[11]

Musical Instrument Design

CAMILLE GOUDESEUNE

Experience Design and Musical Instruments

Experience Design (XD) extends the field of user experience (UX), just as UX extended the older concept of usability. Usability appeared in the 1980s, applying ergonomic principles to that era's emerging technologies: word processors, spreadsheets, and others. In turn, the emerging technology of the 1990s—online shopping, SMS (texting), early camera phones, pre-smartphone PDAs—brought forth UX.

Again, today's emerging technology demands broader design, as previously separate elements combine to form new experiences, such as our society where almost everyone, from awaking to falling asleep, carries a smartphone for browsing and posting to websites for social media. Hence, the need for XD: as the experience is broader than one person using one device, so the designers of the phone's components (camera, operating system, apps, network infrastructure, even its fee schedule) must look beyond their individual specialties to construct the overall experience of the individual and the society. For example, the public was unprepared for the calamity of texting while driving. Only recently has this danger prompted one telecom company to commission a famous movie director to make a documentary about it, distributed for free; but as long as the always-online lifestyle is promoted in advertisements that are also distributed for free, such films will be taken just as seriously as the brewery billboard footnotes that advise consumers to "drink in moderation."

The point of this admittedly sensational example is that texting while driving is not a Designed eXperience. There is a need for XD. To that end, I offer an example from which XD can learn: the field of musical instruments. It clearly connects to human-device interaction. Its social

roles are well studied. Less obvious is its long history of disruptive high technology. Already three centuries ago, the interface for the pipe organ had grown to several keyboards, a pedalboard, and a dizzying array of other buttons, drawknobs, and levers, as depicted in Figure 11-1. So complex were these controls that page-turning assistants were needed to operate them. Organists were the space shuttle pilots of the day, and much effort was spent to help them get the best possible sound out of something that had been built at proportional space shuttle expense.

The design of musical instruments illuminates XD, because these devices present interfaces that are sophisticated, elaborate, refined over centuries, beautiful, and exquisitely adapted to the shape, capabilities, and senses of the human body. (To prepare for rapid change in the future, it helps to take a long view of history.) The interfaces of musical instruments also demand—and reward—tens of thousands of hours of continued use and study. One cannot say the same of handheld electronic doodads. Even if these facts alone warrant the study of musical instruments, the parallels to XD go deeper still. Let's begin with the organ.

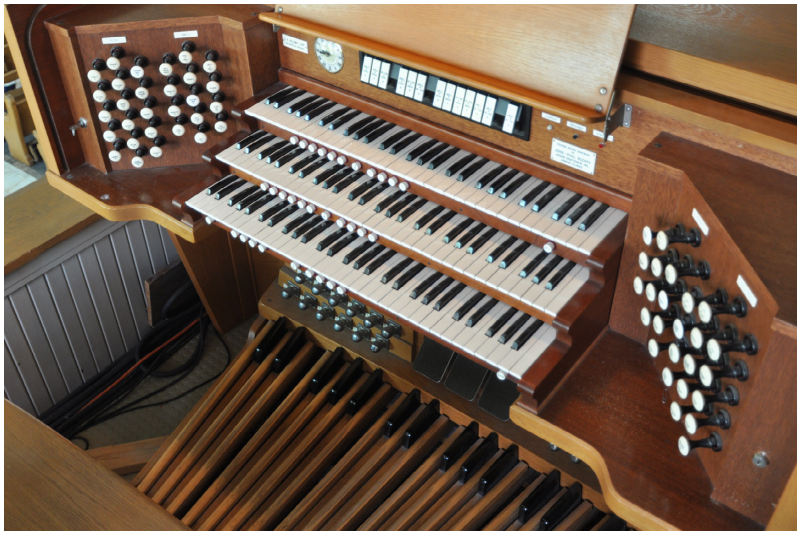

Figure 11-1. Console of the pipe organ at St. John's Catholic Chapel, Champaign, Illinois

