ChatGPT

Multimodal Interaction Design for Real-Time Control (Gesture, Voice, Biofeedback)

Gaze Targeting and Dwell Selection

Gaze is used as the primary pointing modality, letting the user **target** herbs or UI elements by looking at them. To select a target without accidental "Midas touch" activation, a **dwell selection** mechanism is employed: the user keeps their gaze on the object for a short threshold (e.g. ~0.5–1.0 seconds) until a visual indicator (like a filling circle) completes, triggering selection [1] [2]. This dwell time acts as a **confirmatory delay**, balancing speed against false triggers – shorter dwell makes selection faster but risks unintended activation if the user just glances around [3]. Studies find that dwell-based gaze selection can be very efficient: in fact, gaze dwell was observed to be faster and more accurate than voice or pure gesture triggers in head-mounted display interactions [4]. The system should provide clear feedback (e.g. highlighting the object and showing a progress ring during dwell) so the user knows when the selection will fire [1]. Gaze targeting is inherently hands-free and works in any posture, but *requires a confirmation step* (via dwell or a gesture) to avoid inadvertent commands – this confirmation is part of our conflict avoidance strategy for gaze input (addressing the classic "Midas touch" problem [5] [2]).

If the user's hands are free, an alternative to dwell is using a quick **hand gesture** (like an "air tap" pinch) to confirm after gazing at the target. For example, on HoloLens an *"air tap"* (raising the index finger then tapping it down) while gazing at a hologram performs a selection, functioning like a mouse click [6]. We adopt a similar approach: the user looks at the desired herb and then taps their index finger and thumb together (a pinch gesture) to instantly select it. This gaze+pinch method leverages the natural fact that eye gaze usually precedes hand movement, making the interaction fast and intuitive [7] [8]. Modern AR/MR headsets already exploit such combinations – using eye tracking to aim and a pinch to commit – as a standard multimodal selection technique [9]. In summary, **gaze provides the aiming** (highlights the intended object) and **dwell or a pinch provides the commit**, ensuring that an herb is only "picked" when the user deliberately indicates so.

Hand Gesture Controls and Minimal Gesture Set

Hand gestures form the core control inputs for manipulating the herbs. We design a minimal set of **intuitive, low-fatigue gestures** to cover the required actions (selecting, blending, pinning) while avoiding complex or tiring motions. Freehand interaction is popular in XR because it feels natural – users can grab and move virtual objects similarly to real ones [10]. However, prolonged mid-air gestures can cause arm fatigue [11], so our gesture vocabulary is kept lean and designed for comfort. The primary gestures are:

- **Pinch (Air-Tap) for Select:** As described above, a simple pinch of thumb and index finger while gazing at an herb will select or "pick up" that herb [6]. This is analogous to clicking an item. It's a quick gesture with minimal movement, reducing strain. The pinch gesture doubles as a general **activate/confirm** command in our grammar – for example, pinching while gazing at a UI button would activate that button, similar to how HoloLens' air-tap is a universal click [6]. Pinch is a

fundamental gesture recognized across many XR systems and is easy to perform within the comfortable field of view of the device's hand tracking sensors (within the "gesture frame" in front of the user) [12].

- **Grab and Release for Move/Pin:** To **pin an herb** in place (anchoring it to a location), the user can perform a grab and release motion. In practice, once an herb is selected (by pinch or dwell), the system will allow the user to "grab" it by closing their hand or pinching and holding. The user can then move their hand to drag the virtual herb and **release** (opening the pinch) to drop/pin it at the desired spot. This mimics real-life placement and uses the same pinch gesture in an extended way (pinch-and-hold). The pin action thus doesn't require a new discrete gesture – it's achieved by the **context** of releasing a grabbed object. For example, a user might gaze at a surface (like a table in AR) and release the herb there, causing it to pin to that surface. This approach aligns with direct manipulation principles and avoids additional "command" gestures. It also leverages the headset's spatial anchoring capability so that pinned herbs remain fixed in the world [13] [14] (even if the user moves around). If needed, a voice command ("pin here") could serve as an alternative to release for pinning (more on voice in the next section).

- **Two-Hand Combine for Blending:** To **blend herbs**, we use a bimanual gesture to make the action feel tangible. The user selects or grabs two different herbs (one with each hand, or one after the other with the same hand) and then brings their hands together as if mixing or clapping the two virtual ingredients. When the two selected herbs come into close proximity, the system will initiate a blend (visualizing the herbs merging or a new combined herb appearing). A subtle twisting or stirring motion of the hands can be used to indicate a thorough mix. This two-handed gesture is minimal in the sense that it's a natural extension of the grab gesture – essentially **moving two held objects together** – rather than an abstract symbol. Two-handed interactions in AR can increase expressiveness without additional symbolic commands [15] [16]. By using the position and movement of the two herbs (which the system tracks in real time), we avoid adding a special "blend" sign; the *intention* is inferred from the context (two herbs selected + moving together). This reduces cognitive load on the user to remember specific gestures. The system's conflict-avoidance here is to only trigger blending when two herbs are deliberately brought close, with a small tolerance to avoid accidental brushes. If the user cannot perform this gesture, an alternative could be a voice command ("blend these") after selecting both herbs, but the gesture is the primary method for a seamless hands-on experience.

