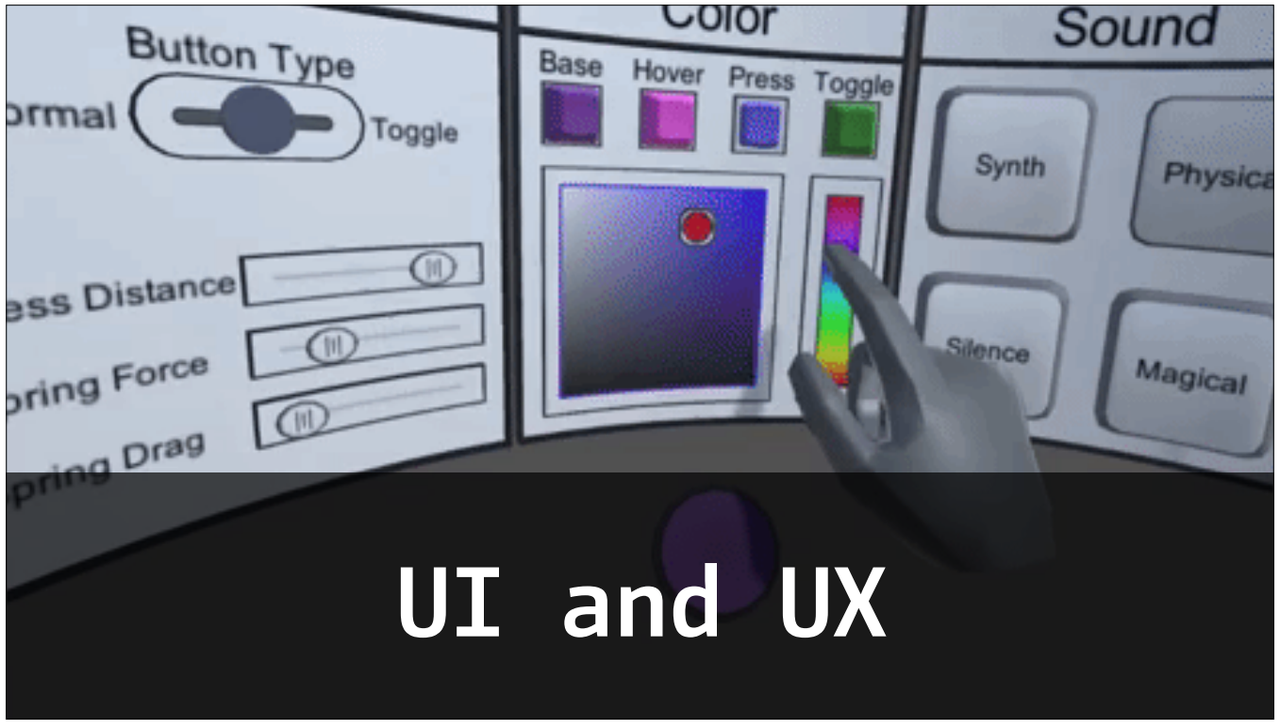

CSMA 113 - Mixed Reality Studio



UI and UX

User Interfaces and User Experience

UX is the design of *how* the user interacts with your program and UI is the *visual manifestation* of the design

Why does this matter?

People need to be able to approach something without any sort of prior knowledge and context and **still be able to figure out how to use it**.

AKA they need to be taught/shown (remember the example of the baby trying to swipe and zoom on the magazine)

Final will (partially) be judged on ability of people to use it without explanation.