The organ's primary feedback mechanism is not acoustic, but haptic. (The pipes can be so far from the console that organists joke about playing a fugue and then sitting back to listen to it.) Precise standards

are set for the keys and pedals: required force, traveling distance, travel point where air gets admitted to the pipes, surface friction. Physically, keyboards have not grown in width like the piano has, because the lowest and highest notes are attained through other means than mere reach. On large theatre organs, the upper keyboards tilt down, sometimes almost vertically. Cognitive issues relate to which pipes are sounded by a particular keyboard. (To the newcomer, the organ's profusion of modes may be its most terrifying aspect.) Individual drawknobs enable particular timbres, such as clarinets, flutes, or trumpets. "Mixture" drawknobs enable several timbres (ranks of pipes) simultaneously. Combination pistons set an entire collection of drawknobs with a single push, from a thumb or a toe, whichever is free at that moment during a performance. Couplers connect one keyboard's pipes to another keyboard, or sound pipes an octave higher or lower than usual. A bass coupler connects the pedalboard's pipes to the lowest note currently held on a keyboard. Most revealingly, even with recent computerization and touchscreens, organs don't try to helpfully draw a diagram of active connections. That would only be a cluttersome distraction; instead, visual feedback is limited to indicating which couplers are active. The organist memorizes how these are wired together, a task not much more complicated than memorizing the layout of a car's stick shift.

Speaking of computers, the parallels to XD grow stronger for newer musical instruments that incorporate software.

[NOTE]

Later in the chapter, we'll see how adding a tilt sensor to a pitch-tracked electric guitar can make it either easier to play or impossibly harder, depending only on the software. We'll also see the many ways that dancers can affect the music to which they're dancing by using software to interpret their movements sensed by a motion capture rig.

Even if a software-based instrument lacks the organ's combinatorial explosion of mapping keyboards to ranks of pipes, software's sheer flexibility seduces the designer into making the instrument so reconfigurable that the player has little attention left for actual playing. (In the limit, live coding replaces all traditional performance with writing software. But this chapter deals only with instruments that still

respond to real-time physical gestures.) The player's cognitive limits must be respected but also challenged: the kazoo has spawned no Mephistophelean virtuosos, no grand concertos. A reasonable challenge rewards the player with repeated levels of mastery, like a well-paced video game.

Software lets us arbitrarily connect the player's physical gestures to the resulting sounds. Properly exploiting such richness requires guidance from XD. Only then can that realm of infinite possibility be concentrated into one instrument, one that is not merely enjoyable to listen to, but also worth playing and worth mastering.

The Evolution of the Musician

Constant change, despite its related pains, has been celebrated in music for centuries. Musicians have been quick to embrace emerging technology from the printing press to assembly-line instrument manufacturing. In our own day, we have seen not musicians but rather lawyers feebly oppose technological changes in how music is recorded, distributed, and sold.

In Mozart's era, musicians specialized. Composers wrote, cellists performed, craftsmen chiseled, princes commissioned, aristocrats in overstuffed armchairs listened. Certainly there was crossover: in his time Bach's only fame was as an improvising organist; Mahler's compositional skill came from many years of orchestral conducting; and even Frederick the Great was respected as both a composer and a flutist. But within a particular evening's entertainment, these roles were clear and distinct.

But, we no longer live in this "common practice period." Since then, musical language has fragmented into thousands of genres. Musical technology, too, has fragmented into a market of software and hardware designed for narrow niches. This would appear to demand even more specialization than what musicians accepted 250 years ago, but in fact the opposite is true. One can no longer be just a composer, just a performer, just an instrument designer. Mozart could simply complain about how a tanner had improperly cured the leather on the hammers of his fortepiano. But these days, he himself would have to dig through submenus to tweak the hammers' oil and tannin content. The common

practice period's model of the social interactions of music-making no longer fits. By now, the word musician has grown to encompass many more activities:

- Designing a musical instrument, learning to play it, and then learning to compose for it (or the reverse, or back and forth)
- Repackaging recordings, from humble playlists and mix-tape compilations to elaborately crafted seamless multihour DJ shows
- Moving living room furniture and tweaking tone controls to adjust reverberation and find the sweet spot

These days, it has become difficult to reconstruct who made what decisions that led to the final acoustic result. In short, today's musician does many things, not just one thing. We see this in how popular music has adopted technology. Two early examples are the prog-rock guitarist and stompbox wizard, Adrian Belew, and his inspiration Jimi Hendrix; since then, the vocoder/synthesizer duo Daft Punk has hardly restored the eighteenth-century demarcations.

Specialization now moves in a different direction. Instead of improving one instrument, say, a violin that can play anything from polkas to Pachelbel, the musician explores one subgenre, perhaps as narrow as Electro House Moombahcore. This exploration includes all aspects, from instrument design to performance. Having all aspects in a single person or a small group happily compresses the feedback loops that optimize the instrument for that genre. On the other hand, this rapid optimization might explain why genres come and go as quickly as Lower East Side restaurants.

