VR Best Practices Guidelines

https://developer.leapmotion.com/vr-best-practices

Last modified: June 12, 2015 | Version 1.2

Introduction

Hand tracking and virtual reality are both emerging technologies, and combining the two into a fluid and seamless experience can be a real challenge. In both cases, developers need to overturn many longstanding ideas that have served them well for traditional PC setups. It's also an incredible opportunity – the chance to experiment and create in ways that previously existed only in the pages of science fiction. Where the physical and digital worlds collide, there be dragons.

This guide brings together longstanding user experience design principles and VR research with what we've learned from ongoing development and user testing. Like everything in this field, it's just a stepping stone on the path to the Metaverse.

Important: *Experiences of sensory conflict (e.g. inner ear vs. visual cues) and object deformation (due to variations of pupil distance) have a large amount of variability between individuals. Just because it feels fine for you does not mean it will feel fine for everyone! User testing is always essential.*[1]

[1] You don't need a formal testing lab for solid user testing – just a laptop, cell phone camera, Google Hangouts, and the willingness to ask your neighbors to try a project. See more in our blog post The Essentials of Human-Driven UX Design.

Table of Contents

Part 1. User Interface Design

- *The "physical" design of interactive elements in VR should afford particular uses.*
- *Every interactive object should respond to any casual movement.*
- *Clearly describe necessary gestures with text-based prompts.*
- *Interactive elements should be appropriately scaled.*
- *Place text and images on slightly curved concave surfaces*

The Dangers of Intuition

In the world of design, intuition is a dangerous word – highly individual assumptions driven by what we know as familiar. No two people have the same intuitions, but we're trained by our physical world to have triggered responses based on our expectations.[2] The most reliable "intuitive gestures" are ones where we guide users into doing the proper gesture through affordance. When an interaction relies on a user moving in a certain way, or making a specific pose, create affordances to encourage this.[3]

In the real world, we never think twice about using our hands to control objects. We instinctively know how. By thinking about how we understand real-world objects, you can harness the power of hands for digital experiences – bridging the virtual and real worlds in a way that's easy for users to understand.

Semantic and Responsive Gestures

Gestures break down into two categories: semantic and responsive. **Semantic gestures** are based on our ideas about the world. They vary widely from person to person, making them hard to track, but it's important to understand their impact on meaning to the user. For example, pointing at oneself when referring to another person feels foreign.

Responsive gestures occur in response to the ergonomics and affordances of specific objects. They are grounded and specific, making them easy to track. Because there is a limited range of specific interactions, multiple users will perform the exact same gestures.

---

[2] Science fiction also plays a major role in setting cultural expectations for the next generation of real-world interfaces. From *Minority Report* and *Matrix Reloaded* to *The Avengers* and Ender's Game, you can see a recap of many fictional user interfaces in our two-part video, How Do Fictional UIs Influence Today's Motion Controls?

[3] For example, ask two people to first close their eyes. Then ask them to pantomime driving a car. They will likely perform different gestures. However, if you first show them a steering wheel, they will perform similar gestures.

By leveraging a user's understanding of real-world physical interactions, and avoiding gestures that don't make sense in semantic terms, we can inform and guide them in using digital objects. From there, the opportunities to expand a user's understanding of data are endless.

Creating Affordances

In the field of industrial design, "affordances" refers to the physical characteristics of an object that guide the user in using that object. These aspects are related to (but distinct from) the aspects indicating what functions will be performed by the object when operated by the user. Well-designed tools afford their intended operation and negatively afford improper use.[4]

In the context of motion controls, good affordance is critical, since it is necessary that users interact with objects in the expected manner. With 2D Leap Motion applications, this means adapting traditional UX design principles that condensed around the mouse and keyboard. VR opens up the potential to build interactions on more physical affordances. Here are several best practices for designing interactions in VR:

The more physical the response of an object is, the wider the range of interactions may be correctly responded to. For example, a button that can only be pressed by entering a bounding volume from the right direction requires a very clear affordance. Buttons that are essentially physical pistons simply need to look "pushable."[5]

The more specific the interaction, the more specific the affordance should appear. In other words, the right affordance can only be used in one way. This effectively "tricks" the user into making the right movements, and discourage them from making an error. In turn, this makes the UI more efficient and easy to use, and reduces the chances of tracking errors.[6]

Look for real-world affordances that you can reflect in your own projects. There are affordances everywhere, and as we mentioned earlier, these form the basis for our most

---

[4] The classic example is the handle on a teapot, which is designed to be easy to hold, and exists to prevent people from scorching their fingers. (Conversely, the spout is *not* designed to appear grabbable.) In computer interface design, an indentation around a button is an affordance that indicates that it can be moved independently of its surroundings by pressing. The color, shape, and label of a button advertise its function. Learn more in our post What Do VR Interfaces and Teapots Have in Common?

[5] Learn more about button design with our infographic Build-a-Button Workshop: VR Interaction Design from the Ground Up.

[6] For example, a dimensional button indicates to the user to interact with the affordance by pushing it in (just like a real world button). If the button moves expectedly inward when the user presses the button, the user knows they have successfully completed the interaction. If however, the button does not move on press, the user will think they haven't completed the interaction.

commonly held intuitions about how the world works:

- *Doorknobs and push bars.* Doorknobs fit comfortably in the palm of your hand, while push bars have wide surfaces made for pressing against. Even without thinking, you know to twist a doorknob.

- *Skateboard prevention measures.* Ever seen small stubs along outdoor railings? These are nearly invisible to anyone who doesn't want to grind down the rail – but skaters will find them irritating and go elsewhere.

- *Mouse buttons vs. touch buttons.* Mouse-activated buttons look small enough to be hit with your mouse, while touchscreen buttons are big enough to be hit with your finger.

Everything Should Be Reactive

Every interactive object should respond to any casual movement. For example, if something is a button, any casual touch should provoke movement, even if that movement does not result in the button being fully pushed. When this happens, the kinetic response of the object coincides with a mental model, allowing people to move their muscles to interact with objects.[7]

Because the Leap Motion Controller affords no tactile responses, it's important to use other cues to show that users have interacted with a UI element. For example, when designing a button:

- use a shadow from the hand to indicate where the user's hand is in relation to button
- create a glow from the button that can be reflected on the hand to help understand the depth relationship
- ensure the button moves in relationship to the amount of pressure (Z-press) from the user
- use sound to indicate when the button has been pressed ("click")
- create a specific hover state and behavior for interaction widgets

When done effectively, people will feel themselves anticipating tactile experiences as they interact with a scene. However, if an object appears intangible, people have no mental model for it, and will not be as able to reliably interact with it.

[7] For a real-world example, we can ask people to close their eyes just before catching objects. People can learn to catch based on extrapolated motion fairly reliably.

Gesture Descriptions

While affordances are important, text- or audio-based tutorial prompts can also be essential for first-time users. Be sure to clearly describe intended interactions, as this will greatly impact how the user does the interaction.

For example, if you say “pump your fist,” the user could interpret this in many different ways. However, if you say “fully open your hand, then make a fist, and repeat,” the user will fully open their hand then fully close it, repeatedly. Be as specific as possible to get the best results, using motions and gestures that track reliably.

Alternatively, you can use our Serialization API to record and playback hand interactions. For an example of how this works, take a look at our post on designing *Playground*, which is open sourced on GitHub.

Interactive Element Targeting

Appropriate scaling. Interactive elements should be scaled appropriate to the expected interaction (e.g. full hand or single finger). One finger target should be no smaller than 20 mm in real-world size. This ensures the user can accurately hit the target without accidentally triggering targets next to it.

Limit unintended interactions. Depending on the nature of the interface, the first object of a group to be touched can momentarily lock out all others. Be sure to space out UI elements so that users don’t accidentally trigger multiple elements.

Limit hand interactivity. Make a single element of the hand able to interact with buttons and other UI elements – typically, the tip of the index finger. Conversely, other nearby elements within the scene should not be interactive.

