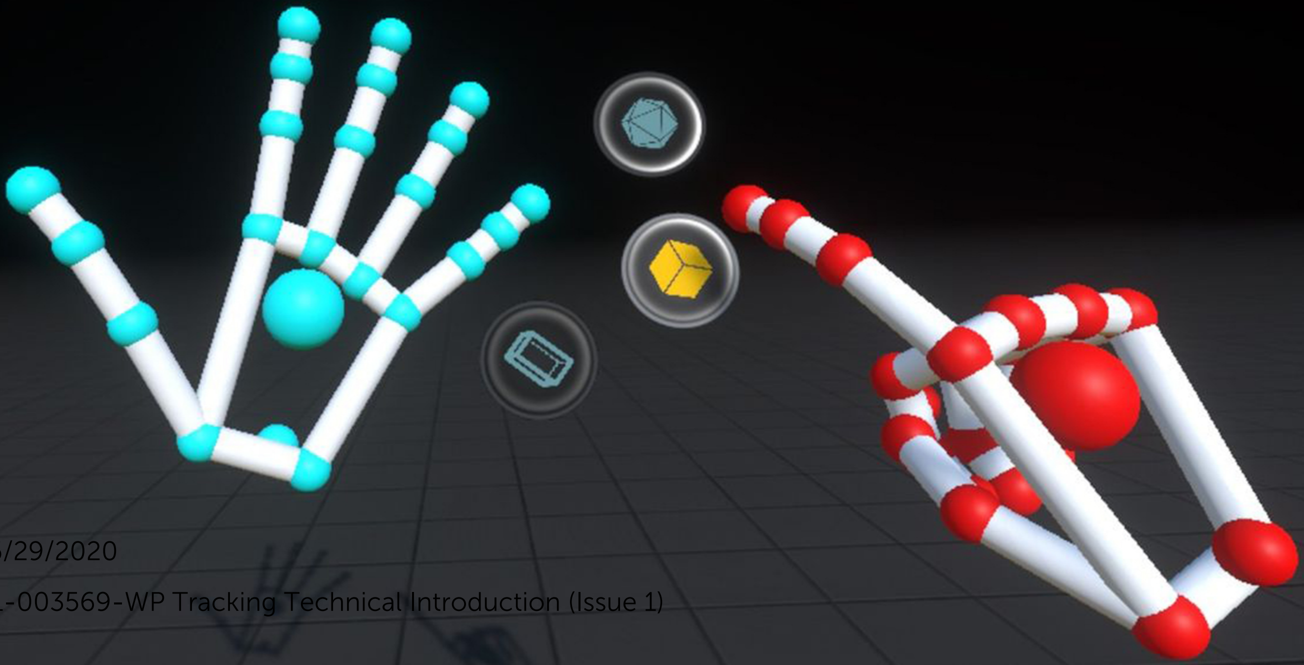
Ultraleap Hand Tracking: Technical Introduction

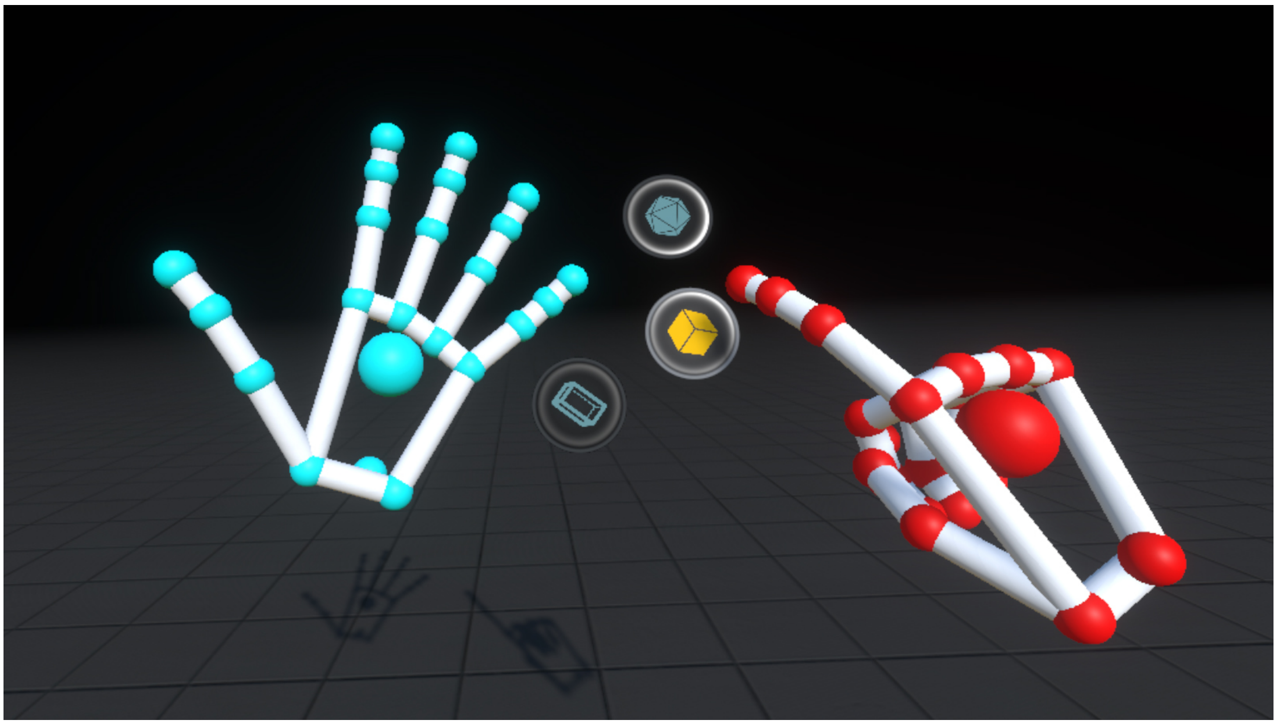
Tech Knowledgebase



06/29/2020

UL-003569-WP Tracking Technical Introduction (Issue 1)

Table of Contents



Ultraleap (formerly Leap Motion and Ultrahaptics) is the world leader in tracking and haptics technologies that make the digital world feel more human. Our hand tracking technology is designed from the ground up to provide the world's fastest and most accurate hand tracking.