Overall, our gesture set largely reuses the **pinch/grab** gesture in various forms (single pinch to select, sustained pinch to grab, release to drop) and a simple two-hand movement for blending. This consistency keeps the "grammar" simple, as the user essentially learns one core hand pose and uses it in different contexts. Recent design guidance for AR emphasizes using the same basic hand poses for multiple interactions (with system automation to switch between near manipulation vs. far ray-casting as needed) [17]. We adopt that philosophy: *one mental model* for hand input, to minimize mode switches. Furthermore, hand interactions are tracked continuously so slight body motions are tolerated – the user can be seated or standing and need not hold perfectly still. The system's hand tracking (e.g. via computer vision and IMU sensors) updates in real time, ensuring that gentle swaying or repositioning doesn't break the gesture recognition [18]. In practice, the user should keep their hands within the headset's tracking range (roughly a few feet in front) for optimal detection [12], but normal head and torso movements are fine. This ensures the interface remains usable even if the user is in light motion (for example, reaching around or if presenting to a small group while standing).

Voice Commands as Optional Input

Voice input serves as an **optional complementary modality** for users who have speech available and want a hands-free or quicker way to issue commands. The system can parse simple voice commands (e.g. "select *[herb name]*", "blend these", "pin this herb") to either replicate the gesture-based actions or perform higher-level intents. Voice can be very powerful for commands that might be cumbersome with gestures – for instance, saying *"blend rosemary and thyme"* could directly initiate that mix without needing to grab each herb in hand. It also allows naming herbs or actions in a way that hand gestures alone cannot. In our design, voice is **not mandatory** (since gestures and gaze suffice), but it provides an extra layer of convenience and accessibility when appropriate.

However, we carefully design voice interaction to avoid conflicts and ensure usability. One consideration is that voice recognition requires the user to remember the correct commands and speak them clearly. This can increase cognitive load, especially if the command set is large or complex [19]. We mitigate this by keeping the voice command grammar simple and aligned with on-screen prompts (for example, the system might display hints like **"say 'blend' to mix selected herbs"** so the user doesn't have to recall exact phrasing). Another consideration is the environment: in a noisy setting or in a shared space, voice may be impractical. Voice input also has **social acceptability** issues – users may feel self-conscious speaking out loud or might disturb others [20]. Because the primary use-case here is single-user, speaking a short command is feasible, but if the mode shifts to a group demonstration, the user might prefer silent gestures to avoid confusion. We therefore treat voice as an **on-demand mode**: the system could require a push-to-talk (e.g. a specific voice activation keyword or a press of a wearable button) to ensure that only deliberate speech is captured as commands. This prevents stray conversation from triggering anything.

When used, voice commands integrate with gaze and gesture in a complementary way. For example, the user can gaze at a particular herb and say "pin this here" – the system will pin the gazed-at herb at the current gaze target location (perhaps the surface the user is looking at). In this case gaze provides context (which herb, which location) and voice provides the action intent ("pin") – a multimodal combination that is robust against ambiguity. This division of labor follows a known pattern in multimodal interface design: *speech for the verb, gaze/gesture for the object* [21]. By assigning distinct roles to modalities, we reduce conflicts (the voice command is only executed in reference to what the user is already targeting with eyes/hands, not on some random object). Additionally, if voice and gesture inputs occur simultaneously, the system interprets them together. For instance, if the user says "blend this with that" while pointing at two different herbs in turn, the temporal coincidence of the pointing gestures with the spoken "this"/"that" lets the system know which herbs to blend. This kind of **cross-modal disambiguation** is a strength of multimodal systems: speech and gesture together make the intention clearer than either alone [16] [22]. We take advantage of this by designing the voice parser and gesture recognizer to share state – e.g. the word "this" maps to the currently gazed-at item, etc. In summary, voice input is a flexible add-on that can speed up or simplify certain actions, but it is used in tandem with gaze and gesture context to avoid confusion. Users who are unable to use voice (or prefer not to) lose no core functionality, they simply rely on the gesture/gaze alternatives.

Biofeedback Integration (EEG/HRV, Optional)

Biofeedback inputs – such as EEG (electroencephalography) signals or heart rate variability (HRV) from a wearable – are an **optional enhancement** to the system, used only when such sensors are available and the context permits. These physiological signals will not issue direct "commands" like a gesture or voice

would; instead, they provide a continuous stream of user state data that the interface can use to adapt or personalize the experience. Our design follows established patterns in biofeedback-driven interfaces, where physiology is used to gauge the user's cognitive or emotional state and adjust the UI accordingly [23].