Text and Image Legibility

While VR is amazing at many things, there are sometimes issues involved with rendering text. Due to resolution limitations, only text at the center of your FOV may appear clear, while text along the periphery may seem blurry unless users turn to view it directly. For this reason, be sure to avoid long lines of text in favour of scrollable columns like our Scroll Widget.

Another issue arises from lens distortion. As a user's eyes scan across lines of text, the positions of the pupils will change, which may cause distortion and blurring.[8] Furthermore, if the distance to the text varies – which would be caused, for example, by text on a flat laterally extensive surface close to the user – then the focus of the user's eyes will change, which can also cause distortion and blurring.

The simplest way to avoid this problem is to limit the angular range of text to be close to the center of the user's field of view (e.g. making text appear on a surface only when a user is looking directly at the surface). This will significantly improve readability.

[8] See, for example, https://www.youtube.com/watch?v=lsKuGUYXHa4.

Part 2. Interaction Design

- *No movement should take place unless it's user-driven.*
- *Keep in mind that human hands naturally move in arcs, rather than straight lines.*
- *Limit the number of gestures that users are required to learn.*
- *All interactions should have a distinct initiation and completion state.*
- *Ensure that users can interact with objects occluded by their hands.*

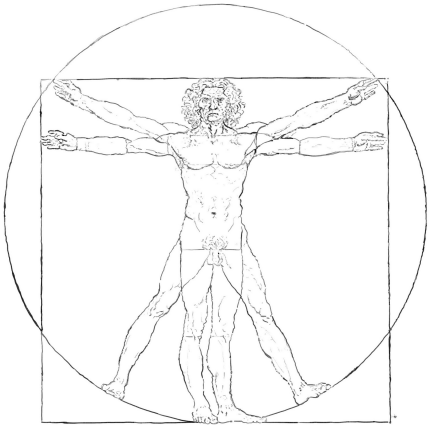
Restrict Motions to Interaction

One of the biggest barriers to VR is simulator sickness, which is caused by a conflict between different sensory inputs (i.e. inner ear, visual field, and bodily position). Oculus' best practices guide covers this issue in great detail. Generally, significant movement – as in the room moving, rather than a single object – that hasn't been instigated by the user can trigger feelings of nausea. Conversely, being able to control movement reduces the experience of motion sickness.

- The display should respond to the user's movements at all times, without exception. Even in menus, when the game is paused, or during cutscenes, users should be able to look around.
- Do not instigate any movement without user input (including changing head orientation, translation of view, or field of view).
- Do not rotate or move the horizon line or other large components of the environment unless it corresponds with the user's real-world motions.
- Reduce neck strain with experiences that reward (but don't require) a significant degree of looking around. Try to restrict movement in the periphery.
- Ensure that the virtual cameras rotate and move in a manner consistent with head and body movements. (See *Part 5: Space and Perspective* for more details about how to achieve this with the Leap Motion Controller.)

Ergonomics

Designing based on how the human body works is an essential to bringing any new interface to life. Our bodies tend to move in arcs, rather than straight lines, so it's important to compensate by allowing for arcs in 3D space. For example, when making a hand movement in Z, the user inherently makes a Y movement as well. The



same is true when the user moves along the X axis – this motion will result in the user also creating a z movement.

Hand movements can also vary greatly based on posture. For example, when tracking the index finger:

- Pivoting on a fixed shoulder with elbow raised: wandering in the X and Y axes.
- Pivoting on a fixed elbow on a table: wandering in Y
- Movement pivoting on wrist with relaxing index: minimum wandering, very small Z depth

For more general ergonomics guidelines, be sure to consult our post Taking Motion Control Ergonomics Beyond Minority Report along with our latest ergonomics and safety guidelines.

Limit Learned Gestures

There are a very limited number of gestures that users can remember.[9] When developing for motion controls, be sure to build on a base set, or try combining gestures for more advanced features. Even better, create specific affordances that users will respond to, rather than having to learn a specific pose or movement.

Eliminate Ambiguity

All interactions should have a distinct initiation and completion state. The more ambiguous the start and stop, the more likely that users will do it incorrectly:

- Clearly describe intended poses and where the user should hold their hand to do that pose.
- If the intended interaction is a motion, make a clear indicator where the user can start and stop the motion.
- If the interaction is in response to an object, make it clear from the size and shape of the object how to start and stop the interaction.

Hand-Occluded Objects

In the real world, people routinely interact with objects that are obscured by their hands. Normally, this is achieved by using touch to provide feedback. In the absence of touch, here are some techniques that you can use:

[9] Most studies put this number at 5. For example, touchpads on Apple computers recognize 1 to 4 fingers. These poses are used to distinguish the functionality of motions.

- Provide audio cues to indicate when an interaction is taking place.
- Make the user's hand semi-transparent when near UI elements.
- Make objects large enough to be seen around the user's hand and fingers. (See the section *Interactive Element Targeting* for more information about scaling.)
- Avoid placing objects too high in the scene, as this forces users to raise their hands up and block their view. (See the section *Ideal Height Range* for more information.)
- When designing hand interactions, consider the user's perspective by looking at your hands with a VR headset.

Locomotion

World navigation is one of the greatest challenges in VR, and there are no truly seamless solutions beyond actually walking around in a Holodeck-style space.[10] Generally, the best VR applications that use Leap Motion for navigation aren't centered around users "walking" around in a non-physical way, but transitioning between different states. Here are some interesting experiments in locomotion:[11]

- ***World of Comenius.*** This educational application features glowing orbs that can be tapped to travel from place to place.

- ***Fragmental.*** From the developer of the Hovercast VR menu system, *Fragmental*'s navigation controls let you revolve around a central axis.

- ***VR Intro* and *Weightless.*** Two-handed flight has been a cultural touchstone since the days of George Reeves' Superman. This is a great way to give your users superpowers, but it can get tiring unless used in a short demo, or alongside other interactive elements.

- ***Planetarium.*** The Joyball widget makes it easy to move with small displacements from a comfortable posture, while also providing compass data that helps to orient the user.

- **Three.js Camera Controls.** This set of experiments from Isaac Cohen explores several different ways that users can navigate 3D space.

[10] Oculus' best practices guide is an excellent resource to avoid simulator sickness due to motion and acceleration. One key consideration is to keep acceleration (i.e. changes in speed) as short and infrequent as possible.
[11] These are explored in greater detail in our blog post 5 Experiments on the Bleeding Edge of VR Locomotion.

Sound Effects

Sound is an essential aspect of truly immersive VR. Combined with hand tracking and visual feedback, it can be used to create the “illusion” of tactile sensation.[12] It can also be very effective in communicating the success or failure of interactions.

[12] For instance, button clicks on touchscreen keyboards help users become confident treating the keys as independent interactive objects. In particular, this provides useful feedback on interactions in the periphery of the user’s FOV.

Part 3. Optimizing for VR Tracking

- *Include safe poses to avoid "the Midas touch" where everything is interactive.*
- *Encourage users to keep their fingers splayed out.*
- *Keep interactive elements in the "Goldilocks zone" between desk height and eye level.*
- *Filter out implausible hands.*
- *Use visual feedback to encourage users to keep their hands within the tracking zone.*
- *Avoid interactions involving fingers that would be invisible to the controller.*

The Sensor is Always On

As we've discussed elsewhere, The Leap Motion Controller exhibits the "live-mic" or "Midas touch" problem.[13] This means that your demo must have a safe pose, so that users can safely move through the device's field of view without interacting. Gesture-based interactions should be initiated with specific gestures that are rarely a part of casual movement (e.g. making a fist and then splaying the fingers).

However, safety should never be at the expense of speed. Do not require a pause to begin an interaction, as your users will become frustrated.

