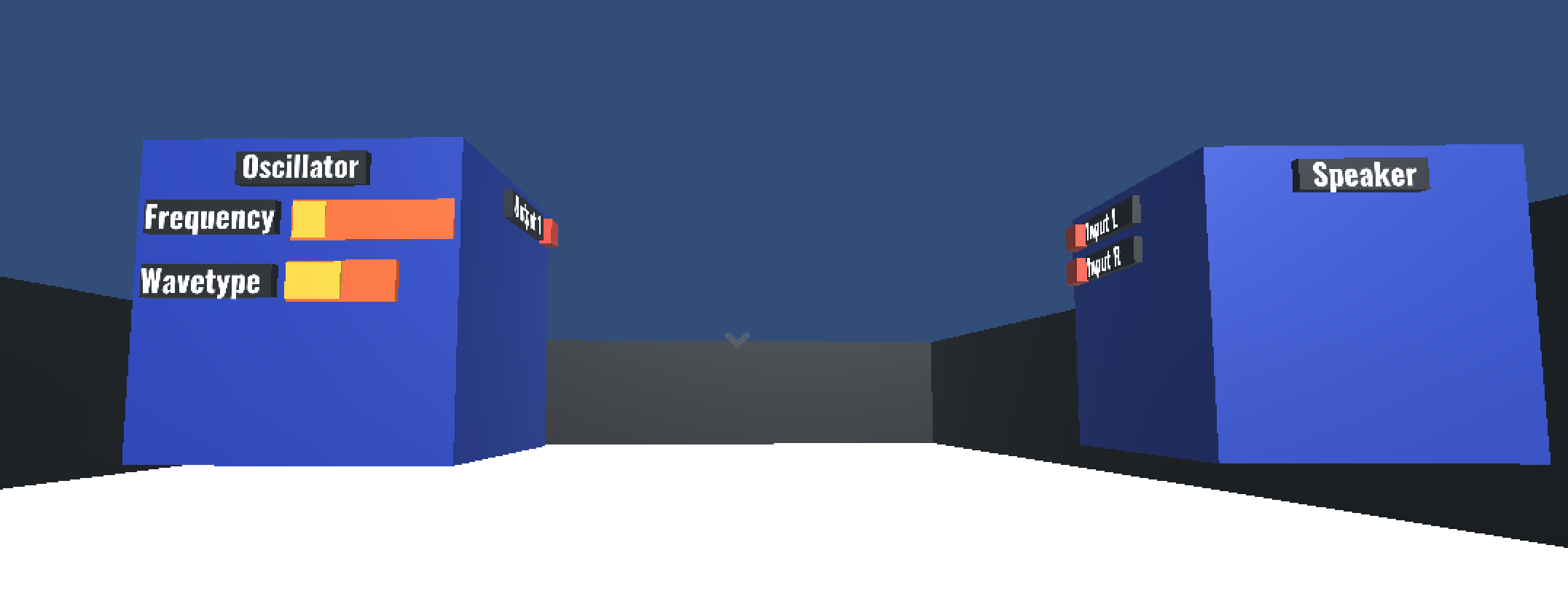
Click on one of the sockets and then move around, you’ve started creating a connection! Unfortunately there’s nothing to connect it to right now so let’s just put that back by clicking anywhere.

Now delete the block by looking at the casing of the block, hold down shift, and click. Poof, gone.

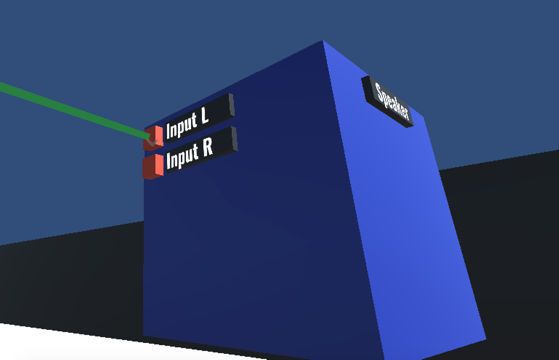
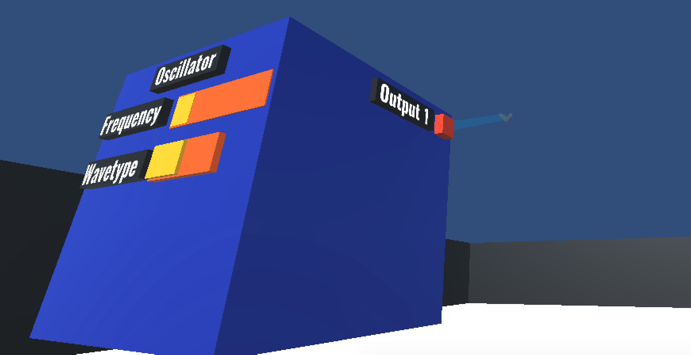
Task 2: Play a sine wave

Now we’re going to make some sound.