Nowhere is this constant change and recasting of specialization more evident than in instrument design.

THE NEW INSTRUMENT

In our era of ubiquitous music, why are musical instruments still intriguing? Hikers and kayakers lug them along. Airlines make special exceptions for these fragile devices. Even NASA spent almost six figures to include an otherwise unremarkable guitar in a cargo shipment to the International Space Station.

The difference between an iPod and an instrument is not output—sound—but rather input. If playing an instrument is like driving a race car, then using an iPod is like being chief of a pit crew. The crew

chief makes decisions at most every few minutes, radioing commands to drive more aggressively or to stop for fresh tires. The driver, on the other hand, continuously makes precise muscular movements, notes their effect, corrects for subtle misjudgments, and makes snap decisions based on thousands of hours of direct experience. That kind of enjoyable activity is why people schlep musical instruments to campsites and space stations—and perhaps why few five-year-olds dream about growing up to become a crew chief.

So a musical instrument is something that turns your physical gestures (more than just occasional button-pushing) into sounds. How much you enjoy the act of playing, in the moment, is tied to how intimately and immediately those gestures are connected to the sounds—consider a toddler with a wooden spoon and a few pots and pans. How much you enjoy *mastering* the act of playing, over thousands of hours, is tied to how sensitive that connection is, how much expressive variety of sound you can get. Kitchen percussion quickly reaches its limit; but as a jazz set moves from ballads to bebop, a graybeard saxophonist can coax whispers, groans, flurries, or howls from his horn. It's hard to say who enjoys this more, the jazzman or the audience. Still, the audience clearly enjoys not just the acoustic result, but also the fantasy of being so expressive themselves. This vicarious thrill is found in motorsport, too, where pit overviews are broadcast less often than onboard views.

But since the days of Adolphe Sax, technology has hardly stood still. The critical connection from gesture to sound can now be mediated by something even more byzantine than valvework: the hyper-flexible material called software. The number of ways to connect inputs to outputs now dwarfs even the billions of combinations in Rubik's Cube advertisements. The question becomes how many of these mappings are worthwhile? (As a mathematical analogy, consider how few images are interesting to humans, of all the possible collections of pixels that could be displayed on a computer screen.)

As an example of this input-to-output flexibility, consider a simple drum. It's louder when you hit it harder, and sounds slightly different when you hit it in different places or with a different kind of stick. But what about an electronic drum, a pressure-sensitive pad connected to software connected to a loudspeaker? That could play louder when you hit it softer; or, different parts of the drum head could sound quite different, like a set of pitched tom-toms; or, softer sticks (mallets or

brushes) could delay the sound's onset; or, continued pressure after impact could sustain and modulate the sound, turning a percussion instrument into a melodic one—or, all of the above at once. In a few sentences, software has just let us design a rich instrument using a Stone Age interface. Later on, we'll see some more intricate examples.

THE CURSE OF FLEXIBILITY

The flexibility introduced to musical instruments by software has a dark side. When the pace of high technology meant better varnish than 20 years earlier, a lutenist could reasonably expect that his skills would transfer from one lute to another; that his investment of hours of practice was secure; that, no matter how skilled he became, he could find good teachers; and that he could find a steady supply of freshly composed music to play.

Every one of those expectations is destroyed by software.

- A software-based instrument is often a unique device. Even worse, as obsolete technologies in it are upgraded, it can change irreversibly.
- Developing playing skill on an instrument with an installed customer base of one, and with an expected lifetime of a few months, is a poor investment.
- Performers must be their own teachers.
- Of the few composers who even hear of the instrument (before it changes yet again), even fewer will be motivated to write for an instrument with so few performers and so little chance to develop their own compositional craft. The story is told that Haydn, at the time having completed almost a hundred symphonies, confessed that he was finally getting the hang of writing for woodwinds.

These problems are milder, but not absent, outside music. The faster the pace of technology, the less patience its consumers have for reading instructions. Why master every aspect of your mobile phone, when you expect to replace it within a year? Worse yet, how can you master it when everyone around you also quickly discards it, preventing good teachers from appearing?

Imagine reprogramming your car to make the brake pedal adjust windshield wiper delay, and brake when you blinked your left eyelid in Morse code. Your friends wouldn't dare to drive it. You yourself *could* learn to drive it, but mastering that skill would be expensive.