Flat Splayed Hands

Whenever possible, encourage users to keep their fingers splayed and hands perpendicular to the field of view. This is by far the most reliable tracking pose. You can encourage this by requiring interactions to be initiated from this pose, and providing positive indicators when the pose is detected.[14]

Ideal Height Range

Interactive elements within your scene should typically rest in the "Goldilocks zone" between desk height and eye level. Here's what you need to consider beyond the Goldilocks zone:

Desk height. Be careful about putting interactive elements at desk height or below. Because there are often numerous infrared-reflective objects at that height, this can cause poor

[13] The design challenges outlined in Part 3 are explored in our blog post 4 Design Problems for VR Tracking (And How to Solve Them).
[14] For example, the Image Hands assets for Unity achieve this by displaying a bright blue glow when tracking confidence is high.

tracking. (For example, light-colored or glossy desks will overexpose the controller's cameras.)

Above eye level. Interactive objects that are above eye level in a scene can cause neck strain and "gorilla arm." Users may also occlude the objects with their own hand when they try to use them. See *Hand-Occluded Objects* in Part 2 for design techniques that can compensate for this.

Implausible Hands

The Leap Motion API returns some tracking data that can be safely discarded, depending on the use case. For head-mounted devices, hands may be detected that are entirely implausible given a head-mounted device. Use the Confidence API to eliminate hands with low confidence values. Allow a maximum of one right and one left hand, and only animate those two hands.

Edge of FOV

Avoid interactions that require a hand to be at rest when near the edge of the field of view, or when horizontal. Both of these configurations result in spontaneous hand movement. To help resolve FOV issues, use the confidence API and filter held objects.

Hand Out of View

If the user can't see their hand, they can't use it. While this might seem obvious to developers, it isn't always to users – especially when focused on the object they're trying to manipulate, rather than looking at their hand. Here are a few techniques to make it clear to users that they must keep their hand in view at all times:

- **Disappearing skins.** Create an overlay for the hand, such as a robot hand, spacesuit, or alien skin, or a glow that envelops an image of the passthrough hand (which is available through the Image Hands asset).[15] When tracking is reliable, the overlay appears at full opacity. When Confidence API values drop, the overlay fades out or disappears. This affordance can be used to let the user know when the tracking sees the hand, and when it doesn't.

- **Error zone.** Delineate a clear safety zone to indicate where the hands should be placed. You can notify the user when their hands enter (or exit) the zone with a simple

[15] For some insight into the technical and design challenges for designing rigged hands for Leap Motion tracking, see our blog post 6 Design Challenges for Onscreen Hands.

change in color or opacity to that area of the screen.

- **Contextually correct persistent states.** If the user grabs something, that thing should remain in their hand until they release it. If their hand leaves the field of view, the object should also smoothly exit.

- **Failure vs. exit.** It's possible to distinguish tracking failures (when the hand abruptly vanishes from the center of the field of view) from tracked exiting (when the hand vanishes at or near the edge of the field of view). Your application should handle these in a way that yields smooth and plausible motion, without causing unintended interactions.

Finger Occlusion

As with any optical tracking platform, it's important to avoid the known unknowns. The Leap Motion tracking software includes "target poses" to which it will default in the absence of direct line-of-sight data. Thus, it is possible to correctly identify the poses of fingers that are outside of the line of sight of the device. However, this pose may not be reliably distinguished from other poses.

As a result, be sure to avoid interactions that depend on the position of fingers when they are out of the device's line of sight. For example, if pinching is allowed, it should only be possible when the fingers can be clearly seen by the controller. Similarly, grabbing is tracked most reliably when the palm faces away from the device.

Part 4. Hands and Body

- *Create or select virtual hands that are appropriate to the setting.*
- *Virtual hand positions should match real-world positions as closely as possible.*
- *Only render the user's body in situations where you can expect reasonable agreement. While being disembodied is disconcerting, having multiple inconsistent bodies is usually worse.*

Choosing Your Hands

For Unity projects, we strongly recommend using the Image Hands assets for your virtual hands. This asset combines the absolute realism of live video, with the full interactivity and 3D behavior of our classic rigged hands, by projecting the raw images of your hands into a 3D mesh that can interact with other objects in real-world space. The part of the mesh that you can see is covered by the passthrough from the twin Leap Motion cameras. Each camera provides a separate view of your hands, so that what you see has actual depth.

This hybrid effect has a powerful impact in VR because your real hands can now actually pass through (or disappear behind) virtual objects. The hands interact properly with other 3D objects because they are 3D – complete with the interactive and visual capabilities that you expect. This approach also reduces jitter effects, since there's no longer an artificially generated rigged hand to worry about. While the hand image in VR might shift or become transparent, it won't suddenly shift in the wrong direction.[16]

Hand Position

Just as accurate head tracking can place you inside a virtual world, hand tracking can reinforce the sense that you've actually traveled to another place. Conversely, when hands appear in the wrong spot, it can be very disorienting. To enhance immersion and help users rely on their proprioception to control movements and understand depth, it's essential that virtual hand positions match their real-world counterparts as much as possible.

[16] Alternatively, we've also created many photorealistic hands across skin colors and genders, as well as robot hands, minimal hands, and other models for Unity. These have proven to be popular with developers who want to enhance the user's sense of immersion in sci-fi or minimalist experiences.

Body Position

By providing a bodily relationship to the elements in the scene, you can increase immersion and help ground your user. Only create an avatar for the user if they are likely to be in alignment. For example, if the user is sitting in a chair in a game, you can expect that they will do the same in real life. On the other hand, if the user is moving around a scene in a game, they are unlikely to be moving in real life.

People are usually comfortable with a lack of a body, due to experiences with disembodied observation (movies) and due to the minimal presence of one's own body in one's normal field of view (standing looking forward). However, adding a second body that moves independently is unfamiliar and at odds with the user's proprioception.

Part 5. Space and Perspective

- *Adjust the scale of the hands to match your game environment.*
- *Align the real-world and virtual cameras for the best user experience.*
- *Use 3D cinematic tricks to create and reinforce a sense of depth.*
- *Objects rendered closer than 75 cm (within reach) may cause discomfort to some users due to the disparity between monocular lens focus and binocular aim.*
- *Use the appropriate frame of reference for virtual objects and UI elements.*
- *Use parallax, lighting, texture, and other cues to communicate depth and space.*

Controller Position and Rotation

To bring Leap Motion tracking into a VR experience, you'll need to use (or create) a virtual controller within the scene that's attached to your VR headset's camera objects.[17] In this section, we'll use the Oculus Rift Unity plugin as an example, but this approach can be applied to other headsets as well.

By default, the two Oculus cameras are separated by 0.064,[18] which is the average distance between human eyes. Previously, to center the controller, you would need to offset it by 0.032 along the x-axis. Since the Leap Motion Controller in the physical world is approximately 8 cm away from your real eyes, you'll also need to offset it by 0.08 in the z-axis.

However, Oculus 0.4.3.1B+ includes a "CenterEyeAnchor" which resides between LeftEyeAnchor and RightEyeAnchor. This means that you can attach the HandController object directly to the CenterEyeAnchor with no offset along the x axis. Assuming that HandController is a child of CenterEyeAnchor, there are two possible XYZ configurations you might want to use:

- *To match passthrough:* 0.0, 0.0, 0.0 and scale according to the augmented reality case of the section below.[19]
- *To match real-life:* 0.0, 0.0, 0.08.

Note that objects in the passthrough video will also appear slightly closer because of the eyestalk effect,[20] which is one of many reasons why we're working on modules designed to

[17] In our Unity assets, this is the HandController. For more specific details about how the Unity asset works, see our blog post Getting Started with Unity and the Oculus Rift.
[18] 6.4 cm in real-world length. Note that distances in Unity are measured in meters.
[19] As of v2.2.4 this can also be obtained dynamically via Leap::Device::baseline().
[20] This results from seeing a direct video feed from cameras that are elevated slightly away from your face. For example, the eyestalk effect with the Leap Motion Controller and Oculus Rift is 80 mm.

be embedded in VR headsets. For more information on placing hand and finger positions into world space, see our guide to VR Essentials: What You Need to Build from Scratch.