The hardware itself is relatively simple – the magic is in our tracking software and Interaction Engine, which together ensure a seamless and intuitive end-user experience. This guide covers how the technology works, essential developer tools, user experience design best practices, and practical tips.

1 Technology Overview

1.1 Hardware

Our *Leap Motion Controller* features two infrared cameras and multiple LEDs. The LEDs illuminate your hands with infrared light that is invisible to the human eye, while image sensors send data back to the computer via USB to track your hands.

The light and cameras both operate within a narrow range of the near-infrared spectrum, allowing them to be robust in a range of environments and lighting conditions.

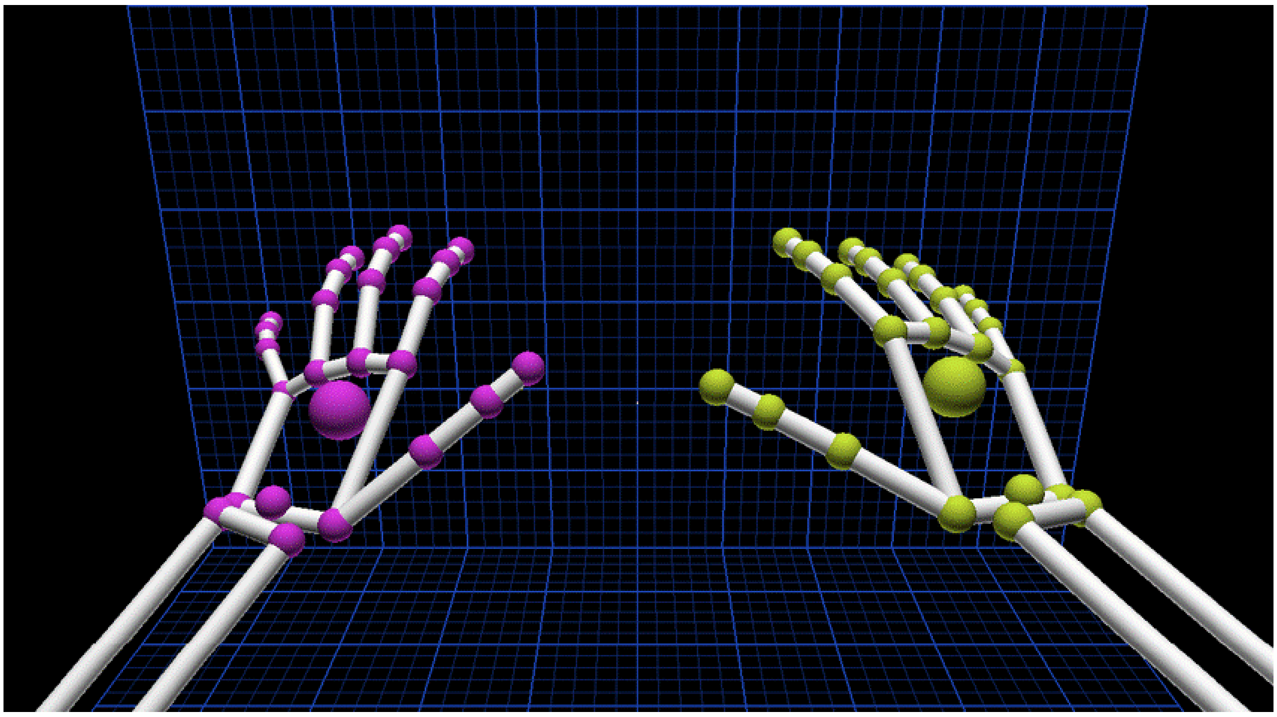
- Designed for productivity applications with Windows computers, integration into enterprise-grade hardware solutions or displays, or attachment to virtual/augmented reality headsets for AR/VR/XR prototyping, research, and development
- Range: Between 10cm (4") to 60 cm (23.5") preferred, up to 80 cm (31.5") maximum;
- Field of view: 140°(H) x 120°(V)
- Framerate: 120 fps

1.2 System Requirements

Minimum system requirements (desktop): Windows 7+ or Mac OS X 10.7 (note that OSX is no longer formally supported; older 32-bit applications are no longer supported by OSX 10.15 Catalina); AMD Phenom™ II or Intel Core™ i3/i5/i7 processor; 2GB RAM; USB 2.0 port. VR headsets may come with their own system requirements.

1.3 Tracking Software

Ultraleap's hand tracking software is designed to model the underlying structure of the hand: not just the palm or fingertips, but the joints and bones. The software can accurately guess the position of a finger or thumb, even if it's hidden from view.



Our SDK features a C-style API called LeapC, which is designed for accessing tracking data from LeapService as well as creating bindings to higher-level languages. Integrations for the Unity and Unreal engines are built on this API. An official binding for C# is available, along with legacy bindings for C++, Java, JavaScript, Python, and Objective-C.

Our tracking software has changed substantially over the past few years. Based on the nature of your deployment, you may select one of the following release generations:

- **Legacy Desktop software** (*Version 2*) may be used to build with desktop tracking on OSX or Linux. Many of the development assets associated with this software are outdated and the software is no longer supported. We strongly recommend using more recent software versions unless OSX or Linux is a hard requirement. (Note that users running Windows 10 version 1709 or above will need to apply a *manual fix* to run V2 software.)
- **Orion software** is optimized for VR but also supports desktop applications. *Version 3* preserved the legacy APIs, including JavaScript. *Version 4* is further optimized for performance on VR platforms, with all legacy APIs fully deprecated. Developers have the option to use an open source binding tool to integrate legacy-reliant applications with LeapC.

