Chapter 1. How Humans Interact with Computers

Timoni West

In this chapter, we explore the following:

- Background on the history of human–computer modalities
- A description of common modalities and their pros and cons
- The cycles of feedback between humans and computers
- Mapping modalities to current industry inputs
- A holistic view of the feedback cycle of good immersive design

Common Term Definition

I use the following terms in these specific ways that assume a human-perceivable element:

Modality
: A channel of sensory input and output between a computer and a human

Affordances
: Attributes or characteristics of an object that define that object's potential uses

Inputs
: How you do those things; the data sent to the computer

Outputs
: A perceivable reaction to an event; the data sent from the computer

Feedback
: A type of output; a confirmation that what you did was noticed and acted on by the other party

Introduction

In the game Twenty Questions, your goal is to guess what object another person is thinking of. You can ask anything you want, and the other person must answer truthfully; the catch is that they answer questions using only one of two options: yes or no.

Through a series of happenstance and interpolation, the way we communicate with conventional computers is very similar to Twenty Questions. Computers speak in binary, ones and zeroes, but humans do not. Computers have no inherent sense of the world or, indeed, anything outside of either the binary—or, in the case of quantum computers, probabilities.

Because of this, we communicate everything to computers, from concepts to inputs, through increasing levels of human-friendly abstraction that cover up the basic communication layer: ones and zeroes, or yes and no.

Thus, much of the work of computing today is determining how to get humans to easily and simply explain increasingly complex ideas to computers. In turn, humans are also working toward having computers process those ideas more quickly by building those abstraction layers on top of the ones and zeroes. It is a cycle of input and output, affordances and feedback, across modalities. The abstraction layers can take many forms: the metaphors of a graphical user interface, the spoken words of natural language processing (NLP), the object recognition of computer vision, and, most simply and commonly, the everyday inputs of keyboard and pointer, which most humans use to interact with computers on a daily basis.

Modalities Through the Ages: Pre-Twentieth Century

To begin, let's briefly discuss how humans have traditionally given instructions to machines. The earliest proto-computing machines, programmable weaving looms, famously "read" punch cards. Joseph Jacquard created what was, in effect, one of the first pieces of true mechanical art, a portrait of himself, using punch cards in 1839

(Figure 1-1). Around the same time in Russia, Semyon Korsakov had realized that punch cards could be used to store and compare datasets.

Figure 1-1. Woven silk portrait of Joseph Jacquard, 1839, who used more than 24,000 punched cards to create the portrait

Punch cards can hold significant amounts of data, as long as the data is consistent enough to be read by a machine. And although pens and similar handheld tools are fantastic for specific tasks, allowing humans to quickly express information, the average human forearm and finger tendons lack the ability to consistently produce near identical forms *all* the time.

This has long been a known problem. In fact, from the seventeenth century—that is, as soon as the technology was available—people began to make keyboards. People invented and reinvented keyboards for all sorts of reasons; for example, to work against counterfeiting, helping a blind sister, and better books. Having a supportive plane against which to rest the hands and wrists allowed for inconsistent movement to yield consistent results that are impossible to achieve with the pen.

As mentioned earlier, proto-computers had an equally compelling motivation: computers need very consistent physical data, and it's uncomfortable for humans to make consistent data. So, even though it might seem surprising in retrospect, by the early 1800s, punch-card machines, not yet the calculation monsters they would become, already had keyboards attached to them, as depicted in Figure 1-2.

Figure 1-2. A Masson Mills WTM 10 Jacquard Card Cutter, 1783, which were used to create the punched cards read by a Jacquard loom

Keyboards have been attached to computational devices since the beginning, but, of course, they expanded out to typewriters before looping back again as the two technologies merged. The impetuous was similarly tied to consistency and human fatigue. From Wikipedia:

By the mid-19th century, the increasing pace of business communication had created a need for mechanization of the writing process. Stenographers and telegraphers could take down information at rates up to 130 words per minute.

Writing with a pen, in contrast, gets you only about 30 words per minute: button presses were undeniably the better alphanumeric solution.

The next century was spent trying to perfect the basic concept. Later features, like the addition of the shift key, substantially improved and streamlined the design and size of early typewriters.