Scale: Augmented vs. Virtual Reality

For VR experiences, we recommend a 1:1 scale to make virtual hands and objects look as realistic and natural as possible. With augmented reality, however, the scale needs to be adjusted to match the images from the controller to human eyes. Appropriate scaling can be accomplished by moving the cameras in the scene to their correct position, thereby increasing the scale of all virtual objects.[21]

This issue occurs because the Leap Motion Controller sensors are 40 mm apart, while average human eyes are 64 mm apart. Future Leap Motion modules designed for virtual reality will have interpupillary distances of 64mm, eliminating the scaling problem.

Positioning the Video Passthrough

When using the Image API, it's important to remember that the video data doesn't "occupy" the 3D scene, but represents a stream from outside. As a result, since the images represent the entire view from a certain vantage point, rather than a particular object, they should not undergo the world transformations that other 3D objects do. Instead, the image location must remain locked regardless of how your head is tilted, even as its contents change – following your head movements to mirror what you'd see in real life.

To implement video passthrough, skip the modelview matrix transform (by setting the modelview to the identity matrix) and use the projection transform only.[22] Under this projection matrix, it's sufficient to define a rectangle with the coordinates (-4, -4, -1), (4, -4, -1), (-4, 4, -1), and (4, 4, -1), which can be in any units. Then, texture it with the fragment shader provided in our Images documentation.[23]

Depth Cues

Whether you're using a standard monitor or VR headset, the depth of nearby objects can be difficult to judge. This is because, in the real world, your eyes dynamically assess the depth

[21] In the case of the Oculus Rift, this is ±½ of Leap::Device::baseline() in the x direction, e.g. (+0.02, 0, 0), (-0.02, 0, 0).
[22] In the case of the Oculus SDK, the perspective projection returned by the Oculus API.
[23] In Unity and Unreal Engine, this means locking the images as a child of the Controller object, so that it stays locked in relation to the virtual Leap Motion Controller within the scene. Our Unity image passthrough example does this for you, as does the unofficial Unreal plugin v0.9.

of nearby objects – flexing and changing their lenses, depending on how near or far the object is in space. With headsets like the Oculus Rift, the user's eye lenses will remain focused at infinity.[24]

When designing VR experiences, you can use 3D cinematic tricks to create and reinforce a sense of depth:

- objects in the distance lose contrast
- distant objects appear fuzzy and blue/gray (or transparent)
- nearby objects appear sharp and full color/contrast
- shadow from hand casts onto objects, especially drop-shadows
- reflection on hand from objects
- sound can create a sense of depth

Rendering Distance

The distance at which objects can be rendered will depend on the optics of the VR headset being used. (For instance, Oculus recommends a minimum range of 75 cm for the Rift DK2.) Since this is beyond the optimal Leap Motion tracking range, you'll want to make interactive objects appear within reach, or respond to reach within the optimal tracking range of the Leap Motion device.

Virtual Safety Goggles

As human beings, we've evolved very strong fear responses to protect ourselves from objects flying at our eyes. Along with rendering interactive objects no closer than the minimum recommended distance, you may need to impose additional measures to ensure that objects never get too close to the viewer's eyes.[25] The effect is to create a shield that pushes all moveable objects away from the user.

Multiple Frames Of Reference

Within any game engine, interactive elements are typically stationary with respect to some frame of reference. Depending on the context, you have many different options:

[24] This issue is even more pronounced with a two-dimensional screen, where there is no fundamental perception of depth – instead, awareness of depth must be created using visual queues in a single flat image.
[25] In Unity, for instance, one approach is to set the camera's near clip plane to be roughly 10 cm out.

- *World frame of reference:* object is stationary in the world.[26]
- *User body frame of reference:* object moves with the user, but does not follow head or hands.[27]
- *Head frame of reference:* object maintains position in the user's field of view.
- *Hand frame of reference:* object is held in the user's hand.

Be sure to keep these frames of reference in mind when planning how the scene, different interactive elements, and the user's hands will all relate to each other.

Parallax, Lighting, and Texture

Real-world perceptual cues are always useful in helping the user orientate and navigate their environment. Lighting, texture, parallax (the way objects appear to move in relation to each other when the user moves), and other visual features are crucial in conveying depth and space to the user.

[26] This is often the best way to position objects, because they exist independently from where you're seeing it, or which direction you're looking. This allows an object's physical interactions, including its velocity and acceleration, to be computed much more easily. However, hands tracked by a head-mounted Leap Motion Controller require an additional step – because the sensor moves with your head, the position of your hands depends on the vantage point. For more information on how to place the hand and finger positions tracked by the controller into world-space, see our blog post VR Essentials: What You Need to Build from Scratch.

[27] It's important to note that the Oculus head tracking moves the user's head, but not the user's virtual body. For this reason, an object mapped to the body frame of reference will not follow the head movement, and may disappear from the field of view when the user turns around.

Appendix A. Optimizing for Latency

Needless to say, no latency is good latency. And considering the motion sickness factor, reducing latency to an absolute minimum is crucial for a compelling VR experience. Here are some insights on our platform's motion-to-photon latency.

As with any optical tracking solution, the Leap Motion Controller's latency is determined by the camera framerate, plus the computation time of the hand tracking. Currently, this amounts to around 8 ms for the camera, plus another 4 ms for our tracking algorithms.[28] That's 12 ms overall with today's peripheral. Upcoming generations of embedded Leap Motion VR technology will have much higher framerates, cutting the latency down further.

Of course, motion tracking is just one part of the system's total end-to-end latency. VR displays themselves also add to overall latency – one functioning at 90 Hz will add least another 11 ms, while the GPU will add at least another frame's worth.[29] Adding the three together, we have a total of >35 ms.

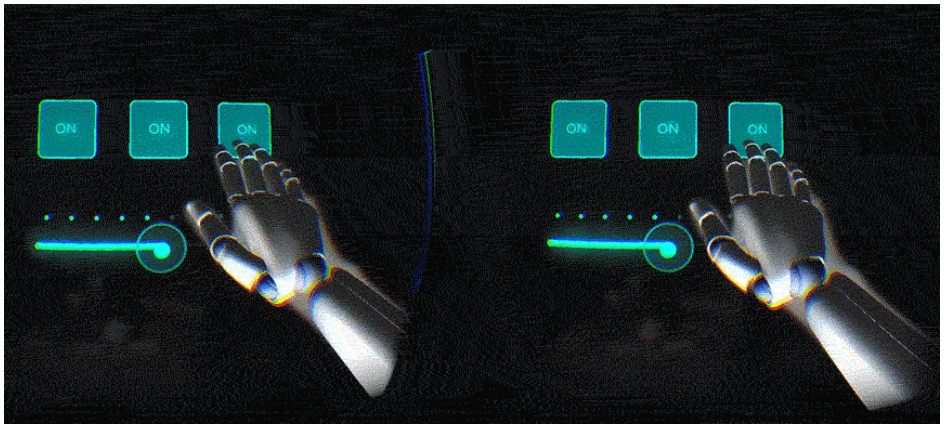
At this point, if your game engine is running in a separate thread at a minimum of 120 frames per second, then it can roughly keep pace with the tracking data, and you have effectively reached optimal motion-to-photon latency.[30]

[28] 1/120 second = ~8 ms.
[29] 1/75 second = ~11 ms.
[30] Of course, this only applies to the Leap Motion hand tracking data. There is evidence that while <20 ms is needed for optimal head tracking, higher values can still provide a positive user experience for hands.