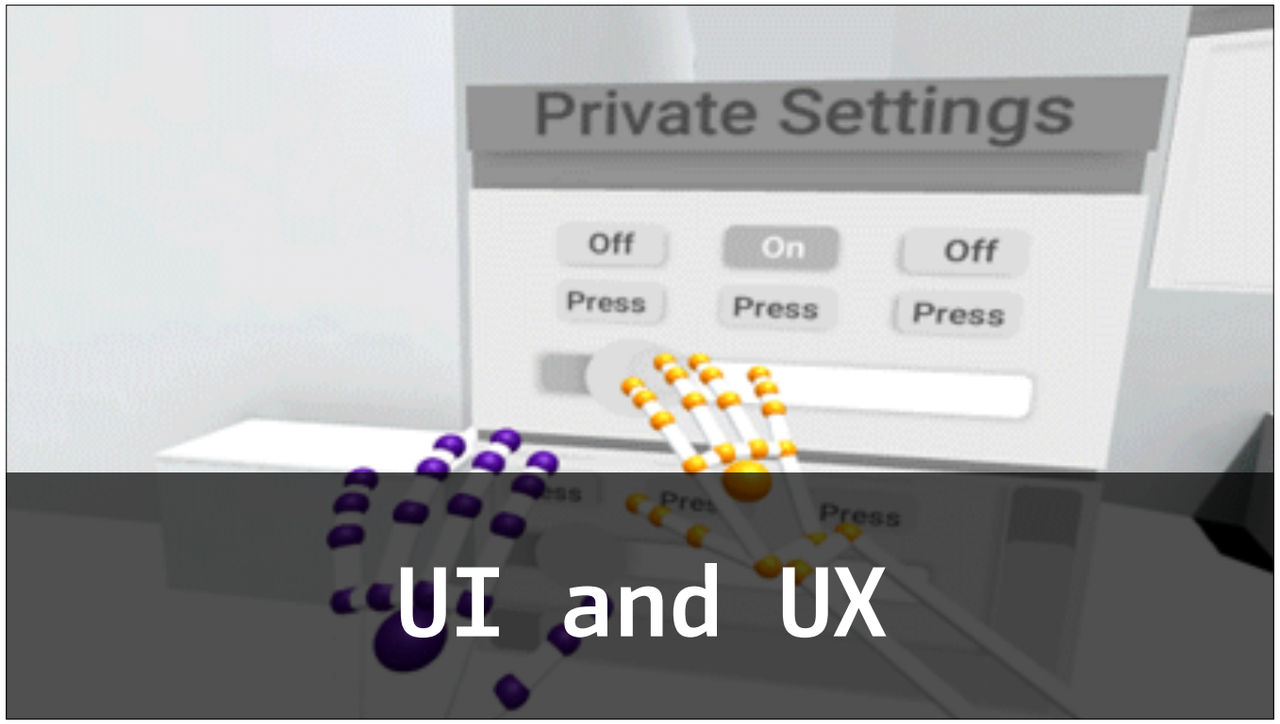
What is the interface?

Moving through an environment to discover content - no direct manipulation.

With new types of interface, we must teach users *how* to perform new kinds of interactions.