I want to pause for a moment here to point out the broader problem everyone was trying to solve by using typewriters, and specifically with the keyboard as input: at the highest level, people wanted to capture their ideas more *quickly* and more *accurately*. Remember this; it is a consistent theme across all modality improvements.

Modalities Through the Ages: Through World War II

So much for keyboards, which, as I just pointed out, have been with us since the beginning of humans attempting to communicate with their machines. From the early twentieth century on—that is, again, as soon as metalwork and manufacturing techniques supported it—we gave machines a way to communicate back, to have a dialogue with their operators *before* the expensive physical output stage: monitors and displays, a field that benefited from significant research and resources through the wartime eras via military budgets.

The first computer displays didn't show words: early computer panels had small light bulbs that would switch on and off to reflect specific states, allowing engineers to monitor the computer's status—and leading to the use of the word "monitor." During WWII, military agencies used cathode-ray tube (CRT) screens for radar scopes, and soon after the war, CRTs began their life as vector, and later text, computing displays for groups like SAGE and the Royal Navy.

Figure 1-3. An example of early computer interfaces for proprioceptive remapping; WAAF radar operator Denise Miley is plotting aircraft in the Receiver Room at Bawdsey "Chain Home" station in May 1945 (notice the large knob to her left, a goniometer control that allowed Miley to change the sensitivity of the radio direction finders)

As soon as computing and monitoring machines had displays, we had display-specific input to go alongside them. Joysticks were invented for

aircraft, but their use for remote aircraft piloting was patented in the United States in 1926. This demonstrates a curious quirk of human physiology: we are able to instinctively remap *proprioception*—our sense of the orientation and placement of our bodies—to new volumes and plane angles (see Figure 1-3). If we weren't able to do so, it would be impossible to use a mouse on a desktop on the Z-plane to move the mouse anchor on the X. And yet, we can do it almost without thought—although some of us might need to invert the axis rotation to mimic our own internal mappings.

Modalities Through the Ages: Post-World War II

Joysticks quickly moved out of airplanes and alongside radar and sonar displays during WWII. Immediately after the war, in 1946, the first display-specific input was invented. Ralph Benjamin, an engineer in the Royal Navy, conceived of the rollerball as an alternative to the existing joystick inputs: "The elegant ball-tracker stands by his aircraft direction display. He has one ball, which he holds in his hand, but his joystick has withered away." The indication seems to be that the rollerball could be held in the hand rather than set on a desk. However, the reality of manufacturing in 1946 meant that the original roller was a full-sized bowling ball. Unsurprisingly, the unwieldy, 10-pound rollerball did not replace the joystick.

This leads us to the five rules of computer input popularity. To take off, inputs must have the following characteristics:

- Cheap
- Reliable
- Comfortable
- Have software that makes use of it
- Have an acceptable user error rate

The last can be amortized by good software design that allows for nondestructive actions, but beware: after a certain point, even benign

errors can be annoying. Autocorrect on touchscreens is a great example of user error often overtaking software capabilities.

Even though the rollerball mouse wouldn't reach ubiquity until 1984 with the rise of the personal computer, many other types of inputs that were used with computers moved out of the military through the mid-1950s and into the private sector: joysticks, buttons and toggles, and, of course, the keyboard.

It might be surprising to learn that styluses predated the mouse. The light pen, or gun, created by SAGE in 1955, was an optical stylus that was timed to CRT refresh cycles and could be used to interact directly on monitors. Another mouse-like option, Data Equipment Company's Grafacon, resembled a block on a pivot that could be swung around to move the cursor. There was even work done on voice commands as early as 1952 with Bell Labs' Audrey system, though it recognized only 10 words.

By 1963, the first graphics software existed that allowed users to draw on MIT Lincoln Laboratory's TX-2's monitor, Sketchpad, created by Ivan Sutherland at MIT. GM and IBM had a similar joint venture, the Design Augmented by Computer, or DAC-1, which used a capacitance screen with a metal pencil, instead—faster than the light pen, which required waiting for the CRT to refresh.