Appendix B. Designing the Unity Widgets

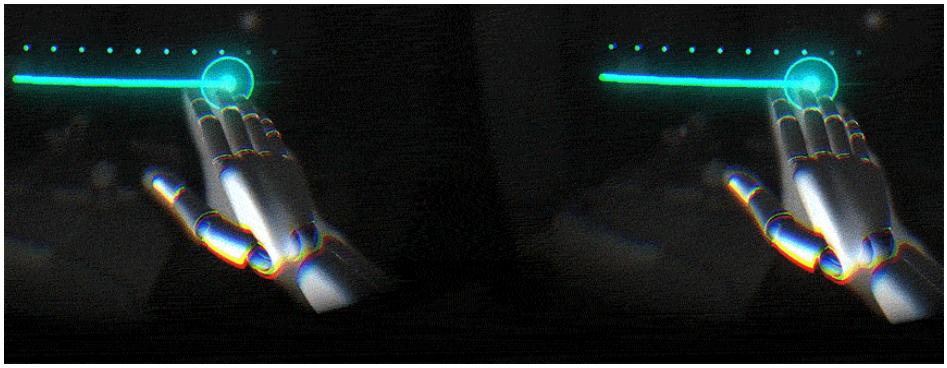
The design process behind our Unity Widgets was driven by many of these best practices, and in turn informed new ones. While you can get a more in-depth perspective on our blog's Developer Diaries series,[31] here's a quick analysis of each Widget:

Button



- Appears as a discrete element that can easily be distinguished as interactive.
- Size and spacing makes triggering the button easy, and reduces the chances of accidentally triggering neighboring buttons.
- Buttons compress in z-space until they are triggered and light up.

Slider



- The slider is highlighted when activated.

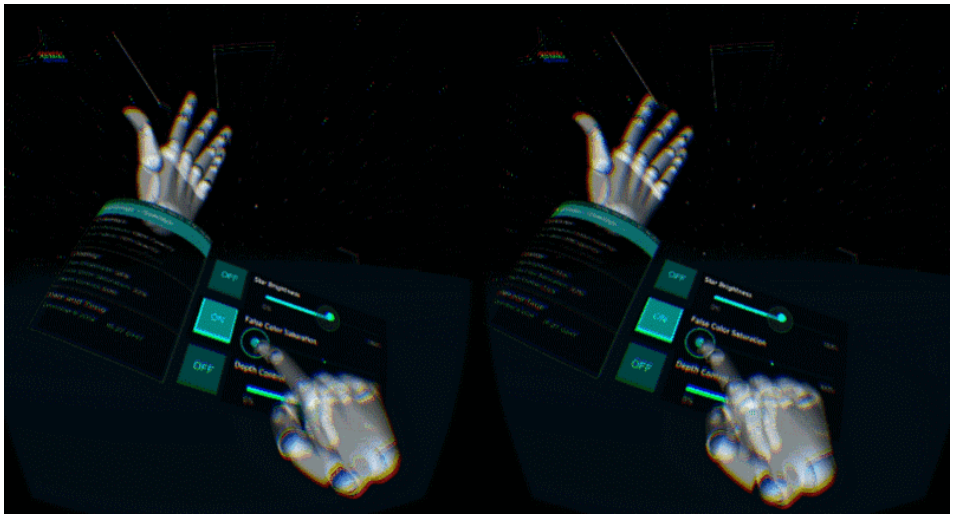
[31] See Introducing Widgets: Fundamental UI Elements for Unity, The Evolution of Arm HUD, A Brief History of Time Dial, and Traveling Around the Globe (and Under the Sky) in *Planetarium*.

- Indicator lights above the slider change color to create an additional affordance.

Scroll

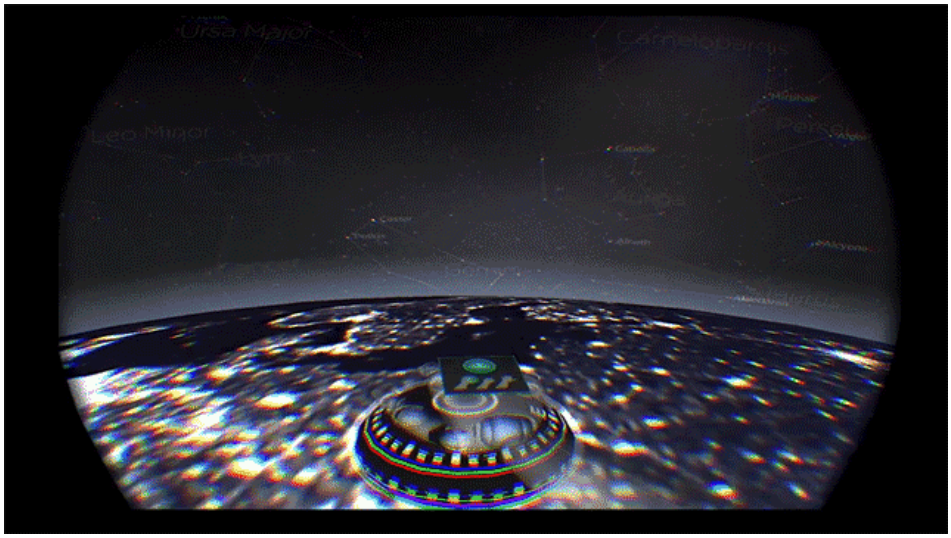
- Users can move the content directly rather than attempting to target a small, mouse-style scrollbar.
- The scrollbar within the widget indicates your position within the accessible content.

Arm HUD



- Uses a layout that encourages optimal tracking, so that fingers don't blend into the arms under infrared.
- Specific initialization states (turning the arm) ensure that it doesn't clutter up valuable VR real estate.

Joyball



- The user always has direct control over their motion, preventing motion sickness. When the control is not in use, movement stops immediately.
- Our approach to the interaction pose changed after user testing revealed that it was uncomfortable. Good ergonomics is essential!
- The Joyball Widget is triggered when hands are between the user and the globe – line of sight is a very useful indicator for determining what the user wants to interact with.
- Visual feedback ensures that users know how their actions affect the Widget.

Appendix C. Thinking and Browsing in 3D Space

By UX designer Jody Medich

Many parts of the human brain contribute spatial information to a constantly evolving mental map of our surroundings. This spatial memory enables us to understand where one object is in relation to another, how to navigate through the world, and provides shortcuts through spatial cognition.

With spatial cognition, humans free up more working memory[32] or short term memory – the faculty that provides temporary storage and processing power for the task at hand.

Why Space?

As three-dimensional creatures, humans need space to think. Space supports human cognitive abilities in a number of ways[33], allowing us to quickly access memories, reveal relationships, and think more effectively:

1. **Spatial Semantics.** Physical space allows users to spatially arrange objects in order to make sense of data and its meaning, thereby revealing relationships and making connections. Imagine a furious ideation sticky-note session. As participants add data to the wall, sticky notes appear in thematic groupings spatially across the board. Up close, we can see the individual interrelated data points. From a step back, we gain perspective on the overall structure of information. The way the space is organized provides the semantic structure we need to make sense of the information. This is true for sticky notes as well as for our rooms, our homes, our cities, and the world at large.
2. **External Memory.** Allowing space for external memory compensates for humans' limited working memory, allowing people to see more detail and to keep information accessible and visually available. The note to buy milk on the fridge, the family photos stuck in the mirror, and putting "must remember" items near the car keys are all examples of spatial external memory.
3. **Dimension.** Without thinking, we can immediately tell the difference between two objects based on dimension and other cues. Through their dimensionality, we can

[32] A. Baddeley, "Working Memory," Science Vol. 255, No. 5044, 556-559 (31 January 1992).
[33] Christopher Andrews, Alex Endert, and Chris North, "Space to Think: Large High-Resolution Displays for Sensemaking," Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 2010 Proceedings, 55-64 (2010).

innately understand information about either object without having to use much working memory in the process.

Problem: 2D Computing is Flat

With modern operating systems, interaction designers create shells based on a “magic piece of paper” metaphor. Essentially, this means that the OS works like a series of 2D planes that switch, slide, or blend into each other.

Unfortunately, this creates a very limited sense of space and effectively prevents the development of spatial cognition. While smartphones and tablets have made attempts at spatial organization systems with “carousels,” the map space is limited and does not allow for productivity scenarios. For instance, I cannot work on large presentations or content creation on a tablet, as the OS is not extensible to those types of tasks.

Contemporary desktop shells are even more nebulous and do not provide opportunities for spatial cognition – forcing users to spend working memory on menial tasks. Organization is chiefly based on filenames rather than spatial semantics, while properties are mapped only in one dimension at a time. This makes it impossible to tell the difference between items based on multiple dimensions, and severely limits opportunities to visually sort, remember, and access data.