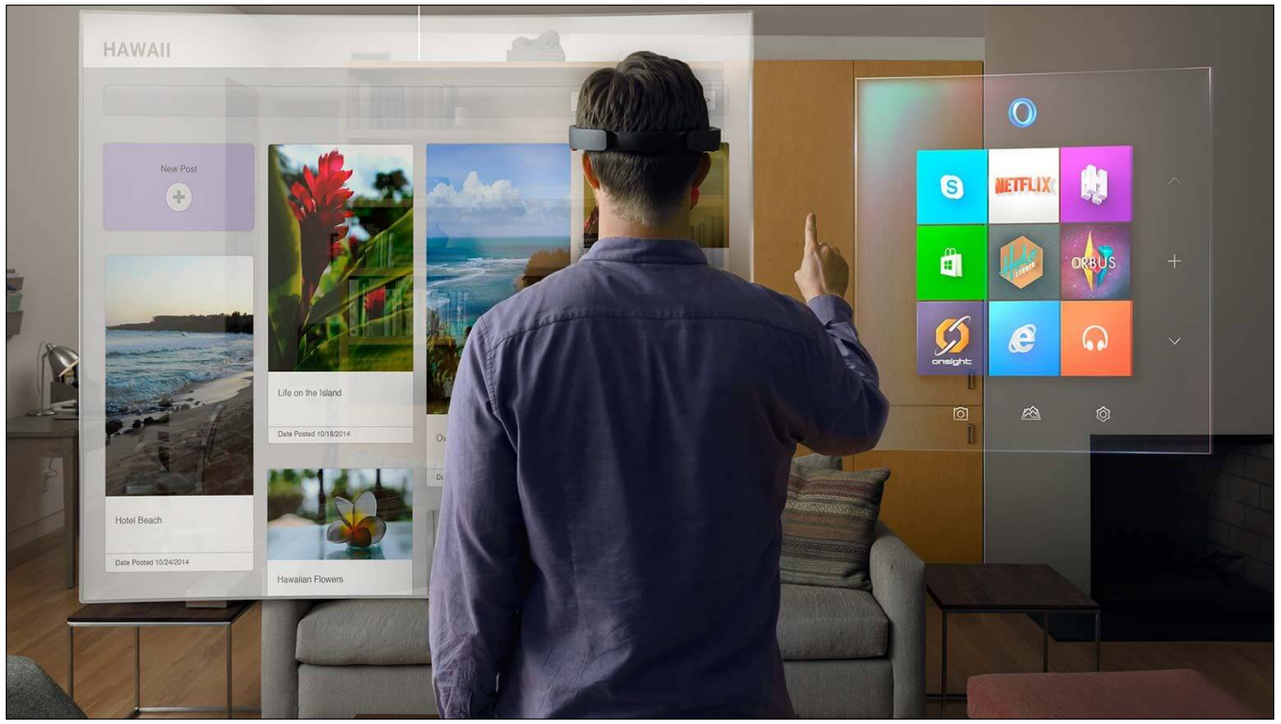
Skeuomorphism

One approach is Skeuomorphism - creating interfaces that resemble their physical counterparts.



Let’s look at “Experience” from the user’s perspective and how we can provide that…

Interface as hardware



Interface as Software/Design

Diegetic elements are part of the fictional world ("part of the story"), as opposed to non-diegetic elements which are stylistic elements of how the narrator tells the story ("part of the storytelling”).

In movies, subtitles and voiceover are non diegetic. The music coming out of John Cusack’s boom box in *Say Anything* is **diegetic**…

… but the music playing in the scenes of Tom Hardy in Dunkirk is **non-diegetic** (We would not hear it if we were in the cockpit with him)…

…and the music playing in the scenes of Tom Hardy in Mad Max is a grey area (Some music is in the scene, some is a score in the movie).

Non-Diegetic - HEADS UP DISPLAY (Terminator)

Not attached to or part of anything in the scene - serves as a “narrator” describing things in the scene.

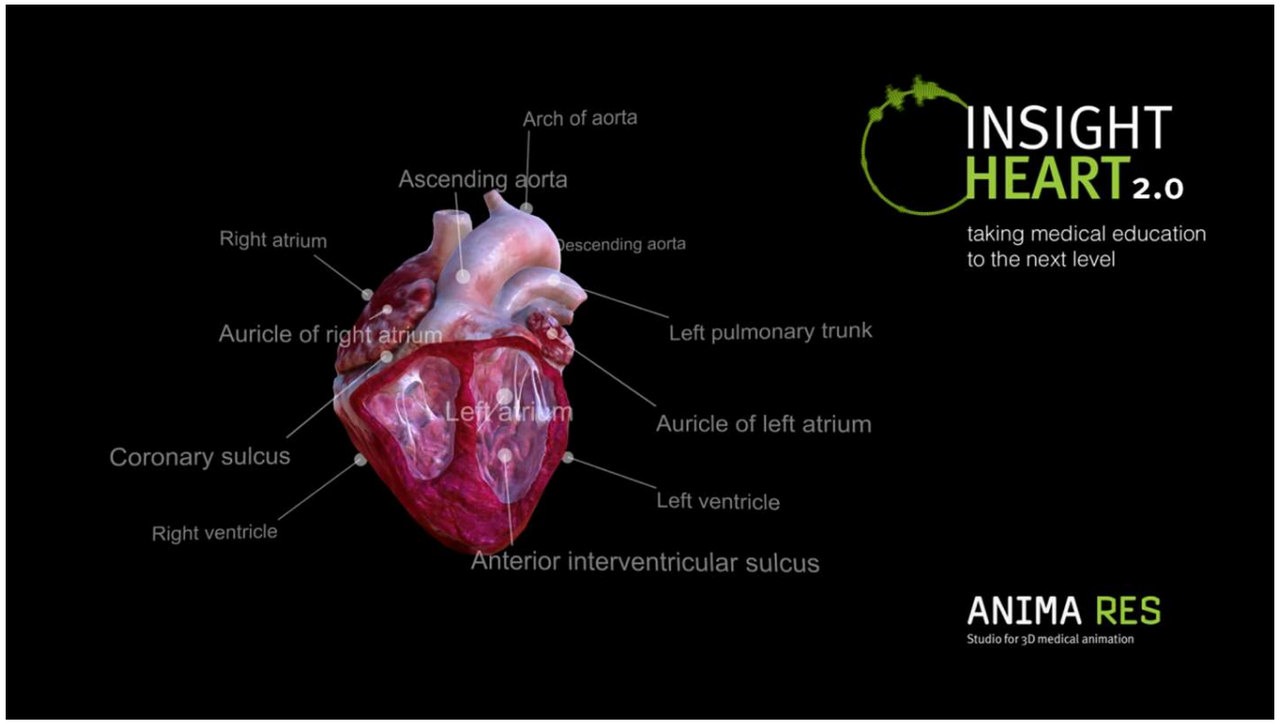
Diegetic - Menu attached to an object you can manipulate in the scene - feels like it some thing **in** the scene.