Unfortunately, in both the light pen and metal pencil case, the displays were upright and thus the user had to hold up their arm for input—what became known as the infamous "gorilla arm." Great workout, but bad ergonomics. The RAND corporation had noticed this problem and had been working on a tablet-and-stylus solution for years, but it wasn't cheap: in 1964, the RAND stylus—confusingly, later also marketed as the Grafacon—cost around $18,000 (roughly $150,000 in 2018 dollars). It was years before the tablet-and-stylus combination would take off, well after the mouse and graphical user interface (GUI) system had been popularized.

In 1965, Eric Johnson, of the Royal Radar Establishment, published a paper on capacitive touchscreen devices and spent the next few years writing more clear use cases on the topic. It was picked up by

researchers at the European Organization for Nuclear Research (CERN), who created a working version by 1973.

By 1968, Doug Engelbart was ready to show the work that his lab, the Augmentation Research Center, had been doing at Stanford Research Institute since 1963. In a hall under San Francisco's Civic Center, he demonstrated his team's oNLine System (NLS) with a host of features now standard in modern computing: version control, networking, videoconferencing, multimedia emails, multiple windows, and working mouse integration, among many others. Although the NLS also required a chord keyboard and conventional keyboard for input, the mouse is now often mentioned as one of the key innovations. In fact, the NLS mouse ranked similarly useable to the light pen or ARC's proprietary knee input system in Engelbart's team's own research. Nor was it unique: German radio and TV manufacturer, Telefunken, released a mouse with its RKS 100-86, the *Rollkugel*, which was actually in commercial production the year Engelbart announced his prototype.

However, Engelbart certainly popularized the notion of the asymmetric freeform computer input. The actual designer of the mouse at ARC, Bill English, also pointed out one of the truths of digital modalities at the conclusion of his 1967 paper, "Display-Selection Techniques for Text Manipulation":

[I]t seems unrealistic to expect a flat statement that one device is better than another. The details of the usage system in which the device is to be embedded make too much difference.

No matter how good the hardware is, the most important aspect is how the software interprets the hardware input and normalizes for user intent.

NOTE

For more on how software design can affect user perception of inputs, I highly recommend the book *Game Feel: A Game Designer's Guide to Virtual Sensation* by Steve Swink (Morgan Kaufmann Game Design Books, 2008). Because each game has its own world and own system, the "feel" of the inputs can be rethought. There is less wiggle room for innovation in standard computer operating systems, which must feel familiar by default to avoid cognitive overload.

Another aspect of technology advances worth noting from the 1960s was the rise of science fiction, and therefore computing, in popular culture. TV shows like *Star Trek* (1966–1969) portrayed the use of voice commands, telepresence, smart watches, and miniature computers. *2001: A Space Odyssey* (1968) showed a small personal computing device that looks remarkably similar to the iPads of today as well as voice commands, video calls, and, of course, a very famous artificial intelligence. The animated cartoon, *The Jetsons* (1962–1963), had smart watches, as well as driverless cars and robotic assistance. Although the technology wasn't common or even available, people were being acclimated to the idea that computers would be small, lightweight, versatile, and have uses far beyond text input or calculations.

The 1970s was the decade just before personal computing. Home game consoles began being commercially produced, and arcades took off. Computers were increasingly affordable; available at top universities, and more common in commercial spaces. Joysticks, buttons, and toggles easily made the jump to video game inputs and began their own, separate trajectory as game controllers. Xerox Corporation's famous Palo Alto Research Center, or PARC, began work on an integrated mouse and GUI computer work system called the Alto. The Alto and its successor, the Star, were highly influential for the first wave of personal computers manufactured by Apple, Microsoft, Commodore, Dell, Atari, and others in the early to mid-1980s. PARC also created a prototype of Alan Kay's 1968 KiddiComp/Dynabook, one of the precursors of the modern computer tablet.

Modalities Through the Ages: The Rise of Personal Computing

Often, people think of the mouse and GUI as a huge and independent addition to computer modalities. But even in the 1970s, Summagraphics was making both low- and high-end tablet-and-stylus combinations for computers, one of which was white labeled for the Apple II as the Apple Graphics Tablet, released in 1979. It was relatively expensive and supported by only a few types of software; violating two of the five rules. By 1983, HP had released the HP-150, the first touchscreen

computer. However, the tracking fidelity was quite low, violating the user error rule.