In practice, this complete lack of spatial mapping demands cognitive-heavy task switching from users. Because there is no spatial memory – no spatial cognition of the digital space – the user must expend their precious working memory. It is up to the user to understand how the data has been structured and how to retrieve it. Each user must develop workarounds to quickly access files and move between seemingly related tasks (e.g. alt-tab, naming conventions, etc.).

As a result, *every* interaction with the OS is an interruption, often requiring many traversals to achieve a goal. These include:

- Launching a new app
- Closing an app to move to another activity
- Finding an item
- Accessing the file browser
- Changing windows across apps
- Actions that cause a new window/screen in an app
- Notifications/conversations

These interruptions are extremely costly to productivity and flow[34]. Throughout the workday, the average user switches tasks three times per minute[35], and once distracted, it may take anywhere from 30 seconds to half an hour to resume the original task[36]. If every OS interaction represents an interruption, whether great or small, imagine how much collective time is lost to overcoming technological interfaces.

Opportunity: Bringing Spatial Cognition into VR

Based on Hick's Law (1952), any interface is vastly improved through the reduction of the number of choices, thereby improving the signal-to-noise ratio [expressed as T = blog2(n + 1)]. Likewise, reducing traversal time between objects will naturally improve efficiency (Fitt's Law)[37]. With the rise of augmented and virtual reality (AR/VR), this can finally be achieved by providing opportunities for spatial cognition.

AR/VR is inherently spatial, offering a much larger and richer surface for the spatial arrangement of tasks. And spatial memory is free – even in virtual worlds[38].

Even now, we are seeing marked increased productivity on larger screens, which allow users to spatially arrange tasks[39]. Czerwinski et al. demonstrated that spatial tasks were significantly improved for women on displays with large fields of view, with AR/VR providing the ultimate open space[40].

---

[34] S.M. Shamim, Md. Aminul Islam, and Md. Arshad Hossain, "A Study on Unplanned Interruptions in Software Development," https://novicearshad.wordpress.com/2012/01/24/unplanned-interruptions-in-software-development/ (24 January 2012).
[35] Dugald Ralph Hutchings, Greg Smith, Brian Meyers, Mary Czerwinski, and George Robertson, "Display Space Usage and Window Management Operation Comparisons between Single Monitor and Multiple Monitor Users," AVI 2004 Conference, 25-28 May 2004, Gallipoli, Italy, 32-39 (2004). Available from http://research.microsoft.com/en-us/um/redmond/groups/cue/publications/AVI2004-DisplaySpace.pdf.
[36] Maggie Jackson, "Fighting a War Against Distraction," excerpt from *Distracted: The Erosion of Attention and the Coming Dark Age*, available from http://www.nytimes.com/2008/06/22/jobs/22shifting.html (22 June 2008).
[37] Paul M. Fitts, "The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement," Journal of Experimental Psychology Vol. 47, No. 6, 381-391 (June 1954).
[38] Eric A. Johnson, "A Study of the Effects of Immersion on Short-Term Spatial Memory," College of Technology Masters Theses, Paper 30, available from http://docs.lib.purdue.edu/techmasters/30 (2010).
[39] NEC, "Monitor Size and Aspect Ratio Productivity Research: A Comparison of Single and Dual Traditional Aspect Displays with a Widescreen Display over Productivity," http://spectraview.nec.com.au/wpdata/files/25.pdf. Also see Mary Czerwinski, Greg Smith, Tim Regan, Brian Meyers, George Robertson, and Gary Starkweather, "Toward Characterizing the Productivity Benefits of Very Large Displays," Human-Computer Interaction – INTERACT'03, 9-16 (2003).
[40] Mary Czerwinski, Desney S. Tan, and George G. Robertson, "Women Take a Wider View," CHI 2002 Conference, April 20-25, 2002. Available from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.18.2458&rep=rep1&type=pdf.

In general, the more space users have available, the more windows and tabs they can open; multi-tasking with a turn of the head rather than a cognitively heavy interaction with the OS. As Andrews et al. point out, “comparisons can be done visually, rather than relying on memory and imperfect internal models” [2]. Meanwhile, Ball et al. proved that physical interaction further improved the way users understand and use virtual space, just as it does in real environments[41].

So how do we accomplish this? Let us start by building on a digital property that already has an underlying spatial system: the browser.

Desktop Space and the Browser

The modern Internet browser is a digital task-switching haven, designed to allow users to access and explore vast amounts of content. For that reason, it already has a baseline spatial structure, built on the tab (resembling a card) and window (deck of cards).

Spatial Semantics
Users combine tabs spatially, grouping like tabs into windows. Within groupings, certain tabs, such as search engines, social networks, and content providers, act as launchers for new tabs. These tabs load to the right of the launcher tab, but before the next launcher tab – creating a spatial structure from left to right, with the tab generators as landmarks. The resulting spatial map provides a sort of timeline, and a method for keeping track of content, as tabs allow users to:

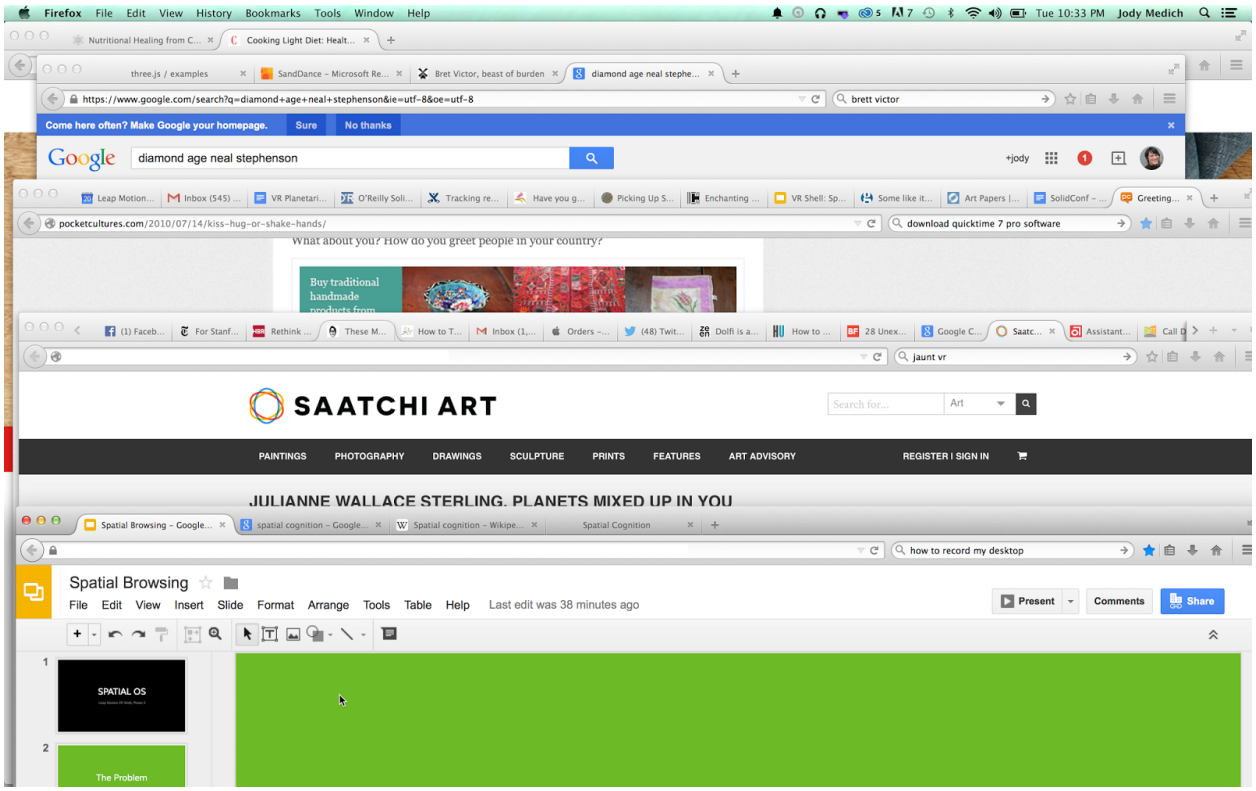
- spatially arrange their content
- put aside a piece of content to revisit later
- set reminders for necessary tasks/activities
- keep their place in a document, even when branching from the initial window
- engage in parallel browsing across multiple tabs while maintaining multiple back stacks (each tab has its own history)
- group similar tasks and tabs for sub-tasks (e.g. one window with multiple social networks or emails open)
- leave page open for a long time over multiple sessions with the intention of returning to them.
- use greater screen space to open more tabs.