For instance, EEG can offer insight into the user's level of engagement or mental workload. If the system detects, say, a high **cognitive load or stress** (perhaps via certain EEG frequency patterns or a drop in HRV indicating stress), it could respond by simplifying the interface – e.g. temporarily highlighting the most relevant herbs or slowing down UI animations to reduce overload. Conversely, if biofeedback suggests the user is **very relaxed or under-stimulated**, the system might gently prompt them or introduce a more stimulating visual effect to maintain engagement. In a scenario where herb blending has a meditative or educational aspect, EEG could even be used as an input to *encourage focus*: for example, only allow blending to complete when the user maintains a focused brain state for a few seconds (this would be an explicit biofeedback interaction design, though we'd implement it only if it feels natural to the use-case). Heart rate and HRV could similarly inform the interface's pacing – a rising heart rate might indicate excitement or anxiety, which could trigger the UI to, say, enter a "calm mode" with softer visuals if the goal is to keep the user relaxed.

We also consider **biofeedback display** as part of the UI patterns. Rather than hiding these signals, we can visualize them in subtle ways to make the user aware of their state and perhaps encourage self-regulation. For example, an ambient background element might glow or pulse at the rate of the user's heartbeat, or an icon could change color when their stress level crosses a threshold. Prior research has done something analogous – for instance, animating virtual elements (like ocean waves) based on EEG rhythms to induce a calming biofeedback loop [24]. Such feedback can enhance immersion and self-awareness. Again, these are optional features: they only appear if the user has opted in with a connected bio sensor.

From a technical perspective, integrating biofeedback requires filtering noise and avoiding over-sensitivity. EEG and HRV data can be erratic, so the system uses short time-window averages and only triggers changes when a consistent pattern is detected (to avoid reacting to every tiny fluctuation). Latency is less critical here than with gesture control – a slight delay (e.g. 0.5–1 second) in interpreting biofeedback is acceptable, as it's influencing background aspects, not moment-to-moment selection. In our model, **physiological inputs modulate the UI rather than directly controlling discrete actions**, ensuring they don't conflict with the user's intentional inputs. For example, if the user is manually blending herbs (a clear deliberate action), the system will not let a biofeedback spike override or interrupt that. Instead, biofeedback might be used before or after the action (e.g. to decide if a result should be presented in a calm vs. exciting visual manner). This design approach prevents bio signals – which can be involuntary – from causing any unexpected or undesired commands, thereby avoiding conflict with the conscious gesture/voice commands. When used appropriately, biofeedback can enrich the interaction by tailoring it to the user's internal state, adding a novel dimension to the herb-blending experience (such as making it a **calming ritual** if the biofeedback indicates stress, etc.). Indeed, studies in VR have shown that EEG and heart-rate data can classify a user's arousal or stress level, allowing systems to adjust difficulty or feedback to keep users in an optimal state [23] [25]. We leverage the same principle in a lightweight manner here.

Interaction Grammar and Control Mapping

Bringing together the modalities, we define an **interaction grammar** that specifies how gaze, gesture, voice (if used), and biofeedback work in concert without interfering with each other. The guiding principle is that each modality has a clear purpose in the interaction, and ambiguous situations are resolved by context

or hierarchy. Below is a summary of the control mapping for the key actions, followed by the grammar rules underpinning them:

- **Select an Herb:** *Primary method:* Gaze at the herb and pinch (air-tap) to select it. *Hands-free method:* Gaze and dwell on the herb for ~0.5s to select. The herb is now "active" – e.g. highlighted or attached to the user's hand cursor. *(Voice alternative:* say "select [herb name]" while gazing at the herb, which is equivalent to the pinch.) In the grammar, this can be seen as: **TARGET (gaze) + CONFIRM (gesture/voice)**. The system only selects when both parts are present, so just gazing alone or just saying "select" alone does nothing – preventing accidental triggers.

- **Blend Herbs:** There are two prerequisites – Herb A and Herb B must be selected (or grabbed). The user then performs the blend gesture: move the two herbs together (the system detects collision/proximity) and optionally rotate/mix them. *Alternate:* With one herb in hand (selected) and another just targeted by gaze, the user can say "blend with this" or a similar phrase, upon which the system will select the gazed herb and blend the two. The grammar here might be described as **[Herb A] + [Herb B] + BLEND(action)**, where the action can be implicit (the act of touching them together) or explicit (a voice command). We ensure **mutual exclusivity** between these triggers to avoid confusion: if the user speaks "blend" we expect them *not* to simultaneously perform the full two-hand gesture (though they might still be holding the items). Conversely, during the physical blend gesture, we momentarily ignore any stray voice input like casual commentary. This is an example of our conflict avoidance: modalities can be used in parallel, but when one modality is clearly executing a command (e.g. a big two-hand motion), we give it priority and treat other inputs as either part of that command or noise. Successfully blending yields a new virtual object (or combined state), which the system could auto-select for convenience (so the user can continue to manipulate the blend result).