Diegetic - Same as VR menu example, but using trackable objects in the real world.

A bridge between screens/space is…

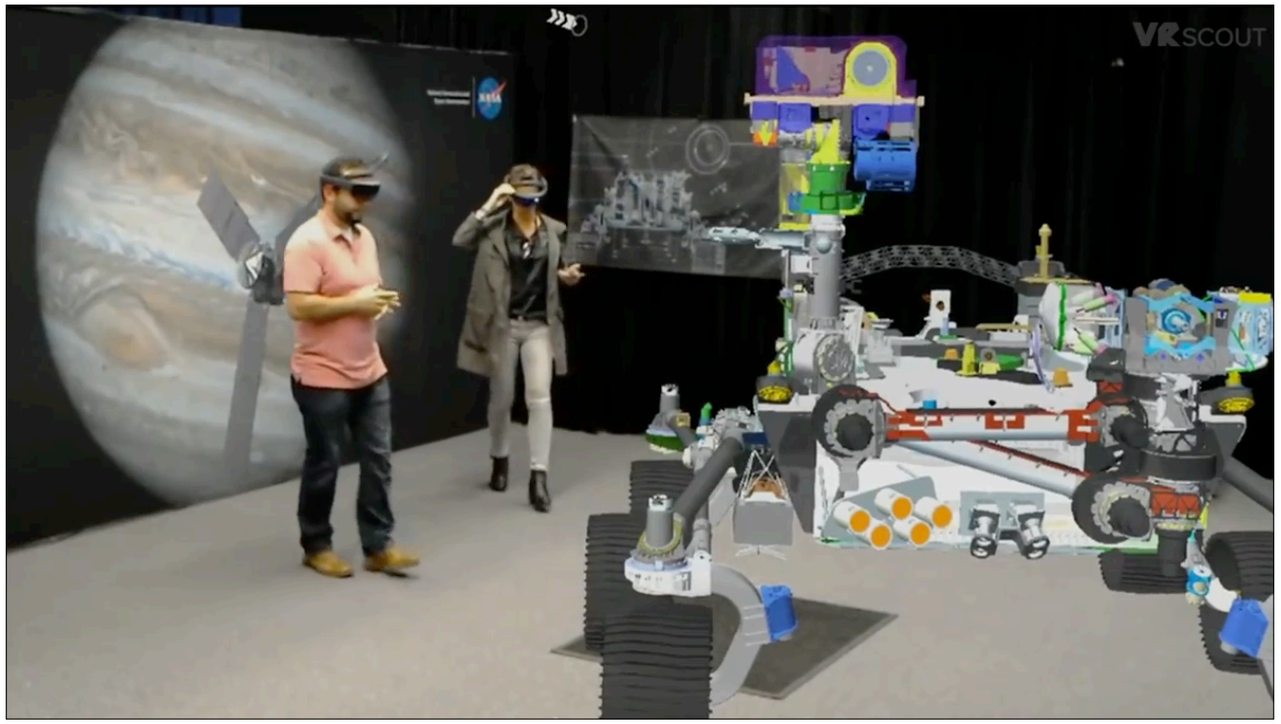
Diagetic - Panes

Screens brought into space - still can use principles of 2D (pixel-based) design where appropriate.