EMERGING TECHNOLOGY: COMPUTATION, SENSORS, AND AUDIO SYNTHESIS

Recall the music technology of 50 years ago, during the British Invasion of 1964. The Beatles, the Who, and the Rolling Stones began to use feedback from electric guitar amplifiers. In academia, mainframes took hours to compute one minute of acoustic signal, to be performed by a tape recorder. Going beyond tape to so-called live electronics, Karlheinz Stockhausen had just begun to experiment with microphones, sine wave generators, and ring modulators.

But nowadays, laptop orchestras are everywhere. If you extend Moore's Law from Stockhausen's four-diode ring modulator through today's billion-transistor laptops, 50 years from now an everyday musical instrument should have more transistors than the entire planet had 5 years ago. An instrument more powerful than all Big Four music labels combined could create an entire genre of music as easily as a Casiotone creates a bleep. (Granted, this overlooks implementation details such as power consumption and heat dissipation. But it will be several decades before such hardware is even invented. These technicalities will be solved.)

What about sensors? They have not advanced as startlingly as the microprocessor. Indeed, *what* they sense—position, pressure, light, EEGs—has hardly grown in half a century. But *how* they sense has advanced, in size, cost, power consumption, and speed. Smartphones, where every milliwatt and every cubic millimeter counts, include sensors for temperature, barometric pressure, humidity, tilt, and of course GPS position. Compare that to the 1964 predecessor of GPS, TRANSIT, which was too heavy for a man to lift, took thousands of times longer to report your location, and was 40 times less accurate.

The sophistication of sensors has also advanced. For example, some image sensor chips in mobile phones report when they detect a smile. (This is not merely software: this is in the chip itself.) Also, combining colocated measurements, called sensor fusion, yields what Stockhausen would have called magic but what we call commonplace: a photograph

that is not merely geotagged but also tagged by content such as email addresses of human faces, or websites of storefronts and visible landmarks. Wilder magic happens when the sensors are online, such as pointing a smartphone's image sensor at a barcode on a supermarket shelf to learn the item's price in nearby stores.

Sensor fusion can also increase noise immunity and decrease latency. For instance, when measuring the pitch of a plucked string, we can fuse conventional pitch-tracking software with a sensor that measures where the string contacts the fingerboard. At low pitches, the software by itself is too slow as it waits for an entire waveform or two. But that's exactly when the contact sensor is fast and precise.

Sensor fusion can also increase sensitivity. Eulerian video magnification amplifies a video signal's otherwise invisible variations of color or motion, such as respiratory motion, the reddening of skin with each heartbeat, or even (as before) the vibration of a guitar string. Fusing a dozen video cameras yields a motion-capture system that tracks the positions of hundreds of points with submillimeter accuracy throughout a large room, thousands of times per second. Fusing several microphones or radiotelescopes into a beamforming array gives them instant, precise aiming. Finally, combining a microphone with clever software yields a sensor for speech—what we usually call speech recognition.

Fifty years hence, we can imagine sensors that are ubiquitous and practically uncountable; cognoscenti call this utility fog. In today's language, a safe generalization is that you will measure anything you can name, as accurately as you want, as fast as you want, under any conditions.

As far as audio synthesis algorithms go, much of the history of computer music consists of clever tricks to extract ever more interesting sounds from only a few—or a few million—transistors: tricks such as filtered broadband noise, frequency modulation, or plucked-string simulation. But such optimizations are pointless when you have a brain the size of a planet. Brute-force additive synthesis of individual sine waves is easy. So is brute-force simulation of a plucked string, all the way down to the molecular bonds that determine the plectrum's stiffness and the string's inertia.

When we summarize all this, the language becomes theological. We get a musical instrument that (within its domain) is omniscient, omnicognizant, and omnipotent. It observes all that can be observed, analyzes these observations completely, and from those conclusions then produces whatever sound is optimal for a particular purpose.

What this means for musical instruments is hard enough to assimilate and ponder. But what such prodigious sensing and computation means for human culture, no one can predict in detail: uploaded minds, computronium (converting all matter into computers), the blurring of human and machine (palely foreshadowed by mobile social media), and immortality. These are aspects of what some call the *Singularity*, the point in history when change becomes so rapid that language before then cannot even describe it. How we then shall make, share, understand, and enjoy music must remain a mystery.