Spawn an Oscillator block and a speaker block.



The simplest way of making a sound is to spawn an oscillator and a speaker. Connect the oscillator’s output socket to one of the input sockets of the speaker by clicking on each of them in turn.

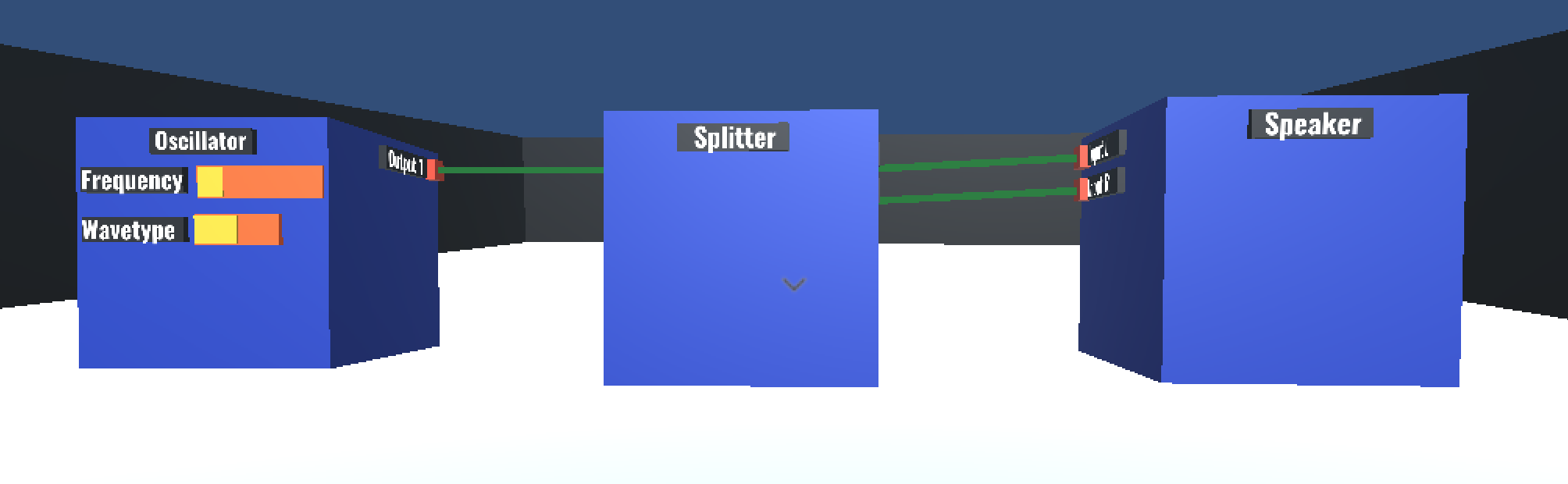


Sound! The problem is it only comes out of one speaker.

Signals are generated in mono, so to split the signal, much like in real life – you would use something to split it.

Disconnect the oscillator from the speaker by double clicking on one of the sockets you linked up.

Move them apart a little bit and spawn a splitter block between them.

Now connect the output socket of the oscillator to the input socket of the splitter. Now the splitter has two output sockets – each the same as the input signal. Connect the two output sockets of the splitter to the input sockets of the speaker and voilà – stereo. 

Task 3: Play a sequence of notes

Following on from the previous task let’s get our oscillator to do more than play a single note, because that gets annoying.

It’s important to note there are two types of socket – audio and data sockets. They are used for different things.

Audio is where the sound of each object comes out – we can either play that sound or use that sound to modulate and control other sounds in strange ways.