1.4 Device Orientation

The hand tracking has two different modes: desktop (facing up from a table) and head-mounted (fastened to a VR/AR headset). This mode is set at the application level by the developer. Other device orientations are not recommended. Our Unity and Unreal integrations feature example code and documentation for both desktop and head-mounted use cases.

1.5 Developer Tools

Plugins for the Unity and Unreal game engines make it easy to incorporate hand tracking into your established workflow. Both feature a powerful Interaction Engine that exists as a layer between the hand tracking data and the game engine physics. This makes it possible to grab, push, and interact with virtual objects in a way that feels natural and intuitive.

The Unity integration also features:

- Example scenes with interactive scripts
- Menu interface utilities with buttons and sliders, including menus that float next to your hand
- Easy detection of hand states like pinch and “thumbs-up” gestures
- Customizable grab and throw mechanics
- Optimized rigged hand models and auto-rigging pipeline for custom hand designs

Unity Assets and Modules: *https://developer.leapmotion.com/unity*
Unity Documentation: *https://leapmotion.github.io/UnityModules/*
Unreal Plugin & Documentation: *https://github.com/leapmotion/LeapUnreal*

2 Integration Tips & Resources

2.1 Installation Guides

Standard installation guides for the Leap Motion Controller may be found on our *developer portal*, including the standard desktop (up-facing) use case and mounting instructions for major VR headsets.

Below are some video resources for setting up and using the desktop tracking mode:

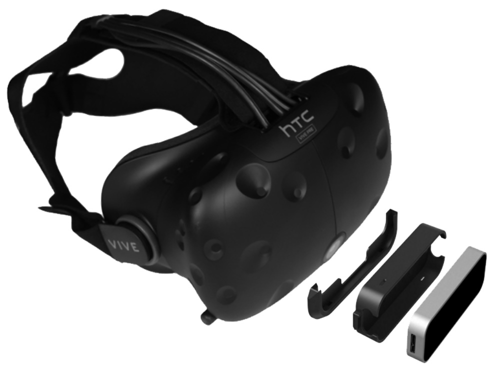
- *Taking care of your Leap Motion Controller*
- *Tips for using your Leap Motion Controller*
- *Improving tracking*
- *Setup and comfortable use*

2.2 Testing for Optimal Tracking

Available from the taskbar menu, the VR Visualizer is a powerful diagnostic tool that will familiarize you with how our hand tracking technology sees the world. You can flip between desktop and head-mounted modes, as well as see the data framerate from the device. If the data framerate is significantly below the values indicated in System Requirements, the hand tracking will appear degraded. (This happens occasionally due to USB bandwidth limitations and can usually be remedied by switching to a different USB host controller.)

2.3 Mounting and Embedding For VR/AR

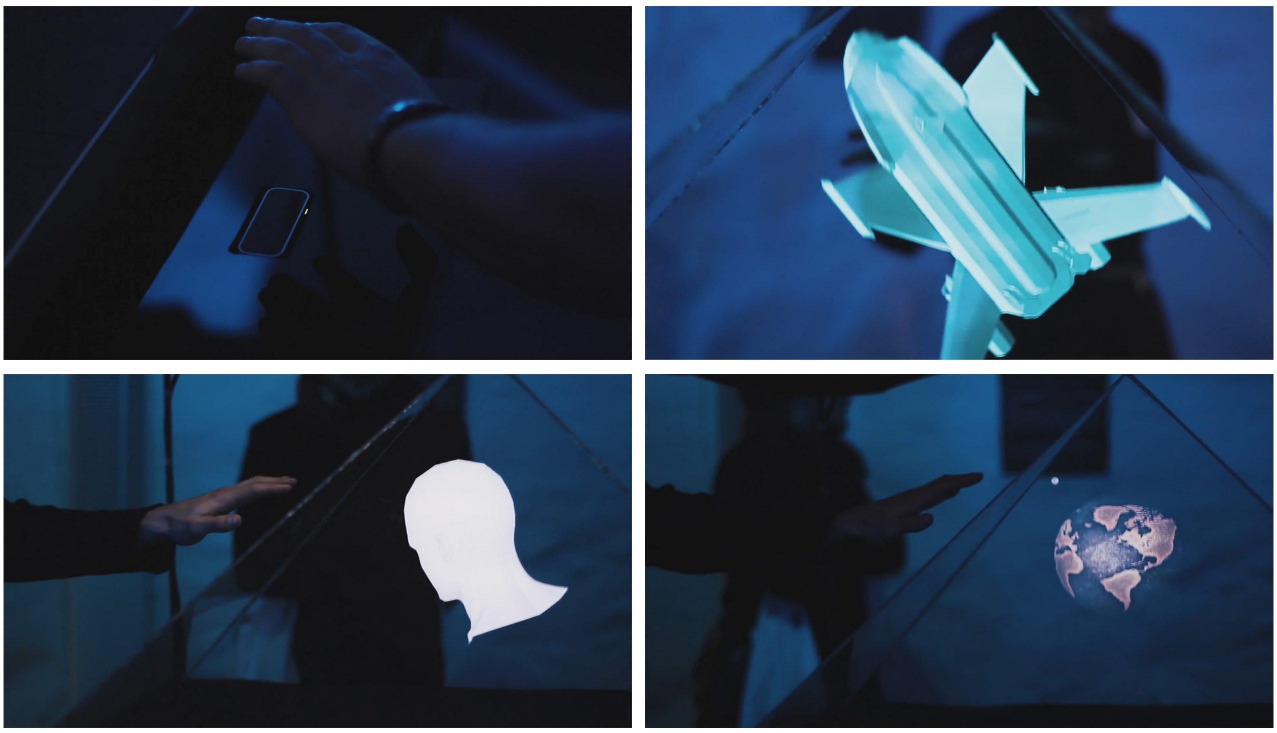
The Leap Motion Controller may be used for hand tracking in VR/AR. It can be easily mounted onto most headsets using the VR Developer Mount, which is available for purchase (as well as open sourced).