When the mouse was first bundled with personal computer packages (1984–1985), it was supported on the operating-system (OS) level, which in turn was designed to take mouse input. This was a key turning point for computers: the mouse was no longer an optional input, but an *essential* one. Rather than a curio or optional peripheral, computers were now required to come with tutorials teaching users how to use a mouse, as illustrated in Figure 1-4—similar to how video games include a tutorial that teaches players how the game's actions map to the controller buttons.

Figure 1-4. Screenshot of the Macintosh SE Tour, 1987

It's easy to look back on the 1980s and think the personal computer was a standalone innovation. But, in general, there are very few innovations in computing that single-handedly moved the field forward in less than a decade. Even the most famous innovations, such as FORTRAN, took years to popularize and commercialize. Much more often, the driving force behind adoption—of what feels like a new innovation—is simply the result of the technology finally fulfilling the aforementioned five rules: *cheap*, *reliable*, *comfortable*, *have software that makes use of the technolgy*, and *having an acceptable user error rate*.

It is very common to find that the first version of what appears to be recent technology was in fact invented decades or even centuries ago. If the technology is obvious enough that multiple people try to build it but it still doesn't work, it is likely failing in one of the five rules. It simply must wait until technology improves or manufacturing processes catch up.

This truism is of course exemplified in virtual reality (VR) and augmented reality (AR) history. Although the first stereoscopic head-mounted displays (HMDs) were pioneered by Ivan Sutherland in the 1960s and have been used at NASA routinely since the 1990s, it wasn't until the fields of mobile electronics and powerful graphics processing units (GPUs) improved enough that the technology became available at a commercially acceptable price, decades later. Even as of today, high-end standalone HMDs are either thousands of dollars or not

commercially available. But much like smartphones in the early 2000s, we can see a clear path from current hardware to the future of spatial computing.

However, before we dive in to today's hardware, let's finish laying out the path from the PCs of the early 1980s to the most common types of computer today: the smartphone.

Modalities Through the Ages: Computer Miniaturization

Computers with miniaturized hardware emerged out of the calculator and computer industries as early as 1984 with the Psion Organizer. The first successful tablet computer was the GriDPad, released in 1989, whose VP of research, Jeff Hawkins, later went on to found the PalmPilot. Apple released the Newton in 1993, which had a handwritten character input system, but it never hit major sales goals. The project ended in 1998 as the Nokia 900 Communicator—a combination telephone and personal digital assistant (PDA)—and later the PalmPilot dominated the miniature computer landscape. Diamond Multimedia released its Rio PMP300 MP3 player in 1998, as well, which turned out to be a surprise hit during the holiday season. This led to the rise of other popular MP3 players by iRiver, Creative NOMAD, Apple, and others.

In general, PDAs tended to have stylus and keyboard inputs; more single-use devices like music players had simple button inputs. From almost the beginning of their manufacturing, the PalmPilots shipped with their handwriting recognition system, Graffiti, and by 1999 the Palm VII had network connectivity. The first Blackberry came out the same year with keyboard input, and by 2002 Blackberry had a more conventional phone and PDA combination device.

But these tiny computers didn't have the luxury of human-sized keyboards. This not only pushed the need for better handwriting recognition, but also real advances in speech input. Dragon Dictate came out in 1990 and was the first consumer option available—though for $9,000, it heavily violated the "cheap" rule. By 1992, AT&T rolled out voice recognition for its call centers. Lernout & Hauspie acquired several companies through the 1990s and was used in Windows XP.

After an accounting scandal, the company was bought by SoftScan—later Nuance, which was licensed as the first version of Siri.

In 2003, Microsoft launched Voice Command for its Windows Mobile PDA. By 2007, Google had hired away some Nuance engineers and was well on its way with its own voice recognition technology. Today, voice technology is increasingly ubiquitous, with most platforms offering or developing their own technology, especially on mobile devices. It's worth noting that in 2018, there is no cross-platform or even cross-company standard for voice inputs: the modality is simply not mature enough yet.