- **Pin (Anchor) an Herb:** *Method 1:* While holding an herb (after grabbing it), the user gazes at the desired location (for example, a spot on a surface or in the air) and releases the pinch. The grammar sequence is **HOLD + GAZE + RELEASE**, meaning "place object here." *Method 2:* Gaze at the herb (not held, just highlighted) and use a voice command like "pin here" or "pin *[object]* there" (with an additional gaze to indicate "there" if needed). In this case the grammar is **TARGET + VERBAL COMMAND + (optional second TARGET)**. We design the voice grammar such that if a single object is targeted when "pin here" is spoken, the system assumes the user wants to pin it at the current gaze point in space; if two gaze targets are involved (first herb, then location) and a command like "put that there" is spoken, it will pin accordingly. This echoes the classic "put-that-there" style interface, using speech and gaze in combination to reduce ambiguity [16] [26]. Again, only the co-occurrence of the cues triggers the action – e.g. saying "pin here" without an object in focus does nothing, and gazing at a surface without releasing or saying the command does nothing. This rule-based combination prevents conflicts. Once pinned, an herb might appear in a "pinned items" UI list for the user, and it remains in the world anchored. To **unpin** it, the user could simply grab it again (breaking the anchor) or issue an "unpin" voice command while looking at it.

- **Other Commands/UI:** The user can always cancel or reset by voice ("cancel") or a specific gesture (perhaps a two-handed wave or pressing a physical button if available). Canceling clears any selections and pending actions. We give *cancel* the highest priority to override other inputs for safety. For any UI panels or menus (for example, a herb list or settings), gaze and pinch/dwell works

for buttons just like selecting herbs. Voice could also be used (e.g. “open menu”) in a non-ambiguous way when no other action is happening.

This multimodal grammar ensures that at any given time, the system knows what the user’s intended focus and action are. By designating gaze as the pointer to **objects of interest** and hand gestures or voice as the **action triggers**, we minimize overlap – it’s clear which modality is doing what. This separation is supported by research showing that assigning different roles to each input channel leads to more robust multimodal interpretation [21]. For example, in our system you wouldn’t have two different modalities trying to perform the exact same sub-action at the same time (no voice-only selection without gaze, no gesture-only selection without gaze). This avoids conflicts such as the system selecting the wrong target or performing an action on the wrong item. Additionally, we fuse inputs within a short temporal window to interpret compound commands. If the user issues a voice command while making a gesture, the system checks if they are part of one combined instruction. If they are logically connected (like pointing at something while saying something that expects a target), the system merges them into one command. If they are unrelated (user talks to someone in the room while grabbing an object), the system, thanks to the **contextual awareness**, will either ignore the irrelevant input or postpone it. This **context-driven fusion** draws on the principle of mutual disambiguation: inputs from one modality can clarify another [22] [26]. For instance, a spoken phrase referring to “this herb” is only actionable if the user is actually gazing at a specific herb at that moment – if not, the system may ask for clarification or do nothing, rather than guessing. By requiring that alignment, we prevent accidental or mis-targeted commands.

In the background, a simple state machine or multimodal interpreter manages these rules. It might have states like *Idle*, *HerbSelected*, *TwoHerbsSelected*, *HoldingHerb*, etc., and events from each modality transition the states when they fit the grammar. Conflicting inputs (those that don’t fit any valid state transition) are ignored or prompt the user to try again. This model inherently avoids conflict by design: only certain combinations are valid, and the rest are filtered out. In practice, users quickly learn the “grammar” because it closely mirrors natural behavior (look at object, do action). The multimodal system thereby feels fluid and responsive, with each modality enhancing the others rather than interfering.

Conflict Avoidance and Modal Coordination

A critical aspect of this design is ensuring that using multiple input modes *in parallel* does not lead to erroneous or unintended activations. We have baked conflict-avoidance into the interaction grammar as discussed, and here we summarize the key strategies:

- **Modality Separation of Concerns:** Each modality has a primary role: gaze for pointing, gesture for confirming or manipulating, voice for commanding, biofeedback for background context. Because each channel addresses a different aspect of the interaction, they are less likely to clash. For example, the system will never interpret a hand gesture as a different command when a voice command is active – it knows the hand gesture is either part of the same combined action or incidental. This approach is supported by multimodal interface research which shows that dividing tasks between modalities (rather than letting them redundantly do the same thing) improves usability and reduces errors [21] [16].

- **Concurrent Input Fusion:** We allow modalities to operate simultaneously by **fusing inputs intelligently**. The system synchronizes input events within a short timeframe (say 300–500 ms) to see if they complement each other. If yes, they are merged into one command (as in the “point at X

while saying Y" scenarios). If not, one modality may take precedence based on context. For instance, during a delicate dragging gesture, we might temporarily give hand input priority and treat any coincident voice input as incidental (unless it's a cancel command). Conversely, if the user is in the middle of speaking a multi-word command, we might momentarily ignore quick gestures unless they clearly relate (like pointing). This temporal coordination prevents inputs from talking over each other in the system's decision process. It's akin to having a conversation: both hands and voice are "listened to," but the system ensures they don't conflict by using timing and context to judge intent [22] [26].