Depending on your headset and application, you may wish to adjust the virtual offset between the device and the user's eyes within the game engine. (For instance, a custom experience/headset integration in which hand alignment needs to be as closely matched as possible.) These values can be adjusted within the game engine. See our *Unity* or *Unreal* documentation for more details.

2.4 Other Embedded Solutions

Embedding the Leap Motion Controller into a larger hardware installation can create a sense of mystery and magic by hiding the underlying technology. It also shields the device from dust, moisture, and accidental damage.



If the user's hands need to be in an enclosed area, we suggest lining the inside surface of the space with a dark material that absorbs infrared light, such as duvetyne – a material used in the film industry to make dark black backdrops.

If you need to cover the device with glass or clear plastic, it should be placed directly on the top window's surface. Ensure the glass or plastic panel stays very clean (smudges will degrade tracking quality) and use the VR Visualizer utility to ensure that it is not blurring or refracting the light.

2.5 Image API

Developers can access stereo infrared images using the Image API. This may be used for augmented reality video passthrough applications. Combined with retroreflective markers and computer vision techniques, the Image API also makes it possible to track the position of physical objects as well as hands. An experimental software build, *LeapUVC*, is also available to provide fuller access to camera controls.

2.6 Multiple Devices

Experimental support for multiple interactive spaces on Windows devices is available for developers working on multiuser AR/VR, art installations, location-based entertainment, and more. It should be noted that this does not include out-of-the-box support for adjacent spaces (where a tracked hand retains the same ID when moving from one device to another) or overlapping spaces (where the same hand could be tracked from multiple angles). Learn more on the *Leap Motion blog*.

2.7 Licensing

Commercial and enterprise customers developing with our tracking technology require a license. Learn more at *https://www.ultraleap.com/licensing/*

2.8 Links

- *Documentation* (including links to current and legacy APIs)
- *Unity Assets and Modules*
- *Unity Documentation*
- *Unreal Plugin & Documentation*
- *Unreal Modules*
- *VR Design Guidelines*
- *LeapUVC*
- *Multiple Device Support*
- *Licensing*

3 Hand Tracking Design Guide

3.1 Interactive Design

Hand tracking is well-suited for a range of interactions – from controlling casual interfaces and menus, to physically interacting with virtual objects, to expressing yourself in social VR. We recommend direct physical interactions with objects and interfaces, rather than touchscreen-style swipes or grasping a virtual sword or gun.

Compelling interactions include:

- Pushing buttons or other UI elements with your fingers.
- Pinch for precise physical control (e.g. stretching or pulling a small target, such as a small object or an interactive anchor on a larger object).
- Grab to interact directly with a larger object.
- Basic hand poses such as "fingers out, palm up" to activate a menu, or "thumbs-up" to trigger scripted actions.

Be sure to mix up different types of interactions so that users can work different muscle groups.

3.2 Designing for Ultraleap Tracking

To ensure a reliable and compelling user experience, it's important to remember:

The sensor is always on. Traditional interfaces like the mouse, keyboard, and touchscreen involve physical surfaces that can push back on your fingers, making it easy for the user to know how to engage or disengage an interaction. With hand tracking, there is no physical barrier that separates interaction from non-interaction. For this reason, we recommend physical interactions such as pinch and grab, which are unambiguous and easy for the user to start and stop. Avoid abstract touchscreen-style gestures (such as swipe or airtap) which can be executed anywhere within the tracking space. Any abstract interactions should be limited in their impact and rarely a part of casual movement.

Dynamic feedback. All interactions should have a distinct initiation and completion state, reflected through dynamic feedback that responds to the user's motions. The more ambiguous the start and stop, the more likely that users will do it incorrectly. This feedback may take the form of visual, auditory, and/or haptics cues.

Designing with mid-air haptics. Haptics not only makes experiences more immersive, it also adds another dimension of dynamic feedback and context cues. With haptics, it's possible to clearly demarcate interaction from non-interaction, so that abstract gestures such as swiping may be used more effectively. Be sure to review our haptics UX guidelines to see what other capabilities haptics can bring to your experience.

Context-awareness is key. Interfaces and objects should be responsive to the user's movements and intent. This will ensure a smoother user journey and reduce false-positive interactions. For example, consider hiding smaller UI elements until the user's hand gets closer, and associate pinches with specific UI elements that may be locked along one or more axes.

Keep hands in range. If the user can't see their hand, they may not be able to use it. Encourage users to look at their hands, and design interactions that won't fail if users move their hands out of range.

Finger occlusion. Avoid interactions that depend on the position of fingers when they are out of the device's line of sight.

3.3 Interface Design

User interfaces designed for hand tracking should combine 3D effects and satisfying sound effects with dynamic visual cues. For virtual and augmented reality, menus tend to fall into two categories: fixed in 3D space, or attached to the hand.