PDAs, handhelds, and smartphones have almost always been interchangeable with some existing technology since their inception—calculator, phone, music player, pager, messages display, or clock. In the end, they are all simply different slices of computer functionality. You can therefore think of the release of the iPhone in 2007 as a turning point for the small-computer industry: by 2008, Apple had sold 10 million more than the next top-selling device, the Nokia 2330 classic, even though the Nokia held steady sales of 15 million from 2007 to 2008. The iPhone itself did not take over iPod sales until 2010, after Apple allowed users to fully access iTunes.

One very strong trend with all small computer devices, whatever the brand, is the move toward touch inputs. There are several reasons for this.

The first is simply that visuals are both inviting and useful, and the more we can see, the higher is the perceived quality of the device. With smaller devices, space is at a premium, and so removing physical controls from the device means a larger percentage of the device is available for a display.

The second and third reasons are practical and manufacturing focused. As long as the technology is cheap and reliable, fewer moving parts means less production cost and less mechanical breakage, both enormous wins for hardware companies.

The fourth reason is that using your hands as an input is perceived as natural. Although it doesn't allow for minute gestures, a well-designed, simplified GUI can work around many of the problems that come up

around user error and occlusion. Much like the shift from keyboard to mouse-and-GUI, new interface guidelines for touch allow a reasonably consistent and error-free experience for users that would be almost impossible using touch with a mouse or stylus-based GUI.

The final reason for the move toward touch inputs is simply a matter of taste: current design trends are shifting toward minimalism in an era when computer technology can be overwhelming. Thus, a simplified device can be perceived as easier to use, even if the learning curve is much more difficult and features are removed.

One interesting connection point between hands and mice is the *trackpad*, which in recent years has the ability to mimic the multitouch gestures of touchpad while avoiding the occlusion problems of hand-to-display interactions. Because the tablet allows for relative input that can be a ratio of the overall screen size, it allows for more minute gestures, akin to a mouse or stylus. It still retains several of the same issues that plague hand input—fatigue and lack of the physical support that allows the human hand to do its most delicate work with tools—but it is useable for almost all conventional OS-level interactions.

Why Did We Just Go Over All of This?

So, what was the point of our brief history lesson? To set the proper stage going forward, where we will move from the realm of the known, computing today, to the unknown future of spatial inputs. At any given point in time it's easy to assume that we know everything that has led up to the present or that we're always on the right track. Reviewing where we've been and how the present came to be is an excellent way to make better decisions for the future.

Let's move on to exploring human–computer interaction (HCI) for spatial computing. We can begin with fundamentals that simply will not change in the short term: how humans can take in, process, and output information.

Types of Common HCI Modalities

There are three main ways by which we interact with computers:

Visual

Poses, graphics, text, UI, screens, animations

Auditory

Music, tones, sound effects, voice

Physical

Hardware, buttons, haptics, real objects

Notice that in the background we've covered so far, physical inputs and audio/visual outputs dominate HCI, regardless of computer type. Should this change for spatial computing, in a world in which your digital objects surround you and interact with the real world? Perhaps. Let's begin by diving into the pros and cons of each modality.

VISUAL MODALITIES

Pros:

- 250 to 300 words per minute (WPM) understood by humans
- Extremely customizable
- Instantly recognizable and understandable on the human side
- Very high fidelity compared to sound or haptics
- Time-independent; can just hang in space forever
- Easy to rearrange or remap without losing user understanding
- Good ambient modality; like ads or signs, can be noticed by the humans at their leisure

Cons:

- Easy to miss; location dependent
- As input, usually requires robust physical counterpart; gestures and poses very tiring
- Requires prefrontal cortex for processing and reacting to complicated information, which takes more cognitive load

- Occlusion and overlapping are the name of the game
- Most likely to “interrupt” if the user is in the flow
- Very precise visual (eye) tracking is processor intensive

Best uses in HMD-specific interactions:

- Good for limited camera view or other situations in which a user is forced to look somewhere
- Good for clear and obvious instructions
- Good for explaining a lot fast
- Great for tutorials and onboarding

Example use case—a smartphone:

- Designed to be visual-only
- Works even if the sound is off
- Works with physical feedback
- Physical affordances are minimal
- Lots of new animation languages to show feedback

PHYSICAL MODALITIES

Pros:

- Braille: 125 WPM
- Can be very fast and precise
- Bypasses high-level thought processes, so is easy to move into a physiological and mental “flow”
- Training feeds into the primary motor cortex; eventually doesn’t need the more intensive premotor cortex or basal ganglia processing
- Has strong animal brain “this is real” component; a strong reality cue
- Lightweight feedback is unconsciously acknowledged

- Least amount of delay between affordance and input
- Best single-modality input type, as is most precise

Cons:

- Can be tiring
- Physical hardware is more difficult to make, can be expensive, and breaks
- Much higher cognitive load during teaching phase
- Less flexible than visual: buttons can’t really be moved
- Modes require more memorization for real flow
- Wide variations due to human sensitivity

Best uses in HMD-specific interactions:

- Flow states
- Situations in which the user shouldn’t or can’t look at UI all the time
- Situations in which the user shouldn’t look at their hands all the time
- Where mastery is ideal or essential

Example use case—musical instruments:

- Comprehensive physical affordances
- No visuals needed after a certain mastery level; creator is in flow
- Will almost always have audio feedback component
- Allows movement to bypass parts of the brain—thought becomes action

AUDIO MODALITIES

Pros:

- 150 to 160 WPM understood by humans

- Omnidirectional
- Easily diegetic to both give feedback and enhance world feel
- Can be extremely subtle and still work well
- Like physical inputs, can be used to trigger reactions that don't require high-level brain processing, both evaluative conditioning and more base brain stem reflex
- Even extremely short sounds can be recognized after being taught
- Great for affordances and confirmation feedback

Cons:

- Easy for users to opt out with current devices
- No ability to control output fidelity
- Time based: if user misses it, must repeat
- Can be physically off-putting (brain stem reflex)
- Slower across the board
- Vague, imprecise input due to language limitations
- Dependent on timing and implementation
- Not as customizable
- Potentially processor intensive

Best uses in HMD-specific interactions:

- Good for visceral reactions
- Great way to get users looking at a specific thing
- Great for user-controlled camera
- Great when users are constrained visually and physically
- Great for mode switching

Example use case—a surgery room:

- Surgeon is visually and physically captive; audio is often the only choice

- Continual voice updates for all information
- Voice commands for tools, requests, and confirmations
- Voice can provide most dense information about current state of affairs and mental states; very useful in high-risk situations

Now that we've written down the pros and cons of each type of modality, we can delve into the HCI process and properly map out the cycle. Figure 1-5 illustrates a typical flow, followed by a description of how it maps to a game scenario.

Figure 1-5. Cycle of a typical HCI modality loop

The cycle comprises three simple parts that loop repeatedly in almost all HCIs:

- The first is generally the affordance or discovery phase, in which the user finds out *what they can do*.
- The second is the input or action phase, in which the user *does the thing*.
- The third phase is the feedback or confirmation phase, in which the computer *confirms the input* by reacting in some way.

Figure 1-6 presents the same graphic, now filled out for a conventional console video game tutorial UX loop.

Figure 1-6. The cycle of a typical HCI modality loop, with examples

Let's walk through this. In many video game tutorials, the first affordance with which a user can do something is generally an unmissable UI overlay that tells the user the label of the button that they need to press. This sometimes manifests with a corresponding image or model of the button. There might be an associated sound like a change in music, a tone, or dialogue, but during the tutorial it is largely supporting and not teaching.

For conventional console video games, the input stage will be entirely physical; for example, a button press. There are exploratory video games that might take advantage of audio input like speech, or a combination

of physical and visual inputs (e.g., hand pose), but those are rare. In almost all cases, the user will simply press a button to continue.

The feedback stage is often a combination of all three modalities: the controller might have haptic feedback, the visuals will almost certainly change, and there will be a confirmation sound.

It's worth noting that this particular loop is specifically describing the *tutorial* phase. As users familiarize themselves with and improve their gameplay, the visuals will diminish in favor of more visceral modalities. Often, later in the game, the sound affordance might become the primary affordance to avoid visual overload—remember that, similar to physical modalities, audio can also work to cause reactions that bypass higher-level brain functions. Visuals are the most information-dense modalities, but they are often the most distracting in a limited space; they also require the most time to understand and then react.