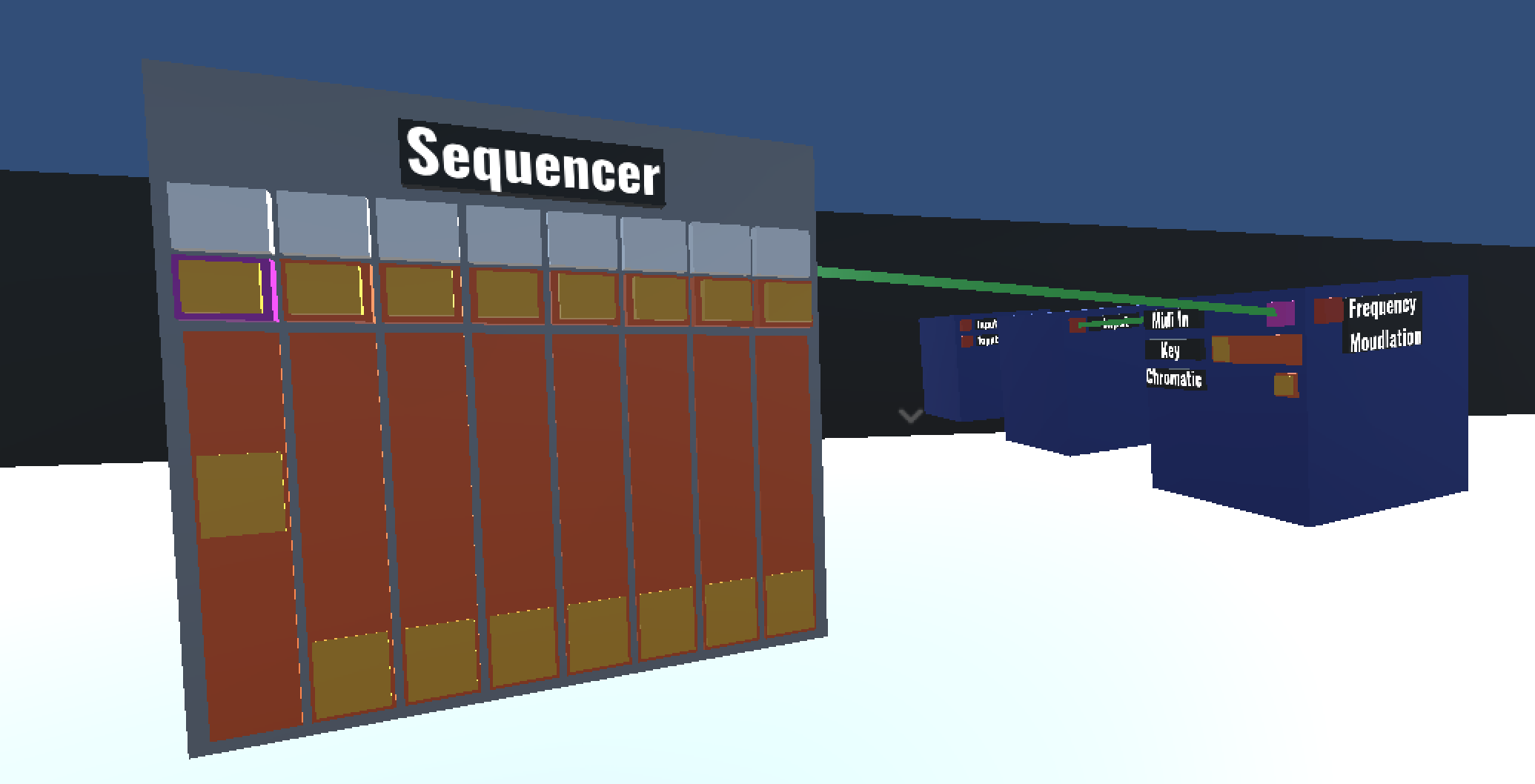
Data is much simpler – for the sequencer block the data sent will just be a single number to tell the oscillator which note to play.

Optional Reading:

Audio signals get sent in ‘buffers’ – a big group of numbers is sent all at once because it is more efficient. Data is a much smaller group of numbers, most of the time just a single number. We can’t send audio to a data socket because the block wouldn’t know what to do with the information, but don’t worry:

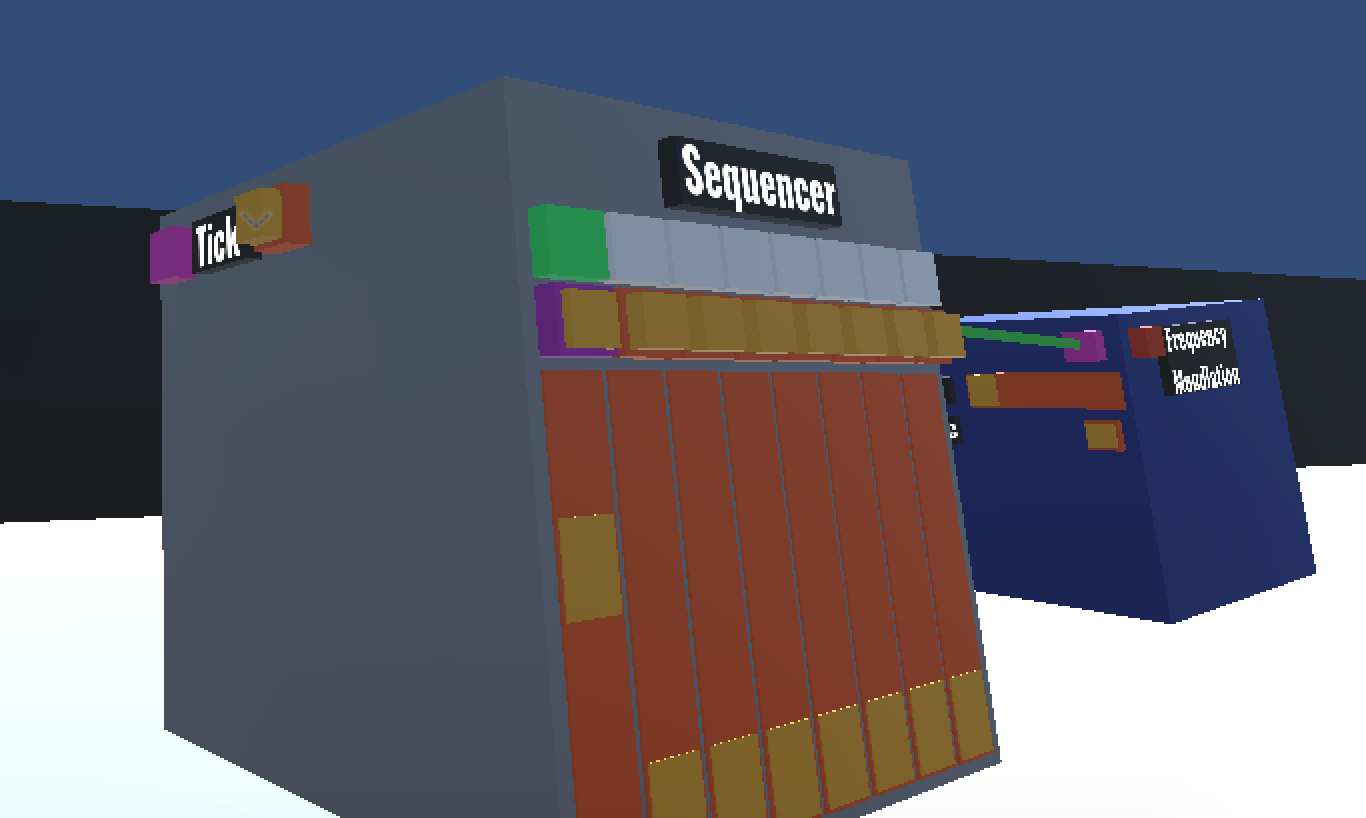
It’s not possible to join audio and data sockets together – if you’re wondering why your sockets aren’t connecting it might be that you’re trying to plug a data output into an audio input, or vice versa.

Spawn a sequencer block.

Connect the output of that to the note input on the oscillator. (The note input is on the back)

It doesn’t sound any different – but not to worry, press the button furthest to the left to enable that note, then move the slider to control its pitch.

Still nothing’s happened! That’s because the sequencer is all ready to send a note, but hasn’t been told to send the note yet. On the face to the left of the control panel, press the tick button. Voilà, your new note is playing. You can see which note is currently playing by looking at the step indicator on the control panel.



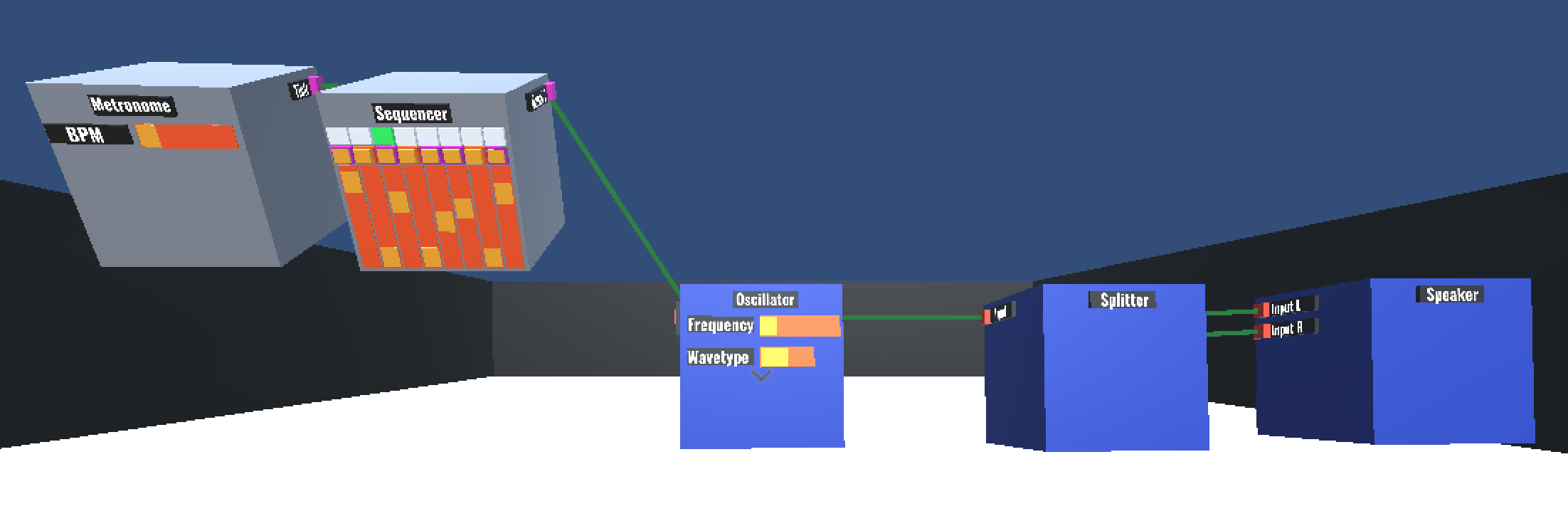
It’s going to get really annoying to press that button every time we want to play a new note – so let’s automate that.