Press
On
Press
On
Press
Off
Off
Press
Press
Off

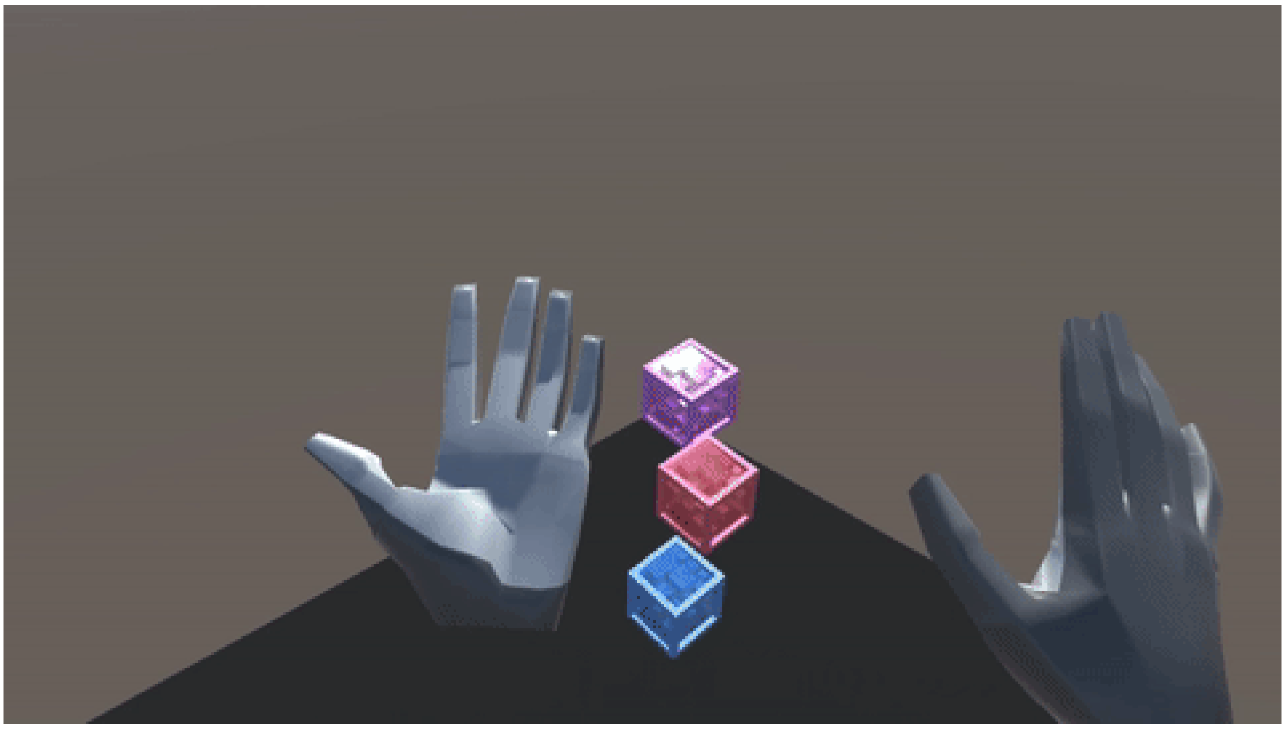
Fixed in 3D space. This a fast and easy way to create a compelling user experience. Floating buttons and sliders are stable, reliable, and easy for users to understand. These interfaces are well-suited to situations where the menu is the user's primary method of interacting with a scene.

Interfaces should be simple and easy to read, with non-obtrusive 3D effects such as shadows to give them a sense of depth and physical presence. Buttons and sliders should be responsive to the user's presence. While touchscreen button interactions are binary (contact vs. non-contact), pushing a button in 3D involves six distinct stages:

- *Approach.* Your finger is near the button, which may start to glow or otherwise reflect proximity.
- *Contact.* Your finger touches the button, which responds to the touch.
- *Depression.* Your finger starts to push the button.
- *Engagement.* The interaction is successful! The button may change its visual state and/or make a sound.
- *Ending contact.* Your finger leaves the button.
- *Recession.* Your finger moves away.

Our Unity assets provide callbacks for all six stages through the Interaction Engine, as well as UI example code that you can use in your project.

Attached to the hand. This creates a "wearable menu" that can be locked to your hand, wrist, or arm. Use our Interaction Engine's user interface toolkit to build a hand menu in just a few minutes.


A hand menu example scene from the Interaction Engine Unity module.

Hand menus are typically attached to the left hand and only appear when the palm is facing up. This allows for users to quickly and easily access the menu, without getting in the way during other interactions.

Hand menus are useful for free-flowing or casual applications where a static menu might be obtrusive. The UI elements can be buttons, such as in *Blocks*, or they can be physical objects that can be pinched and thrown into the scene to create new interfaces, such as in *Paint*.

Additional interface design tips. Text and tutorial prompts are often essential, as they:

- Clearly describe intended interactions.
- Attach prompts to the user's hands, or set them to appear near objects and interfaces contextually.
- Consider using audio and narrative cues, such as voiceover or highlighting relevant UI.

For optimal comfort, position user interfaces around the level of the user's chest, and keep broad movements of the arms to a minimum. Massive eye-level interfaces in the style of *Minority Report* will rapidly exhaust users.

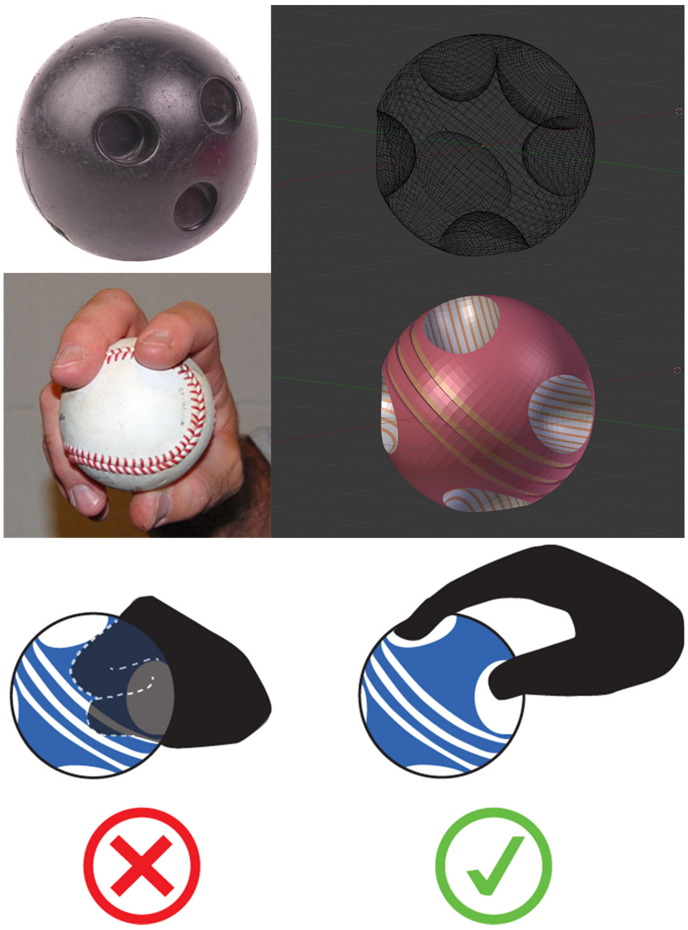
3.4 Object Interactions

The Leap Motion Interaction Engine is designed to handle low-level physics interactions in VR. With it, users can grab objects of a variety of shapes and textures, including handling multiple objects simultaneously.