New Modalities

With the rise of better hardware and new sensors, we have new ways both to talk to computers and have them monitor and react to us. Here's a quick list of inputs that are either in the prototype or commercialization stage:

- Location
- Breath rate
- Voice tone, pitch, and frequency
- Eye movement
- Pupil dilation
- Heart rate
- Tracking unconscious limb movement

One curious property of these new inputs—as opposed to the three common modalities we've discussed—is that for the most part, the less the user thinks about them, the more useful they will be. Almost every one of these new modalities is difficult or impossible to control for long

periods of time, especially as a conscious input mechanic. Likewise, if the goal is to collect data for machine learning training, any conscious attempt to alter the data will likely dirty the entire set. Therefore, they are best suited to be described as passive inputs.

One other property of these specific inputs is that they are one-way; the computer can react to the change in each, but it cannot respond in kind, at least not until computers significantly change. Even then, most of the list will lead to ambient feedback loops, not direct or instant feedback.

The Current State of Modalities for Spatial Computing Devices

As of this writing, AR and VR devices have the following modality methods across most hardware offerings:

Physical

- For the user input: controllers
- For the computer output: haptics

Audio

- For the user input: speech recognition (rare)
- For the computer output: sounds and spatialize audio

Visual

- For the user input: hand tracking, hand pose recognition, and eye tracking
- For the computer output: HMD

One peculiarity arises from this list: immersive computing has, for the first time, led to the rise of *visual inputs* through computer vision tracking body parts like the hands and eyes. Although hand position and movement has often been incidentally important, insofar as it maps to pushing physical buttons, it has never before taken on an importance of

its own. We talk more on this later, but let's begin with the most conventional input type: controllers and touchscreens.

Current Controllers for Immersive Computing Systems

The most common type of controllers for mixed, augmented, and virtual reality (XR) headsets, owes its roots to conventional game controllers. It is very easy to trace any given commercial XR HMD's packaged controllers back to the design of the joystick and D-pad. Early work around motion tracked gloves, such as NASA Ames' VIEWlab from 1989, has not yet been commoditized at scale. Interestingly, Ivan Sutherland had posited that VR controllers should be joysticks back in 1964; almost all have them, or thumbpad equivalents, in 2018.

Before the first consumer headsets, Sixsense was an early mover in the space with its magnetic, tracked controllers, which included buttons on both controllers familiar to any game console: A and B, home, as well as more genericized buttons, joysticks, bumpers, and triggers.

Current fully tracked, PC-bound systems have similar inputs. The Oculus Rift controllers, Vive controllers, and Windows MR controllers all have the following in common:

- A primary select button (almost always a trigger)
- A secondary select variant (trigger, grip, or bumper)
- A/B button equivalents
- A circular input (thumbpad, joystick, or both)
- Several system-level buttons, for consistent basic operations across all applications

Figure 1-7. The Sixsense Stem input system

Generally, these last two items are used to call up menus and settings, leaving the active app to return to the home screen.

Standalone headsets have some subset of the previous list in their controllers. From the untracked Hololens remote to the Google

Daydream's three-degrees-of-freedom (3DOF) controller, you will always find the system-level buttons that can perform confirmations and then return to the home screen. Everything else depends on the capabilities of the HMD's tracking system and how the OS has been designed.

Although technically *raycasting* is a visually tracked input, most people will think of it as a physical input, so it does bear mentioning here. For example, the Magic Leap controller allows for selection both with raycast from the six-degrees-of-freedom (6DOF) controller and from using the thumbpad, as does the Rift in certain applications, such as its avatar creator. But, as of 2019, there is no standardization around raycast selection versus analog stick or thumbpad.

As tracking systems improve and standardize, we can expect this standard to solidify over time. Both are useful at different times, and much like the classic Y-axis inversion problem, it might be that different users have such strongly different preferences that we should always allow for both. Sometimes, you want to point at something to select it; sometimes you want to scroll over to select it. Why not both?

Body Tracking Technologies

Let's go through the three most commonly discussed types of body tracking today: *hand tracking*, *hand pose recognition*, and *eye tracking*.

HAND TRACKING

Hand tracking is when the entire movement of the hand is mapped to a digital skeleton, and input inferences are made based on the movement or pose of the hand. This allows for natural movements like picking up and dropping of digital objects and gesture recognition. Hand tracking can be entirely computer-vision based, include sensors attached to gloves, or use other types of tracking systems.