Still, this undoubtedly highfalutin' talk informs the XD of today. Occasionally taking the long view, either the half century since the British Invasion or the eons of the pipe organ, escapes the rut of the past week's RSS feeds, the past year's product launches. When confronted with the Accordion of Armageddon, even the most far-out creatives must concede that their imaginations could be wilder.

Now, let's rewind those 50 years, to consider nuts-and-bolts details of some unobtanium-free designs that require only a few billion transistors.

DESIGNING FOR SOFTWARE INSTRUMENTS: FROM GESTURES, THROUGH MAPPING, TO SOUND

When designing a software-based musical instrument, either from scratch or by extending a familiar instrument, choosing its inputs and outputs is relatively easy. The instrument's inputs are buttons, knobs, tilt sensors, cameras, or even whichever of these is found in a smartphone. Its outputs might just be dictated by what commands can be sent to the instument's audio synthesizer, be that a chip or software. Common outputs are pitch (how high a sound is) and loudness. Other aspects of timbre can come from a set of discrete presets, such as the trumpet and harpsichord buttons on a department store keyboard.

At this point, after choosing inputs and outputs, the real work of XD begins. To see what a difference is made by the input-to-output mapping, let's consider three real-world examples that use the same gesture-inputs and sound-outputs, varying only the mapping.

1. Conventional pickup-and-amplifier instrument, such as an electric guitar or electric violin, plus a tilt sensor. (Duct-tape a smartphone to the instrument.) Feed the pickup and the tilt sensor into a computer (perhaps that same smartphone), which computes sound to send to the amplifier.

   Inputs: tilt, pitch, and loudness.

   Outputs: pitch and loudness.

   Unless otherwise specified, pitch maps to pitch, and loudness to loudness.

Rock star

High notes are dramatic in everything from Van Halen to Wagner. To make them easier to play while maintaining drama, when the instrument points up, raise the output pitch by an octave or two.

Controllable Auto-Tune

More tilt applies stronger pitch correction, so you can rely on this crutch only in difficult passages.

Brain melt

Ignore tilt, but map pitch to loudness, and loudness to pitch. (Think about that for a moment.) The language that experienced players use to describe this is unprintable.

Brain evaporate

Tilt crossfades between brain melt and conventional pitch-to-pitch, loudness-to-loudness. (Don't even try to think about this one.)

The first two mappings make the instrument easier to play. The last two make it disastrously difficult, but not artistically pointless: the equally obstreperous programming language Brainfuck has inspired surprisingly many publications, by art theorists as well as computer scientists. So, mapping affects at least ease of use. Let's see what else it can affect.

2. Pressure-sensitive tablet computer, scrubbing through an audio recording.

   Inputs: pressure and x-y position of the fingertip on the tablet's surface.

   Secondary inputs, computed from the primary inputs: speed of fingertip, and duration (so far) of the current stroke.

   Outputs: index into recording (position along the audiotape segment); filter parameters (wah-wah); other effects processing.

Scrub

Map x to index, pressure to loudness, and y to a filter sweep. The x-mapping works like Laurie Anderson's tape-bow violin.

Quiet scrub

Also map tip speed to reciprocal loudness, so faster scrubs are quieter. This emulates how, in a movie, we see a whip pan as being out of focus.

Wah-wah pedal

Also map stroke duration to filter sweep, so each stroke sounds like a "wah."

Holding pattern

Map tip speed to index, and ignore all other inputs. Thus, when the tip circles steadily, you hear one fragment of the recording. When the tip speeds up, scrubbing moves forwards in the recording. When it slows down, scrubbing rewinds.

These last two mappings use secondary inputs. They demonstrate the antics that become possible when you use not just an input's raw value, but also that value's history and how fast that value is changing. The formal name for this value-history-change triplet is proportional, integral, and derivative (PID) control. (This is a fundamental mathematical way of connecting inputs to outputs, such as sensors adjusting a car engine to keep it running smoothly, or accelerometers in a quadcopter adjusting rotor speeds to compensate for wind gusts.) The point here is that a mapping need not be moment to moment, where this input value always yields that output value. Instead, the mapping might determine the output from

the *trajectory* of input values. A similar trajectory-based mapping tool is hysteresis, which behaves like gearwheel backlash or the slop in the middle of a joystick's range of motion.

Now that we've seen both playability and input-value trajectories, let's consider how literal a mapping should be.

3. Room-size optical motion capture, playing only the black keys of five stops of a pipe organ. (Although this looks like a connected environment or smart room, it still behaves like a musical instrument.)