When designing grabbable objects for VR, build in affordances – i.e. "physical" characteristics that guide the user in how they can handle and use that object.

- Good affordances ensure that your users understand what they can do. For example, a ball might have indents to suggest that the user should hold it a certain way, or a magical flower might have delicate petals for plucking and an unbreakable crystal stem.

- They also make it easier for you to anticipate how your demo will be used.
- The more specific the interaction, the more specific the affordance should appear. For example, grabbing is a natural interaction in most cases, but pinch-and-pull is a specific interaction that requires specific cues.
- For inspiration, look for real-world affordances that you can reflect in your own projects. The case study below includes pull-cord and turntable interactions with robust affordances inspired by their real-world counterparts.



In Weightless, users are able to grab projectiles and hurl them at targets. This is challenging from a design perspective: users have different ways of grabbing objects, and closed-fist grabs are harder to disengage. To solve this problem, projectiles in Weightless were designed with affordances suggesting how to hold the objects. Taking cues from bowling balls and baseball pitcher's grips, indentations were added to the shapes and highlighted with accent colors.

Everything should be reactive. People enjoy playing with game physics, pushing the boundaries and seeing what's possible. With that in mind, every interactive object should respond to any casual movement, especially if the user might pick it up.

4 Interactive Design Case Study: *Cat Explorer*

In the early days of smartphone UI, designers used skeuomorphic physical-world aesthetics to help users build a new intuition for digital interfaces. Today's users understand both physical and screen-based interfaces, so design can now draw inspiration from both.

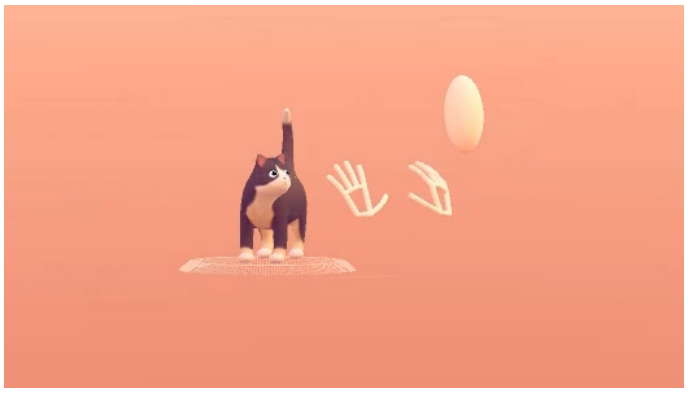
This quick case study outlines some of the design choices behind *Cat Explorer*, a VR proof of concept for intuitive interaction in training, education, visualization, and entertainment. *Cat Explorer* focuses on using hands and context-aware behaviours to explore the structural layout of a relatively complex object – in this case a loose anatomical model of a cat.

Cat Explorer features a record-player turntable and a light switch, but also a minimal graphic style more associated with digital interfaces. It also has some new interactions that are native to VR/AR. The user has never seen any of these before, but they are familiar enough to require no instruction.

Kidney
Spleen
Kidney
Lumbar Vertebrae
Duodenum
Pancreas

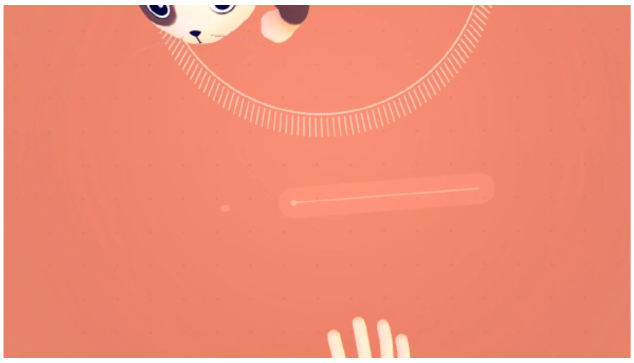
4.1 Dynamic Scene Composition

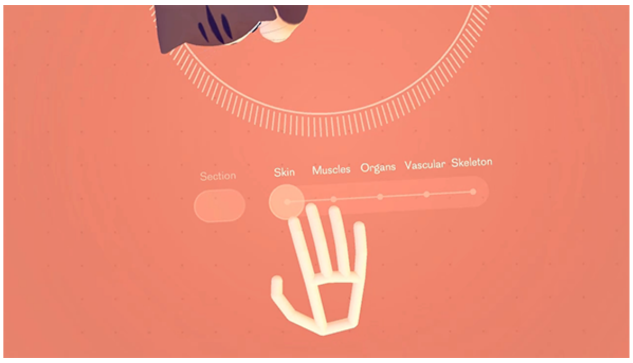
Virtual reality has unique inherent properties that need to be recognized in order for interactions to feel rich and complete. Unlike the physical world, the digital environment is dynamic. Every part of the environment can be aware of everything that's happening in all other parts, so it's possible to design interactive elements that anticipate the user's intentions, providing unambiguous and precise controls.



The scene presents the cat located at about chest height, within an ergonomic area for both seated and standing hand manipulation. Having everything located comfortably within arm's reach makes this "object on a pedestal" type of scene composition well-suited for any kind of close-range interaction.