HAND POSE RECOGNITION

This concept is often confused with hand tracking, but hand pose recognition is its own specific field of research. The computer has been trained to recognize specific hand poses, much like sign language. The

intent is mapped when each hand pose is tied to specific events like grab, release, select, and other common actions.

On the plus side, pose recognition can be less processor intensive and need less individual calibration than robust hand tracking. But externally, it can be tiring and confusing to users who might not understand that the pose re-creation is more important than natural hand movement. It also requires a significant amount of user tutorials to teach hand poses.

EYE TRACKING

The eyes are constantly moving, but tracking their position makes it much easier to infer interest and intent—sometimes even more quickly than the user is aware of themselves, given that eye movements update before the brain visualization refreshes. Although it's quickly tiring as an input in and of itself, eye tracking is an excellent input to mix with other types of tracking. For example, it can be used to triangulate the position of the object a user is interested in, in combination with hand or controller tracking, even before the user has fully expressed an interest.

I'm not yet including body tracking or speech recognition on the list, largely because there are no technologies on the market today that are even beginning to implement either technology as a standard input technique. But companies like Leap Motion, Magic Leap, and Microsoft are paving the way for all of the nascent tracking types listed here.

A Note on Hand Tracking and Hand Pose Recognition

Hand tracking and hand pose recognition both must result in interesting, and somewhat counterintuitive, changes to how humans often think of interacting with computers. Outside of conversational gestures, in which hand movement largely plays a supporting role, humans do not generally ascribe a significance to the location and pose of their hands. We use hands every day as tools and can recognize a mimicked gesture for the action it relates to, like picking up an object. Yet in the history of HCI, hand location means very little. In fact, peripherals like the mouse and the game controller are specifically designed to be hand-location

agnostic: you can use a mouse on the left or right side, you can hold a controller a foot up or down in front of you; it makes no difference to what you input.

The glaring exception to this rule is touch devices, for which hand location and input are necessarily tightly connected. Even then, touch "gestures" have little to do with hand movement outside of the fingertips touching the device; you can do a three-finger swipe with any three fingers you choose. The only really important thing is that you fulfill the minimum requirement to do what the computer is looking for to get the result you want.

Computer vision that can track hands, eyes, and bodies is potentially extremely powerful, but it can be misused.

Voice, Hands, and Hardware Inputs over the Next Generation

If you were to ask most people on the street, the common assumption is that we will ideally, and eventually, interact with our computers the way we interact with other humans: talking normally and using our hands to gesture and interact. Many, many well-funded teams across various companies are working on this problem today, and both of those input types will surely be perfected in the coming decades. However, they both have significant drawbacks that people don't often consider when they imagine the best-case scenario of instant, complete hand tracking and NLP.

VOICE

In common vernacular, voice commands aren't precise, no matter how perfectly understood. People often misunderstand even plain-language sentences, and often others use a combination of inference, metaphor, and synonyms to get their real intent across. In other words, they use multiple modalities and modalities within modalities to make sure they are understood. Jargon is an interesting linguistic evolution of this: highly specialized words that mean a specific thing in a specific context to a group are a form of language hotkey, if you will.

Computers can react much more quickly than humans can—that is their biggest advantage. To reduce input to mere human vocalization means

that we significantly slow down how we can communicate with computers from today. Typing, tapping, and pushing action-mapped buttons are all very fast and precise. For example, it is much faster to select a piece of text, press the hotkeys for "cut," move the cursor, and then press the hotkeys for "paste" than it is to describe those actions to a computer. This is true of almost all *actions.*

However, to describe a scenario, tell a story, or make a plan with another human, it's often faster to simply use words in conversations because any potential misunderstanding can be immediately questioned and course-corrected by the listener. This requires a level of working knowledge of the world that computers will likely not have until the dawn of true artificial intelligence.

There are other advantages to voice input: when you need hands-free input, when you are otherwise occupied, when you need transliteration dictation, or when you want a fast modality switch (e.g., "minimize! exit!") without other movement. Voice input will always work best when it is used in tandem with other modalities, but that's no reason it shouldn't be perfected. And, of course, voice recognition and speech-to-text transcription technology has uses beyond mere input.