- **Spatial and Contextual Gating:** Many commands are only enabled in certain contexts, which avoids conflict by not having multiple active triggers. For example, the pinch gesture means "select" when your gaze is on a selectable object, but if your gaze isn't on something select-able, pinching might do nothing (or could be interpreted differently, like a generic "confirm" in a dialog). This context-sensitive interpretation means the same physical action won't accidentally do two things. Gaze itself is a context filter for voice and gesture — e.g. the word "blend" will only execute if two herbs are selected, otherwise the system ignores it as it doesn't make sense. By designing *commands that require specific contextual prerequisites*, we eliminate a whole class of errors (the system simply doesn't execute impossible or out-of-place commands).

- **Deliberate Gestures with Thresholds:** We tune the gesture recognizer to require a clear deliberate motion before firing. For example, the pinch must cross a certain threshold of finger distance to count as a click. Similarly, bringing herbs together to blend might require them to be within a few centimeters for a second or two before blending triggers. This ensures that casual hand movements or momentary overlaps (which can happen if the user is just rearranging items) don't accidentally blend or select. Essentially, *we add slight hysteresis and confirmation requirements to critical gestures*. The dwell selection already embodies this principle (requiring a sustained look). These thresholds act as friction that filters out unintentional input.

- **Feedback and the Opportunity to Abort:** The UI provides real-time feedback for each modality so users can self-correct if something unintended is about to happen. For instance, when the user says "blend", the system could highlight the two target herbs it thinks should be blended (based on gaze) *before* actually combining them, giving the user a split second to notice if the wrong items are highlighted and cancel. Similarly, a small tooltip might appear when a voice command is recognized ("Command: Blend") for confirmation. Gestures like pinch can have an audio or haptic click feedback as soon as registered. This feedback loop means if any modality misfires or is misinterpreted, the user can quickly catch it and undo or stop (e.g. by a quick "cancel" or simply not completing the action). Providing clear feedback is a known best practice to handle recognition uncertainty and avoid cascading errors [27] [28].

- **Error Recovery Hierarchy:** In case a conflict or ambiguity still occurs, we implement a simple hierarchy to resolve it consistently. Generally, *explicit user-triggered cancel or undo commands override everything*. After that, safety takes priority: for example, if there's ambiguity about which object to act on, the system will choose the action that has the least destructive effect (or ask the user to clarify). Non-critical inputs can be dropped if uncertain. We also favor hand gesture intent over voice if they contradict (since hands usually reflect more conscious spatial intent, whereas voice might be picked up incorrectly in noise). This hierarchy is tuned based on expected use; importantly, it is

communicated to the user through the interface guide so they know, for instance, that a physical gesture will trump a simultaneous voice command if both are given.

In essence, our conflict avoidance strategy is to **design away most conflicts** by structuring the multimodal commands, and to handle any remaining ambiguities through intelligent fusion and user-centric rules (with feedback). By taking advantage of each modality's strengths (e.g. the precision of gaze for indicating targets, the naturalness of gesture for manipulation, the richness of voice for commands, and the insight of biofeedback for context), the system creates a harmonious interaction where modalities reinforce rather than contradict each other. Prior research has shown that such multimodal systems not only reduce error rates compared to single-mode interfaces, but also let users fall back on another modality if one mode fails or is inconvenient [29] [26]. For example, if the speech recognizer doesn't understand an accent, the user can rely on gestures more heavily – and our interface will still function fully. This redundancy and flexibility further prevent conflict because the user can choose the mode that works best in the moment, rather than forcing an ill-suited mode that might lead to mistakes [29].

Latency and Performance Targets

To enable **real-time control**, the system must respond quickly and smoothly to user inputs across all modalities. We establish the following latency targets and performance considerations:

- **Overall System Response:** Aim for **sub-100ms latency** for any primary interaction feedback. A classic HCI principle is that 0.1 second is the threshold for the system reaction to feel instantaneous and directly caused by the user's action [30]. We want the user to feel that when they pinch or speak or look at something, the system reacts immediately, without noticeable lag. Therefore, from the moment a gesture is made or dwell selection confirmed, the application should highlight or select the herb within a few tenths of a second at most. Staying under ~100 ms ensures the user perceives a seamless interaction where the virtual herbs move or activate as if part of the real world [28]. If this threshold is exceeded significantly (e.g. half a second delay), the user will notice and it will break the flow, so we treat 100 ms as a **max target for critical feedback**.

- **Hand and Head Tracking Latency:** The AR headset's built-in tracking (head orientation, hand position) typically runs at high frame rates (30–60+ Hz for hand tracking, and much higher for head IMU). We leverage that so that visual feedback (like herb objects following the user's hand when grabbed) has minimal lag. In practice, the motion-to-photon latency for hand movements should ideally be <50 ms; modern systems often achieve ~20 ms or better for head tracking to prevent registration errors [31] [32]. For our purposes, as long as the herb appears to stick to the user's hand without jitter when moved, we consider tracking latency acceptable. We also enable prediction smoothing if available (to compensate for any minor sensor lag). The **goal** is that when the user reaches out and grabs or moves an object, the virtual rendering of that object keeps up with their hand with no perceivable delay, maintaining immersion.