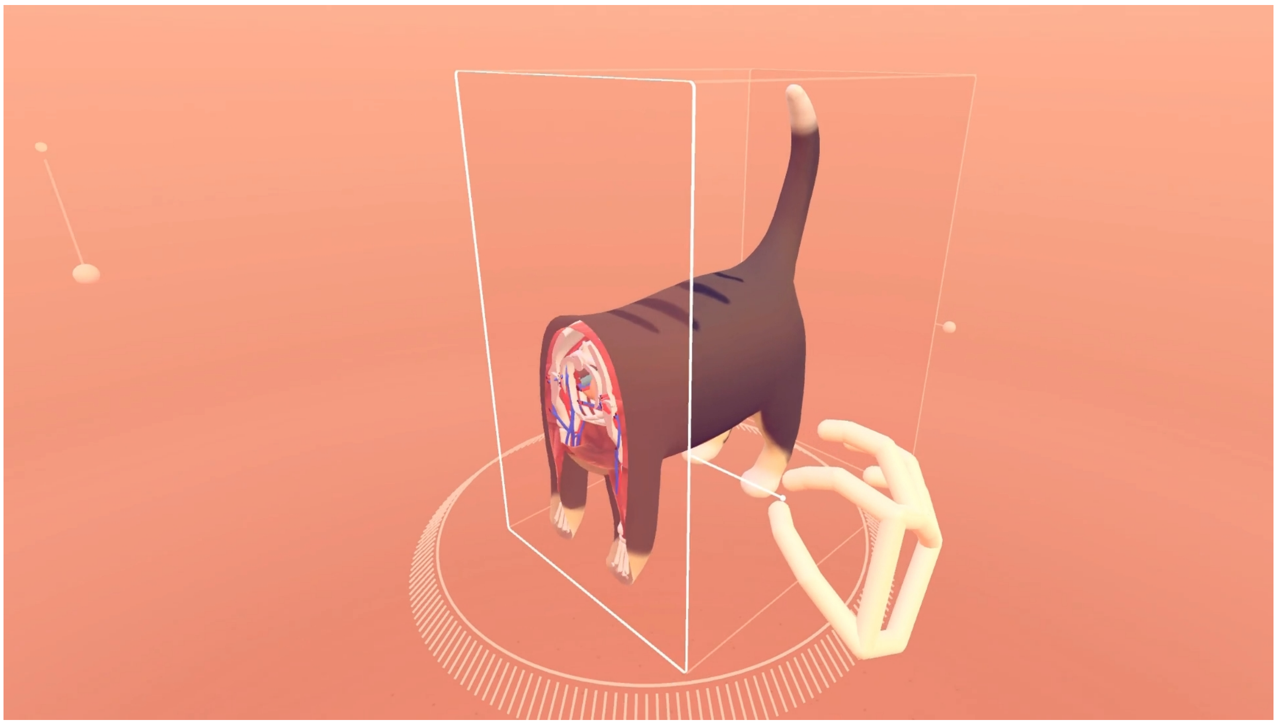
The cat and her associated interfaces dynamically follow the optimal height at all times, except when you lean in to inspect the detail. This makes the transition from seated to standing experience seamless. Environmental references are briefly faded in to make the cat's vertical movement evident, making it clear that the cat (and not the user) is the one moving through space. This ensures that the shift in height isn't disorienting or confusing. The main UI panel rotates to follow you as you move around the pedestal, only stopping once hovered over.





4.2 Fine Finger Control

Productivity interactions tend to happen at the tips of our fingers. This provides enough dexterity for small movements to be precise and expressive, while staying energy-efficient over longer periods of time. An extremely effective type of in-air finger input is the pinch, which provides haptic feedback and can be reliably repeated by most users. In *Cat Explorer*, dynamic hand-aware affordances based on the pinch gesture keep the interface elements visually subtle without losing ease of use and precision of control.