Spawn a metronome block.

Connect the metronome block’s output to the ‘tick’ input socket on the sequencer. And slowly but surely, it will tick through each of those notes in turn.



Press some more buttons and make yourself an 8-step sequence.



You’re probably wondering what else you would ever want to do in the world now that all of your musical desires have been satisfied, but please try to focus.

Task 4: Frequency modulation

Now that you have a full and complete soon to be award winning dance track made, let’s remix it to give it a different timbre by adding a branch to the oscillator.

Spawn an LFO (Low frequency oscillator.) You could use a regular oscillator instead if you’re feeling adventurous.

Connect the output of that into the frequency modulation input socket on the oscillator you made earlier.

To the very astute, yes what a great and amazing change that was!

Let’s explain why that was so anti-climactic:

Digital audio signals are made up of lots of numbers between -1 and +1. These numbers go through a number of processes to be translated into a voltage to control a speaker cone’s position really really fast. Currently our LFO outputs numbers just like a regular digital audio signal. -1 to +1, ready for output.

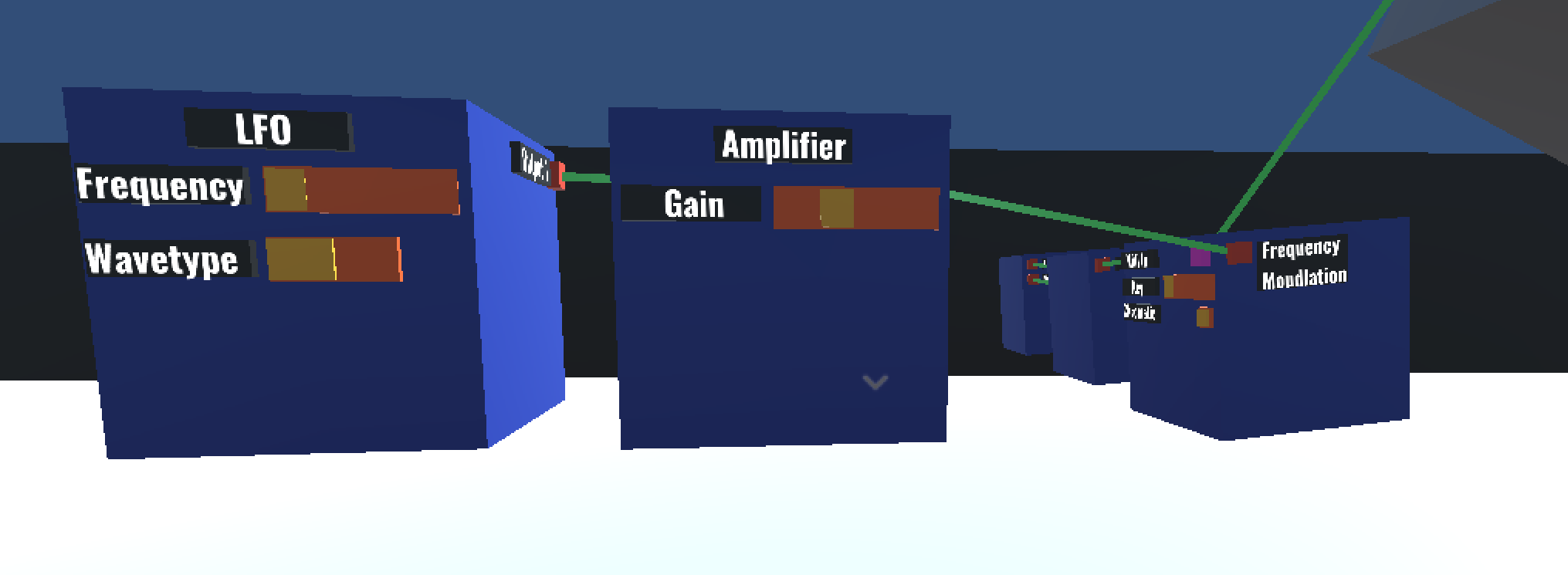
The oscillator block holds its frequency set by either the slider on the front of it, or the note input and then has the option of being modulated by an audio signal from the frequency modulation input. **When we send the oscillator block an audio signal to the frequency modulation input, it’s adding the audio signal (-1 to +1) to the frequency the block is currently playing (e.g. 440Hz)**