The tab was a major step forward in the evolution of the browser, largely replacing the Back button and opening up new possibilities for content exploration. This is because, unlike abstract pages lurking in the browser history, tabs have spatial presence:

[41] Robert Ball, Chris North, and Doug A. Bowman, “Move to Improve: Promoting Physical Navigation to Increase User Performance with Large Displays,” CHI 2007 Proceedings, 28 April – May 3, 2007, San Jose, CA, 191-200 (2007).

- The back button can require too many (or an unknown number) of clicks to return to a desired page.
- While an open tab maintains state, the back button requires the page to reload.
- The browser history (from right-clicking on the Back button) requires users to navigate via link name, while tabs allow users to navigate via spatial relationship or visual browsing.

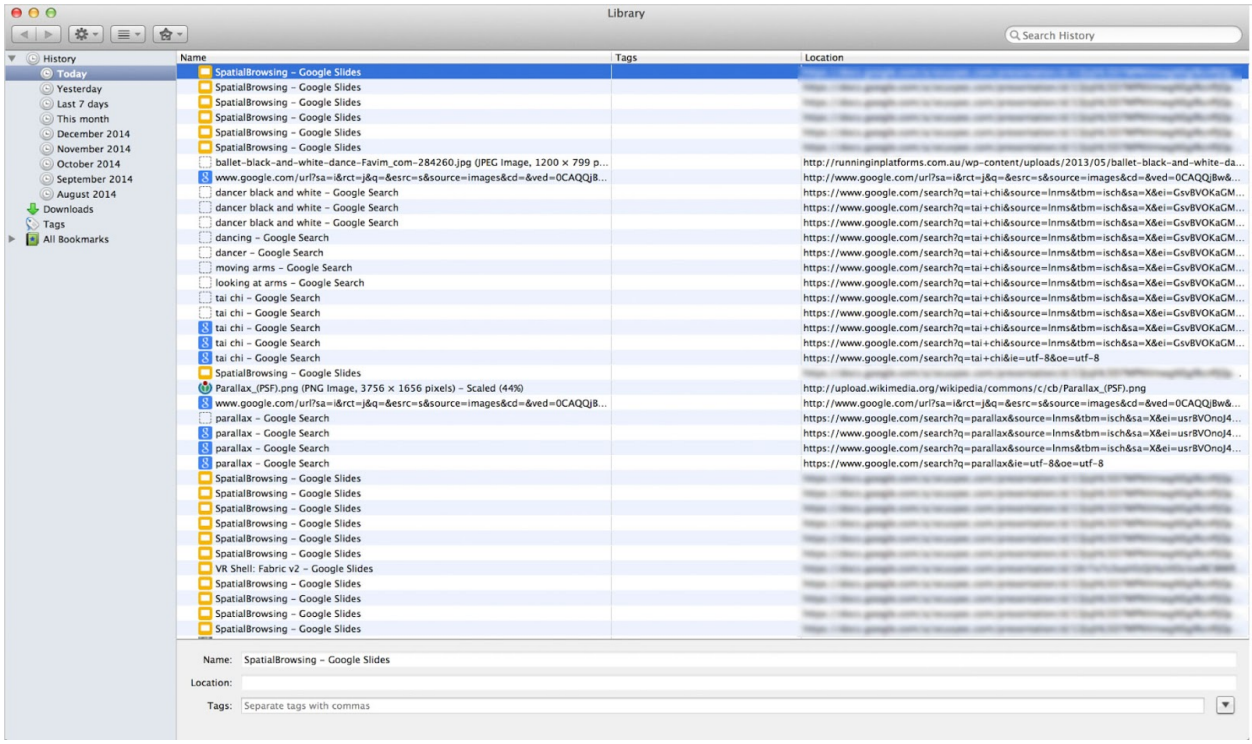
As mentioned previously, the restricted space of mobile device screens often inhibits our ability to access spatial cognition. This issue is just another example – on mobile devices, where tabs are not available, users rely heavily on the back button and new windows. This slows down their ability to navigate between various pages.



External Memory

Like many people, I leave tabs open like “don’t forget” notes on a mirror. These tabs are important – reminding me of tasks I need to do, articles I want to read, and funny videos that I will never get around to watching. Browsers often serve as a user’s active memory, and so it is very important that users be able to easily and accurately jump to any given element quickly and reliably.

Studies show that the more windows and tabs a user has open, the more important the spatial relationships become.[42] Temporal-based interactions (e.g. alt-tab) are far less helpful than spatial consistency even in today's limited digital space, and spatial consistency in the configuration of tabs encourages re-visitation – even three months after use.



My browser history for today in Firefox.

The browser has an excellent spatial system, and yet when I look at my browsing history, I see a mess of links that are all given the same illegible data. As noted earlier, thanks to the emergence of tabs, many users engage in parallel browsing – following multiple strains of thought in different windows or tabs.

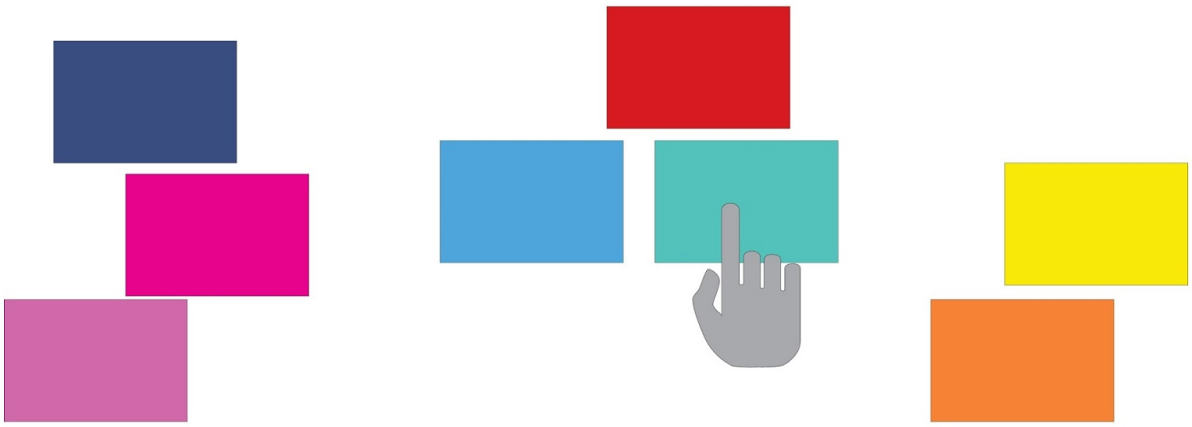
This generates a hodge podge history of activity, which is a nightmare to see in a single dimension like the one above. All the spatial data is lost along with the visual representation of the page, and all that is left is a short description and URL.