- **Gesture Recognition Time:** Simple discrete gestures like a pinch can be recognized almost immediately (within a few frames of the finger meeting thumb). The system should interpret the pinch and trigger the selection within e.g. ~30–50 ms of the fingers touching. We will tune the recognition algorithm to be both quick and reasonably sure (for instance, a pinch might register after a debounce of 1-2 frames to avoid noise). Similarly, releasing a pinch (to drop an object) should be detected with minimal lag so the object drops exactly when the user expects. Our target is that

the **round-trip from gesture to UI update is well under 100 ms**, ideally ~50 ms which feels instantaneous. This falls in line with research that suggests even in typical computer interactions users start to notice latency around 60 ms or more [33]. Keeping gesture response below that threshold ensures it "feels" immediate.

- **Voice Command Latency:** Voice commands inherently take longer, because the user must speak and the system must recognize the speech. A short command might take ~0.5–1.0 second to utter. We target the voice recognition and execution to complete within **~300–500 ms after the user finishes speaking**. This means if a user says "blend these", the system should start blending action at most half a second after the phrase ends. In many cases, we can do incremental processing (streaming ASR) to shave this down. If using an on-device speech recognizer with a limited grammar, sub-0.5s response is feasible. According to usability guidelines, a 1 second response still keeps the user's flow of thought uninterrupted [34], so we treat 1s as an upper bound for voice feedback. If a voice command is going to take longer (due to a complex operation or cloud processing), we will provide immediate acknowledgment (e.g. a brief audio cue or on-screen "…" indicator) to show the command was heard, and then complete the action as soon as possible. This feedback prevents the user from feeling the system missed their command during any unavoidable delay [28].

- **Feedback Frame Rates:** Visual feedback like highlights, progress bars (for dwell), or herb movement animations should run at a comfortable frame rate (ideally 60 FPS) to appear smooth. Jittery or low-FPS feedback can make latency more noticeable. The AR display's refresh rate (for example 60 Hz on HoloLens) sets the baseline; we ensure our content updates are synced to that and do not drop frames under typical loads. In interaction, a consistent 60 FPS with, say, 16 ms frame times means any input processing must fit in that budget too. We will utilize the platform's capability (like HoloLens's hand tracking system which is optimized to feed data each frame) and keep our logic lightweight.

- **Latency and Biofeedback:** Biofeedback signals (EEG, HRV) do not require instant reaction. Typically, you'd analyze EEG over a short window (e.g. last 1–2 seconds) to determine a state like "focused" vs "unfocused." Thus, a few hundred milliseconds to even a second of latency in applying a biofeedback-driven change is acceptable. The user will not expect immediate, snappy responses from these subtle channels. What's more important is stability – avoiding oscillations if the bio signals fluctuate rapidly. We set a smoothing filter so that, for example, heart rate trends need to persist for a couple of seconds before the interface reacts (preventing, say, a momentary spike from causing a sudden UI change). This yields a kind of *low-pass filtered control* with an effective update rate perhaps on the order of 0.5 Hz. That is fine since biofeedback is supplementary. In summary, biofeedback integration is **high-latency tolerant** by nature, and we prioritize meaningful, steady changes over quick reactions for this channel.

- **System and Group Context:** When multiple users are present (in secondary shared viewing mode), performance considerations include maintaining **tracking quality despite potential occlusions or distractions**. If the primary user is demonstrating to others, they might turn their head or body more frequently – the system must keep up with these motions (which it can within the above latency targets). We also avoid any mode that would require ultra-low latency networking or multi-user input sync, since our secondary mode is viewing-only. For local performance, we ensure the device can handle the sensor input streams (eye tracking, hand tracking, voice processing) concurrently. Most modern AR HMDs are designed for this, but we budget CPU/GPU accordingly. If

biofeedback sensors are used (like a Bluetooth HR sensor), we ensure that processing (usually negligible) doesn't introduce hiccups in the main loop.

In terms of **quantitative targets**: to summarize, we strive for **<100 ms end-to-end latency** for hand gesture interactions and basic selections (to feel instantaneous), ~500 ms or less for voice command execution (to feel responsive within the flow of conversation), and on the order of 1–2 seconds for detectable biofeedback-driven adaptations (since those are gradual). These align with human perception thresholds and XR industry standards for responsiveness [30] [31]. Where possible, we aim even lower – e.g. hand tracking visual updates ideally <20 ms motion-to-photon to keep virtual herbs firmly registered in place without lag [31]. By meeting these targets, the interface will feel **real-time and fluid**, allowing the user to manipulate virtual objects as naturally as real ones and trust the system to react at the speed of their thought and action.

User Posture and Multi-User Context Considerations

Our interaction design is built to function in the typical user postures of this scenario: primarily standing or seated with the ability to make natural arm movements, turn the head, and even take small steps. The system tolerates **gentle motion** and does not require the user to be static. For example, if the user is walking slowly around a table while examining a holographic herb, the gaze and hand tracking will continue to work as long as the herb stays in view. The interface is resilient to such movement because the input modalities are relative: gaze selection moves with the user (the dwell selection box stays on whatever object they are looking at), and hand gestures are tracked in the user's coordinate frame. Prior work in multimodal AR interfaces explicitly notes the importance of supporting use while the user is in motion or at a distance, rather than assuming a fixed position [18]. We follow that guidance. The only caveat is that very rapid or erratic movement can reduce tracking accuracy (e.g. sudden head turns might briefly lose a hand in view), but the system is designed to smooth small motions and simply pause input recognition during momentary losses rather than mis-trigger. For instance, if the user jogs or knocks the headset, we might temporarily halt dwell timers and resume when stable, so that a bump doesn't select the wrong item.