   Inputs: x-y-z positions of a few dozen markers on the costumes of dancers (see Figure 11-2).

   Secondary inputs: average and spread (mean and standard deviation) of x, y, and z individually.

   Outputs: pitch average, pitch spread, loudness of each organ stop.

Crossfade

Map average z (height) to overall loudness. Map x to pitch, in both average and spread. Map average y to a crossfade through the organ stops in a fixed sequence. The audience immediately notices that when everyone is near the floor, it gets quiet; many raised arms make it loud. Next, they see that walking from left to right (x) is like moving up the organ's keyboard. Finally, they notice the upstage to downstage crossfade.

Zones

Within the danceable x-y-z volume, define five subvolumes, possibly overlapping. Map the number of markers in each zone to the loudness of the corresponding organ stop. Map x to pitch as before.

Spread-average swap

Map spread of y to organ-stop crossfade. Map average x to spread of pitch, and spread of x to average pitch. Map z to loudness as before. (Ignore average y, to use as pure dance with no musical consequences.) Now the audience still detects a strong cause-and-effect, still feels that the dancers directly affect the music. But the audience isn't quite sure how. Not many could verbalize what happens on stage: low pitches when everyone's tightly clumped left-right, high when they're spread out; different stops depending on upstage-downstage clumping; single pitches at stage left, broad clusters at stage right.

Figure 11-2. Motion-tracked retroreflective balls, worn by a few dancers or many dancers, can be the input gestures for a musical instrument (top: University of Illinois dance faculty Kirstie Simson and Philip Johnston experimenting in the laboratory; bottom: students improvising during a public performance)

In an evening's performance of several dances, a simple mapping such as crossfade works well early in the program, to ensure that everyone in the audience comprehends that the dancers directly control the music. But mickey-mousing won't stay captivating all night, so it's good to finish with less literal mappings. Such a development of mappings, a progression from what is instantly comprehensible to what can be savored longer, also applies outside music and dance. Right after a hard day's work a pilsner quickly slakes your thirst, but later in the evening it's nicer to tarry over an aged port. The holy grail of an intuitive interface is better for a 20-second experience (reclining the driver's seat) than for a 20-hour one (repainting the car). The nonintuitive stick shift may soon be preferred more by videogamers than by commuters. When designing an experience for a specialist, be they seasoned concertgoer, gourmet, car restorer, or videogamer, the experience's very duration justifies some up-front training cost, that is, conscious reasoning (the antonym of intuition).

ASPECTS OF MAPPINGS

Designing a mapping that ends with pitch and loudness is, however, incomplete. That's like designing a cruise control that presses the gas pedal so many inches and leaves it at that. A better mapping goes beyond the convenient abstractions of pitch and loudness, all the way to what enters the ears of the musician (and the audience).

How, then, do we measure sound? A cruise control measures only one number, miles per hour; but describing the full gamut of an instrument's sounds can require many more dimensions. However, we can take a shortcut. Because we finish designing the mapping before the instrument is deployed in performance, we need not actually measure billions of individual sounds; it suffices to measure the *difference* between two sounds. A quick and dirty way to do this is, for both sounds, to measure the loudness in each psychoacoustic critical band, then build a vector of those loudnesses, and then apply to those two vectors a $p = 5$ Minkowski distance metric. (Don't worry about those details. The point is that acousticians really do consider this to be quick and dirty, rather than rocket science. Meaningfully measuring the difference between two arbitrary sounds is indeed possible.)

Then, from a large set of sounds, we know which ones are near each other and which are far apart; that really helps to design a mapping that makes the instrument feel perceptually uniform. The variation of

sound is spread smoothly over the range of input values, without any startling areas where the sound suddenly jumps. By analogy, say you want to place expensive gas stations, not too far apart, but without wasting several stations close together. Given a collection of candidate sites, you don't need a map; you need only a table of distances between sites. This table is just what acoustic difference measurements give you, for a map that is undrawable because it has far more dimensions than the two of latitude and longitude.

Even with a mapping that goes all the way to perceived sound, still more designing is needed to complete a musical instrument. After all, its design considers not just the listener's experience but also that of the performer. Performers desire three things in particular:

- Consistency. The same input gesture should produce the same output sound.
- Continuity. Changing a gesture slightly should change the sound slightly.
- Coherence. The direction that the sound changes should not be astonishing.