So, when we’re trying to modulate a frequency of say 440Hz (A4,) adding and subtracting a maximum of 1 from it isn’t really that significant and so you don’t hear that much of a difference - for reference the next note up is 466.16Hz (A#4). So let’s make that signal REALLY loud.

Disconnect the output of the LFO.

Spawn an amplifier and place that between the LFO and the Oscillator

Connect the output of the LFO to the ‘Input’ of the amplifier, then connect the output of the amplifier to the frequency modulation input socket of the oscillator.

Again! Barely any difference, now move the slider on the amplifier to the right. Voilà you have improved your top 100 hit into a top 10 hit.

Task 5: Amplitude Modulation

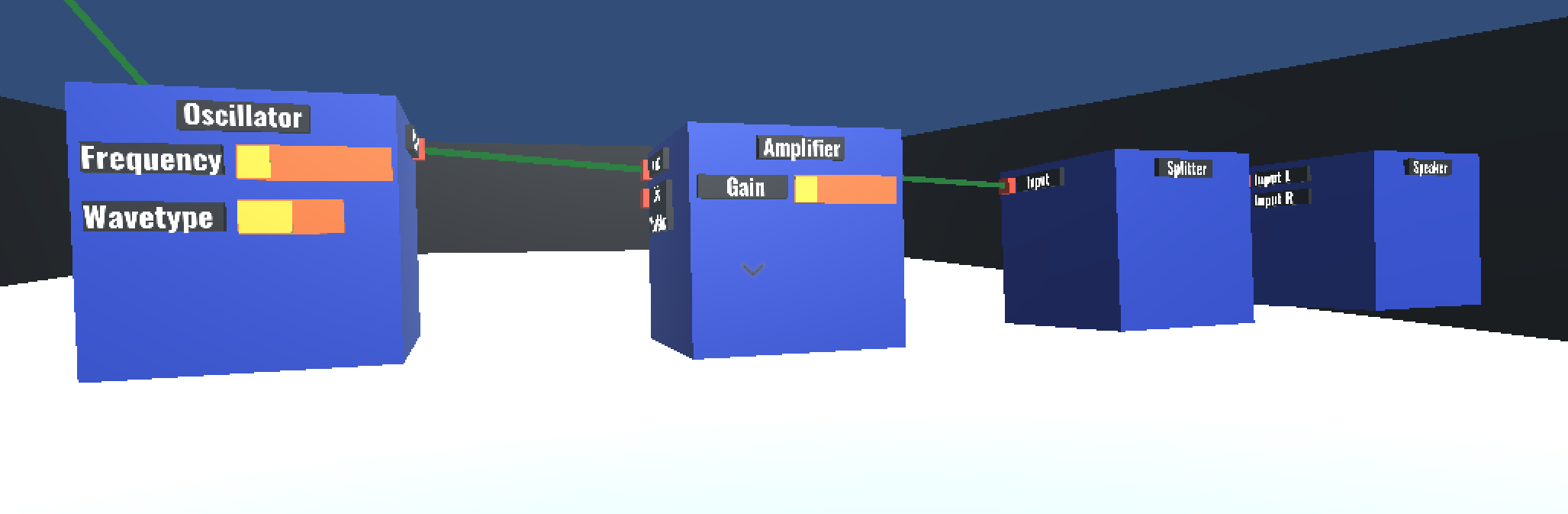
To get to that number 1 spot we’re going to need a couple more DSP (Digital Signal Processing) tricks.

\*\*If you did the previous task you might want to turn that frequency modulator you made off by pulling the slider on the amplifier all the way down to the left. Save it for the Grammys.

Let’s modulate that oscillator you made in task 2.