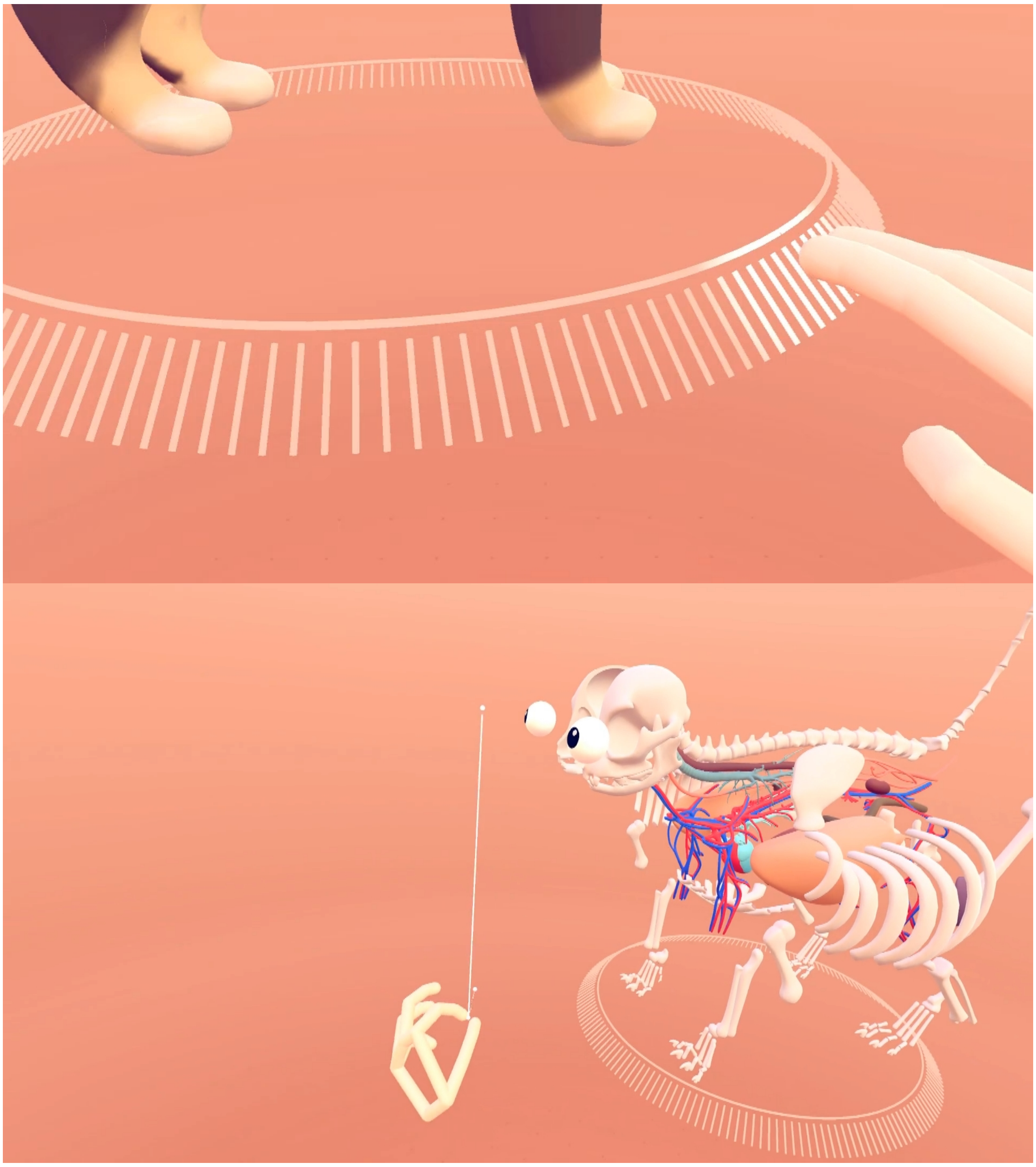
4.3 Lightweight Interfaces

One of the biggest differences between virtual reality and the physical world is the absence of haptic response. The visual aesthetic of *Cat Explorer* is designed with this in mind, featuring interactive controls that look lightweight enough that it's easier to accept the absence of haptic response:

Turntable: allows the user to rotate the cat without awkwardly grabbing it.

Slider and buttons: allow the user to switch between different anatomical states, and toggle the cross-section panels.

Pinchable UI elements: provide physical control for the cross-section panels and a "cat exploding" pull-cord.



The pinchable UI elements in *Cat Explorer* had to meet three conditions:

1. Visually suggest that it can be pinched
2. Be reliable and forgiving with regards to inaccurate hand positioning
3. Work well together with the aesthetic of the rest of the UI. The pinchable UI elements take the form of spheres and lines, combined with context-aware text prompts. Dynamic hand-aware behaviours include having the spheres extend out slightly when the user's hand approaches, inviting the user to touch them.

The pull-cord to control the cat explosion borrows from the traditional light switch design, though it can be operated as a vertical slider as well. Additionally, this handle checks the vertical speed upon release and can be thrown down for a quicker result.

Combined into a single experience, these interfaces make it possible to explore the cat from every angle and explore an annotated anatomical model. Users can phase through skin, muscles, organs, and the vascular system down to the skeleton, or explode the model to discover spatial relationships. *Cat Explorer* demonstrates how hands in VR unlock easy, friendly, and intuitive interactions.

5 More Examples

The VOID uses Ultraleap hand tracking as part of its VR arcade experiences, including Star Wars™ VR: Secrets of the Empire.

Ultraleap hand tracking is part of The Twilight Saga: Midnight Ride *attraction at one of China's premier theme parks, designed by DreamCraft Attractions and CAVU Designwerks:*

> "
>
> *Every second is valuable at theme parks that see tens of thousands of visitors daily. With over 200 players going through the system per hour, Midnight Ride needs to maintain a high throughput while also ensuring guest comfort. This made Ultraleap's hand tracking technology the natural choice. There's no need to waste valuable time strapping hand-held controllers on or teaching people how to use them.*
>
> *To meet the high-capacity demands of VR theme park experiences, DreamCraft designed a virtual reality headset with Ultraleap's embedded hand tracking technology as well as a separable head mount system. This allows the headset to stay with the ride at all times, while riders wear a magnetic mount that can be sanitized after each use.*
>
> "

Read the case study »

The Atrium
SXSW 2018 FILM FESTIVAL
WORLD PREMIERE | virtual reality
SXSW 2018 FILM FESTIVAL
OFFICIAL SELECTION | virtual reality

A groundbreaking VR installation from immersive art experience developers Meow Wolf.

First-aid skills training in VR by St. John Ambulance.

Could virtual reality retrain our brains to reverse some types of vision problems? VR medical technology firm Vivid Vision has already deployed to hundreds of eye clinics:

"

> *James Blaha created a game that forces your eyes to work together in order to win. This effectively tricks the player's brain into strengthening the weaker eye.*
>
> *Using Ultraleap's hand tracking, the game lets the player navigate various controls and play six different games. These include an asteroid shooter, a 3D variant of the classic Atari game Breakout, and targeting levels that force the user to rely on depth and colour perception. The system has a range of difficulty levels and is suitable for all ages.*
>
> *"*

Read the case study »

Designing new production lines is an enormous undertaking, with months of planning and development. With Ultraleap's technology and support, R3DT is accelerating this process with an easy-to-use tool that replaces physical prototyping with VR:

> *"*
> *During early planning, virtual prototypes make it possible for industrial engineers to avoid errors that might not be apparent in 2D. Hand tracking from Ultraleap makes interacting with 3D virtual prototypes easy and intuitive, without the need to learn buttons on handheld controllers.*

> *Prototypes can be generated at the push of a button from 3D CAD data within a few minutes. Everyone involved in the development process can quickly and easily see the latest version of the space, making critical design reviews faster and easier.*
>
> *"*

Read the case study »