VR Space and the Browser

With AR/VR, we have the opportunity to increase and improve spatial cognition in the browser by developing a stronger spatial system and allowing for dynamic data dimensionality. With a strong sense of space, the user can quickly set up spatially optimized

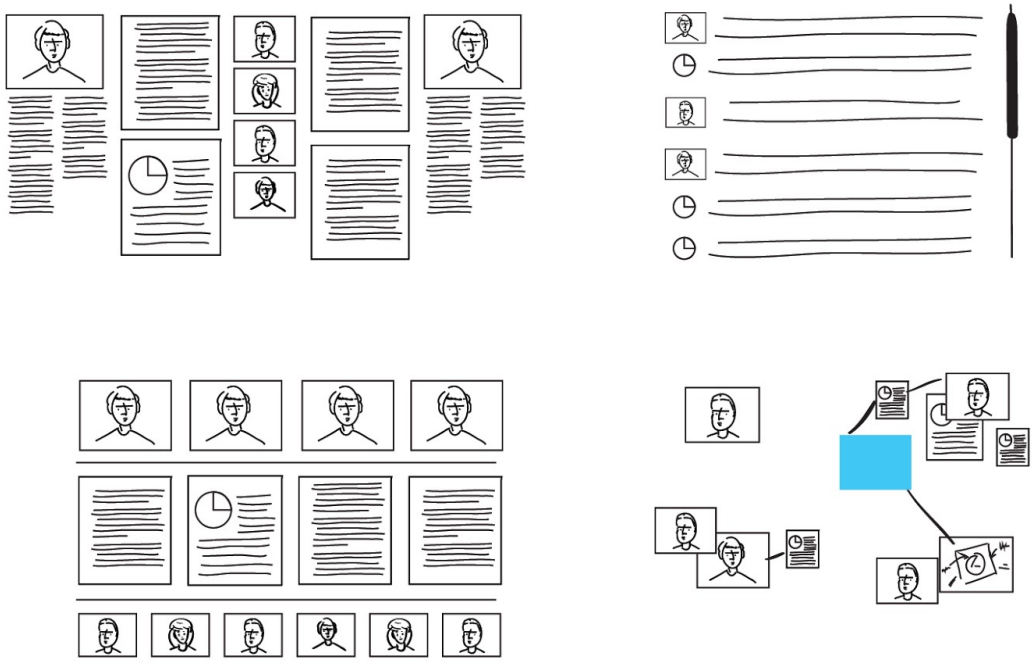
[42] Susanne Tak, Understanding and Supporting Window Switching, available from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.228.8223&rep=rep1&type=pdf.

task flows. AR in particular creates opportunities for users to map their virtual spatial systems to their real ones – opening up rapid development of spatial cognition. In both cases, however, we have a theoretically infinite canvas to spread out.



Spatial Semantics
The key to a successful spatial browser is a strong baseline grid. To lean on users' existing expectations based on over a decade of tab browsing, we can maintain the existing "launch tab to right" pattern. At the same time, allow users the full reach of their space to organize data into spatially relevant areas using simple drag-and-drop interactions over that baseline grid. Regardless of dynamic reshuffling of space, it is essential that this canvas retain the spatial location of each item until specifically altered by the user.

External Memory
With this spatial consistency, the user can maintain "memory tabs" and return to them through spatial memory. This also helps the user create muscle memory for frequent tasks and activities.



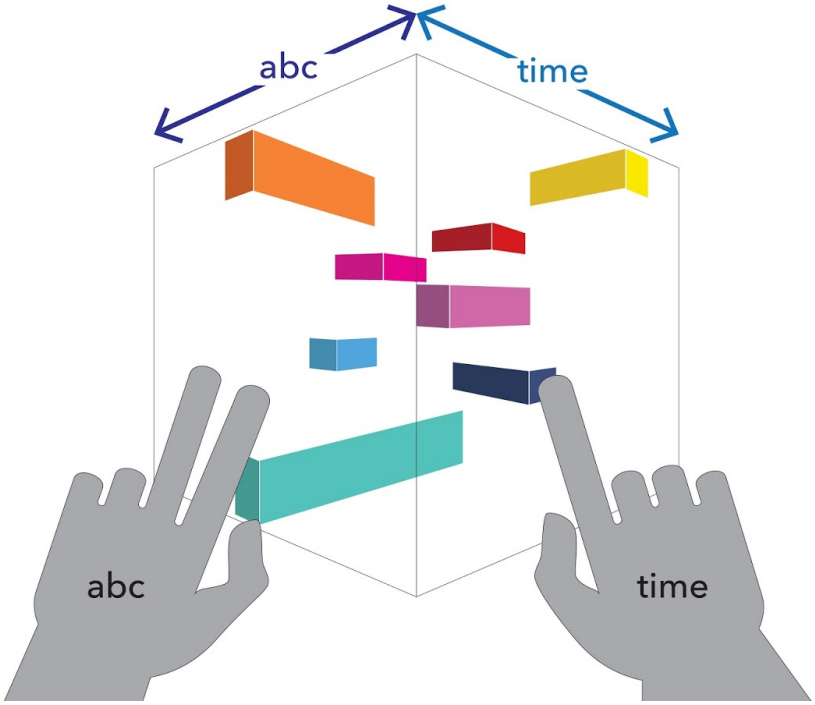
Dynamically resort objects spatially to reveal meaning. Sort methods clockwise from top left: timeline, alphabetical, type, user specified.

Dynamic Spatial Semantics

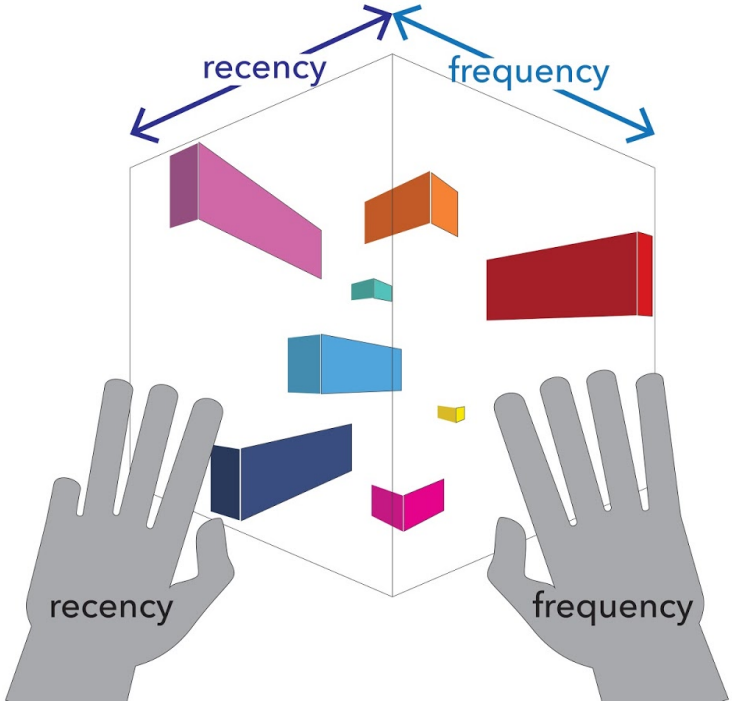
Now that the user can always return to their baseline spatial system, we can capitalize on the digital power of data by providing dynamic spatial semantics. Two projects from Microsoft, Pivot[43] and SandDance[44], demonstrate the power of dynamic movement between data visualizations to reveal patterns within data. The animated transitions between the views help users understand the context.

[43] Microsoft Research, Pivot, available from http://research.microsoft.com/en-us/downloads/dd4a479f-92d6-496f-867d-666c87fbaada/ (17 December 2010).

[44] Microsoft Research, SandDance, available from http://research.microsoft.com/en-us/projects/sanddance/ (date unknown, accessed 27 January 2015).



Dynamic Dimension: ABC x Time



Dynamic Dimension: Recency x Frequency

Dynamic Dimensionality

However, both Pivot and SandDance were developed for screens – a 2D environment. While this reaches the limit of today's shells, AR/VR offers us the opportunity to create 3D intersections of data visualizations. In other words, the intersection of two 2D data visualizations providing a 3D sense of data dimensionality. Data is given a dynamic volume as defined by the values of the intersecting graphs.

In practice, one application of this approach would be that items most related to the two visualizations become large and nearer to the user, while items that are not likely to be relevant fall away. In this way, by quickly looking at the dimensions involved, the user can instantly understand the difference between various items – just like in the real world.

Conclusion

The ability to see and understand the world in three dimensions is an extraordinarily powerful part of our everyday experience. In many ways, this has been lost with traditional digital interfaces, as UIs are inevitably shaped and limited by hardware capabilities. By unlocking the third dimension, VR/AR opens up the opportunity to combine spatial-cognitive tools and experiences with the raw power and infinite malleability of the digital medium.