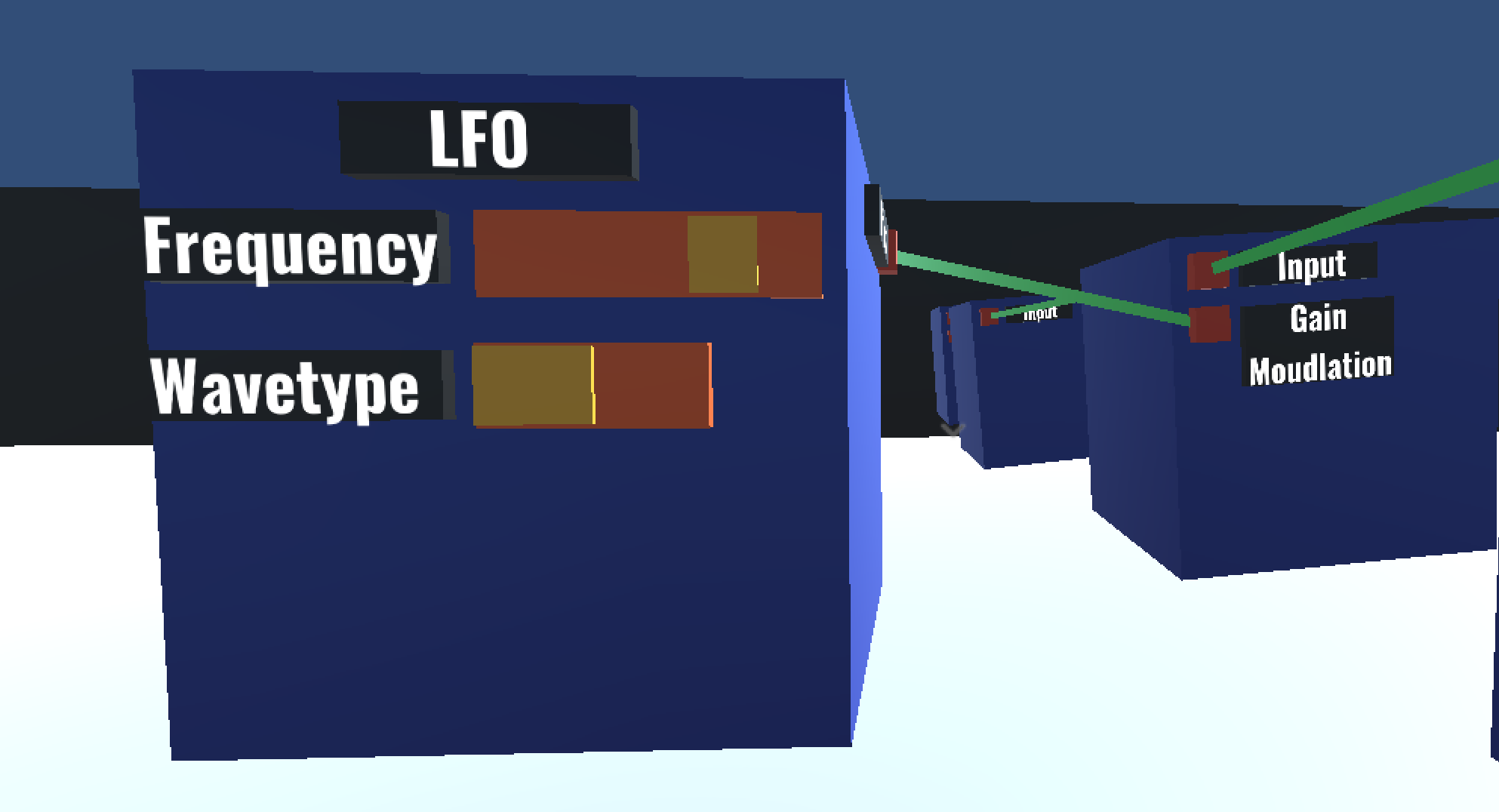
Disconnect its output (Ah isn’t that nice.)

Spawn an amplifier and make some room for it to fit in the middle.

Send the output of the oscillator into the amplifier, and the output of the amplifier into the splitter (Oh no it’s back.)

Optional:

While we’re here play around with the amplifier and hear all the clipping – that’s distortion. If you’re still using the sine wave sound, the more distortion you apply the more it’s going to sound like a square wave. If you distort a perfect square wave absolutely nothing will happen – though in most modern audio applications, synthesised square waves are anti-aliased to make them less square but that’s another topic and isn’t important for this tutorial!

Now branch out an LFO and connect that to the Gain Modulation input socket on the amplifier.