In seated use, the user can rest their arms as needed and perform the pinch gestures in their lap or mid-air comfortably. We ensure that UI elements (like any menus or highlights) are placed at comfortable viewing angles and distances (roughly 1–2 meters away and not requiring extreme head tilt) [35] [36]. This prevents fatigue and accommodates different body positions. The "gesture frame" (the space where hand gestures are recognized) moves with the user and is roughly chest to head height in front of them [12]; this is convenient whether seated or standing.

For the **multi-user context**, we interpret it as a scenario where one person (the primary user) is interacting with the system (wearing the AR device or using the interface) while a small group of others observe, for example watching a projection or the user's viewpoint on a screen, or by standing around the AR view if it's shared holograms. Our design primarily supports single-user control, but there are considerations for this shared experience:

- **Visibility of Actions:** The system provides visual cues for the user's actions that also help bystanders follow along. For instance, when the user gazes at an herb, that herb could glow or get an outline – so others know what is being targeted. When a voice command is spoken, perhaps a subtitle or icon appears ("Blending herbs…") so that even those who didn't hear clearly can understand what's

happening. These cues make the interaction more transparent to observers, enhancing group viewing. It effectively externalizes the multimodal commands into visual feedback.

- **Voice in Group Settings:** As mentioned, using voice commands in front of others might be awkward or could trigger unintended activation if someone else speaks. To manage this, the primary user might use voice sparingly in group mode or use an explicit wake word (like saying the system's name before a command). The system could also have a **presentation mode** where voice recognition is either turned off or set to a stricter keyword spotting to avoid picking up background chatter. This prevents any bystander talk from interfering. Moreover, if multiple people are talking, the primary user can default to gesture control (which is unambiguous and visible). Our multimodal setup allows this flexibility – the user isn't forced to use voice if it doesn't suit the social context.

- **Shared Control (if any):** If the secondary users are not just viewers but might interact (say, by giving verbal suggestions or pointing at things for the primary user), we still funnel actual system control through the primary user to avoid confusion. For example, a colleague might say "try blending mint as well!" – the system will not treat that as a command, but the primary user can then do it via their inputs. In future multi-user interactive versions, one could incorporate networked gaze or gestures from multiple people, but that is outside our current scope. Here, **one user is the driver** at a time to maintain clarity.

- **Ergonomics and Safety:** In a group, the primary user might move around to show things. The system's tolerance for movement ensures the experience isn't disrupted. Also, the UI avoids requiring any actions that could be unsafe or overly dramatic in a group (no wide swinging arm gestures, etc.). All gestures are compact (pinches, small arm motions) so the user won't accidentally hit someone nearby – an important practical consideration for AR in shared spaces.

- **Attention Management:** The Donut of Attention context (if we tie back to the theoretical foundations) might suggest that multiple people's focus could be considered, but as our context is mostly single-user operation, we focus on that user's attention. However, if a secondary mode allows the group to see through a shared display, the system might enlarge certain UI elements or use a spectator camera view to make it more understandable to others. Those are presentation details that don't change the input structure, but help avoid confusion in group view.

In summary, the design is **robust to different postures and slight movements** of the primary user – thanks to real-time tracking and adaptive thresholds – and it gracefully **degrades or adjusts in social settings** (mainly by dialing down voice usage and emphasizing clear visual feedback). Multimodal interaction, by its nature, gives the user alternatives: if noise or social context makes voice impractical, the user still has gestures and gaze which are silent and won't be affected by others' presence [20]. The system remains usable and coherent whether the user is quietly working alone or demonstrating to a small group.

By integrating these considerations of modalities, grammar, conflict avoidance, latency, and context, we achieve a fluid multimodal control system. The user can **select**, **blend**, and **pin herbs in real time** using natural gaze and hand motions, augmented by voice for convenience and biofeedback for adaptivity. Our interaction grammar defines a clear mapping and avoids mode conflicts, so the user's intent is accurately captured. And performance tuning ensures the experience feels instant and coherent, maintaining the illusion of direct interaction with the virtual "herbs" as if they were real objects responding to the user's

hands and voice. This design draws on HCI, XR, and multimodal interaction best practices to create an intuitive yet powerful control system [16] [37].