Consistency

This is essential for learning and mastery. Elaborations like PID control violate consistency, but only in spirit: there, the same result comes from the same input trajectory. For example, on a racetrack, the same timing of braking and steering commands results in the same lap time. The driver can master the car, even though yanking the wheel hard has a different effect when he's been doing 20 mph than when he's been doing 90.

Continuity

Another essential for mastery, but this one is on a time scale of milliseconds rather than months. Even the best singers overshoot or undershoot the change of pitch from one note to the next, and then lock into their target after a few moments by modulating vocal cord tension, diaphragm pressure, and many minor muscles. Without continuity, that lock-in would suffer glitches and hiccups that would sound like a new driver missing a shift from first gear into second. (Early singing-voice synthesizers, which understandably omitted deliberate overshoot, sounded like spooky robots.) The discontinuity in vocal range called

the passaggio takes singers years of study to master. Back in the car, the transitions between dirt and asphalt in rallycross racing are particularly challenging, as the tires' grip suddenly changes, affecting all of the driver's commands.

Coherence

Just like consistency is refined by continuity, continuity is further refined by coherence. Drummers protest when a harder whack makes a quieter sound. This example generalizes: more physical energy in means more energy out, of either the acoustic or the aesthetic kind. It's often obvious which direction of change is proper, but what may be unnoticed is that there is a choice of direction to be made.

Stepping away from mapping for a moment, let's look at just how the sound can change. More than just two or three aspects should change. For example, when pitch changes all by itself, you get the cheesy pitch-bend sound of early 1980s synth pop (Spyro Gyra). When other things change with pitch but in only one way, along only one continuum, you get the milder tasting but still eventually tiresome sampler sound of early 1990s synth pop (Pet Shop Boys). So-called one-to-many mappings are prone to producing such fatigue. Those can't compete with a real violin, whose bow can, from moment to moment, move fast or slow (loud or soft), press firmly or lightly (crunchy or breathy), and be near or far from the bridge (glassy or deep).

Changing many things independently suggests a design that uses many inputs, but this has limits. When performers can no longer attend to all the instrument's inputs, they risk falling back into thinking about what to operate, rather than what sound to produce. The graybeard jazzman's state of flow then collapses into stumbling beginner-think. No general design rule can say where the happy medium is. Instead of a strict rule for all possible designs, we follow a guide: try two designs, one deliberately cheesy, one deliberately overwhelming. If users confirm that of these two, one really is too cheesy and one too hard, then split the difference and repeat. I'm tempted to call this the Goldilocks Guide, but decades ago programmers dubbed it Binary Search and proved that very few iterations are needed before it reaches the happy medium.

Conclusion

Musical instruments are hardly the only things that now use software. Toothbrushes, toasters, and toilets can now sprout a dozen buttons and blinking lights. It's no surprise that the designer, daily immersed in state diagrams, flowcharts, and circuit layouts, might eventually surmise that what needs its teeth cleaned is just another computer. The same point is made by the story told of John XXIII, who rejected an architect's blueprint for the papal apartments with a scribbled *Non sunt angeli* (we're not angels): there were no bathrooms.

However, unlike most consumer appliances, musical instruments *demand* quality XD. An instrument with poor XD is simply abandoned: it's socially more acceptable to not play the oboe than to not brush your teeth. Because of this, instruments make good models for XD. Here are some examples of these musical concepts in extramusical contexts.

- Videogames from the 8-bit era imitated a pressure-sensitive gas pedal by augmenting a simple on-off switch with PID control. The longer you held the button, the faster you went; and when you let go, you slowed only gradually.
- Handheld devices repurpose four on-off switches as a scrollwheel, a secondary input. Tokyoflash's wristwatches have raised to an art form both these scrollwheels and riotously nonintuitive mappings from hh:mm:ss to pixels.
- When analyzing mouse behavior in a viewer of 3D worlds, an end-to-end mapping considers how mouse-pawing and scrollwheel-flicking might unnaturally stutter a rotating gaze or a highlight moving down a menu.
- Consistency, continuity, coherence. Online games that charge monthly fees have excelled for a decade at giving users a sense of mastery with little sense of astonishment, at time scales from milliseconds to months.

Of course, trying to enumerate every application of these principles would produce a list that was obsolete before it was finished. The best summary for what musical instruments have to offer your own XD is pure metaphor. Imagine someone brushing their teeth or toasting bread as emotionally and expressively as rocking out with a guitar.

Happy designing!