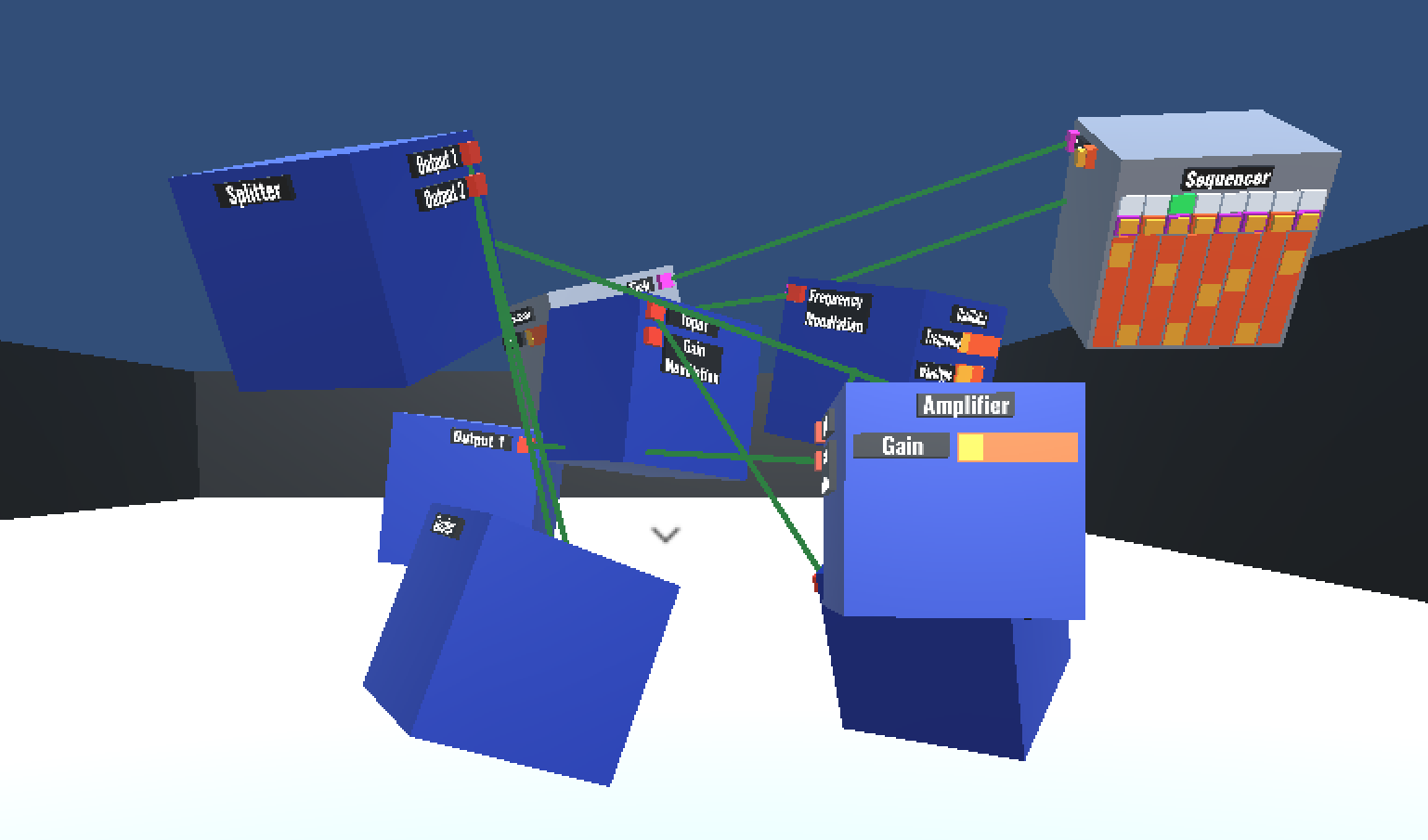
Listen to that! Your whole tune now warbles.

Though perhaps not like you’d expect:

The LFO outputs values from -1 to +1. Really, we would want that to be 0 to +1 to keep that sine wave shape and not have phasing issues, but that’s not important for now and it still sounds cool!

Task 6: Sandbox

So when you’re done with those you should have something that looks like this:



And everyone is going to think you’re really clever!

Now that you’ve gotten to grips with the basics - go experiment! There are lots of interactions that I haven’t mentioned or don’t know about. Here are some things you might want to try:

* Add another layer to go side by side with your 8-step sequence – Play around with the data splitter and mixer blocks
* Extend the 8-step sequence into a 16-step sequence - Try chaining sequencers together.
* Chain a million oscillators and LFOs together to make an absolute racket - I hope you’re not using my laptop.