References:

1. Nielsen, J. (1993). *Usability Engineering* – Response time limits for interactive systems [30] [28].
2. Esteves, A. *et al.* (2020). *Comparing selection mechanisms for gaze input in head-mounted displays* – Found dwell selection (gaze hold) faster and more accurate than mid-air gesture or speech triggers [4] [2].
3. Microsoft (2024). *HoloLens 2 Dynamics 365 Guides* – Gaze targeting and air-tap gesture for selecting holograms (gaze selects a button when its dwell cursor fills, and pinch acts like a click) [1] [6].
4. Wang, Z. *et al.* (2021). *Multimodal Interaction in AR (Gaze, Gesture, Speech)* – Noted that hand gestures are intuitive but can cause arm fatigue over long use; speech commands offer control but require recall and increase cognitive load for complex tasks [38]. Combining gaze+gesture+speech yielded higher efficiency than single modalities [39] [21].
5. Lystbæk, M. *et al.* (2022). *Gaze-Hand Alignment in AR Menus* – Demonstrated that gaze-assisted hand input outperforms hands-only input, highlighting the benefit of using gaze for pre-selection before a hand gesture confirm [7] [8].
6. Pfeuffer, K. *et al.* (2025). *PinchCatcher: Multi-Selection for Gaze+Pinch in XR* – Observes that modern AR/MR headsets (HoloLens 2, Meta Quest Pro, etc.) use **gaze plus pinch** as a primary interaction paradigm, leveraging natural gaze targeting and pinch to execute selection [9].
7. Krum, D. *et al.* (2002). *Speech and Gesture Multimodal Control* – Early multimodal interface research: combining speech (verbal commands) and pointing gestures gives greater flexibility and lower error rates, since users can choose the mode or use both for disambiguation [16] [22]. Multimodal inputs can correct each other (mutual disambiguation) and allow usage while moving or at a distance [18] [26].
8. Halbig, A. *et al.* (2021). *Physiological Measurements in VR (Review)* – Reports that biofeedback signals like EEG and heart-rate variability are effective for classifying user arousal/stress, and have been used to adapt VR experiences in real time [23]. This informs our use of EEG/HRV to gauge user state for UI adaptation.
9. Motion-to-Photon Latency – *VR & AR Wiki* (2023): Industry targets <20 ms latency in VR and even ~5–10 ms in AR to avoid perceptible lag [31] [32]. Our system aligns with keeping visual-feedback latency extremely low to maintain immersion.
10. Frontiers HCI Review (2023). *Hand Interaction in Mixed Reality* – Affirms that hand interactions are among the most natural and widely supported modalities in AR, thanks to progress in real-time hand tracking, making virtual object manipulation feel intuitive [10]. We exploit this by using simple pinch/grab gestures that mirror real-world actions.

---

[1] [6] [12] HoloLens 2 gestures (for example, gaze and air tap) for navigating a guide in Dynamics 365 Guides - Dynamics 365 Mixed Reality | Microsoft Learn
https://learn.microsoft.com/en-us/dynamics365/mixed-reality/guides/operator-gestures-hl2

[2] [3] [4] [20] [27] Comparing selection mechanisms for gaze input techniques in head-mounted displays - ScienceDirect
https://www.sciencedirect.com/science/article/abs/pii/S1071581920300185

5 11 19 21 37 38 39 zhimin-wang.github.io
https://zhimin-wang.github.io/publication/thms_2021/pages/pdf/wang21_THMS.pdf

7 8 Gaze-Hand Alignment: Combining Eye Gaze and Mid-Air Pointing for Interacting with Menus in Augmented Reality
https://pure.au.dk/ws/portalfiles/portal/270196599/145_nocopyright_Final_Gaze_Hand_Alignment_ETRA2022.pdf

9 PinchCatcher: Enabling Multi-selection for Gaze+Pinch
https://arxiv.org/html/2503.05456v2

10 Frontiers | Hand interaction designs in mixed and augmented reality head mounted display: a scoping review and classification
https://www.frontiersin.org/journals/virtual-reality/articles/10.3389/frvir.2023.1171230/full

13 14 35 36 What is a hologram? - Mixed Reality | Microsoft Learn
https://learn.microsoft.com/en-us/windows/mixed-reality/discover/hologram

15 Gestures to Place, Move, Rotate, and Scale Multiple Objects in AR ...
https://www.youtube.com/watch?v=5puGSOSmTYc

16 18 22 26 29 people.ict.usc.edu
https://people.ict.usc.edu/~krum/papers/multimodal-vissym.pdf

17 Point and commit with hands - Mixed Reality | Microsoft Learn
https://learn.microsoft.com/en-us/windows/mixed-reality/design/point-and-commit

23 24 25 Frontiers | A Systematic Review of Physiological Measurements, Factors, Methods, and Applications in Virtual Reality
https://www.frontiersin.org/journals/virtual-reality/articles/10.3389/frvir.2021.694567/full

28 30 34 Response Time Limits: Article by Jakob Nielsen - NN/G
https://www.nngroup.com/articles/response-times-3-important-limits/

31 32 Motion-to-photon latency - VR & AR Wiki - Virtual Reality & Augmented Reality Wiki
https://vrarwiki.com/wiki/Motion-to-photon_latency

33 Are 100 ms Fast Enough? Characterizing Latency Perception ...
https://www.researchgate.net/publication/317801603_Are_100_ms_Fast_Enough_Characterizing_Latency_Perception_Thresholds_in_Mouse-Based_Interaction