Diagetic - In-scene indicators (presenting information)

Argument **for** planes/screens
The less realist the shape is, the easier the user can understand it's a foreign element, and be enticed to start interacting with it.



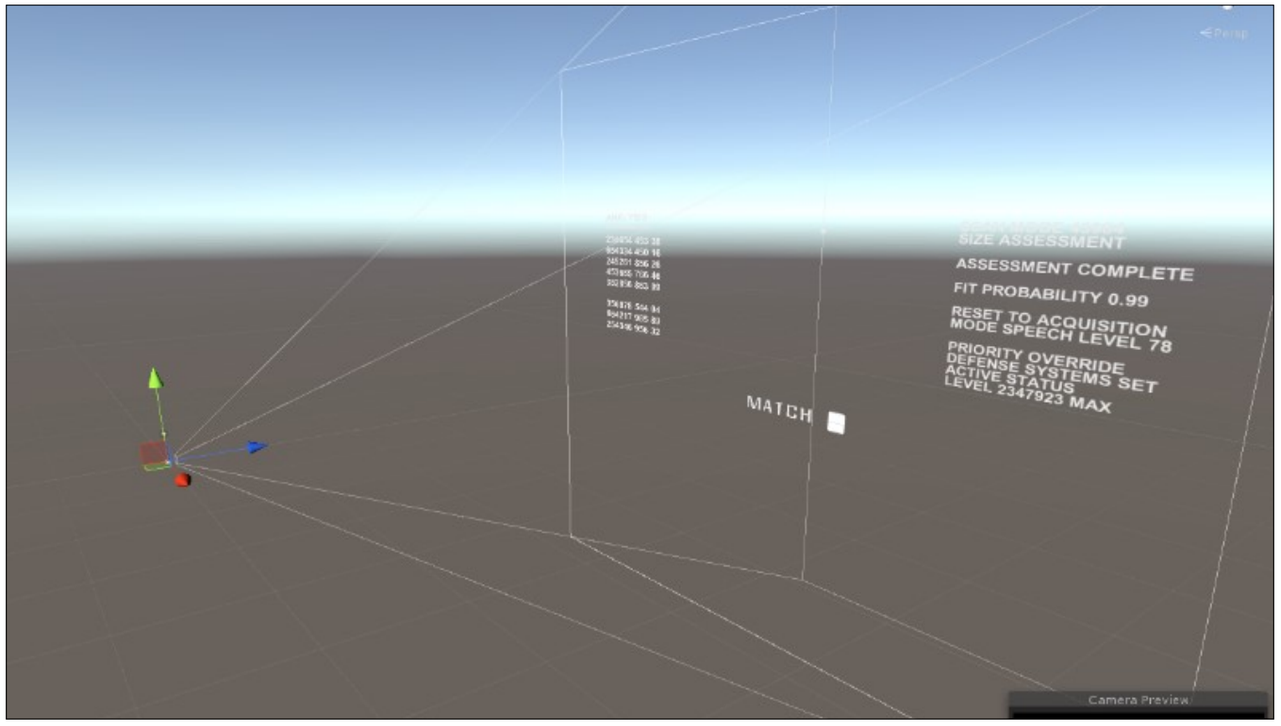
Diagetic - Moving Around

Discover new info about scene by moving around and actually looking from different points of view.

Interaction can be built around this - not just passive “looking”

(Tangent…)

Someone on the hololens team at Microsoft posted a tutorial to recreate the Terminator HUD on hololens…



(Tangent…)

Someone on the hololens team at Microsoft posted a tutorial to recreate the Terminator HUD on hololens…

…of course they tested it on their pets

IMITATING A MOVIE IS TOTALLY VALID!

This is a totally valid scope for a final project.

Designing to **prevent** certain experiences also applies.

https://en.wikipedia.org/wiki/Virtual_reality_sickness
The most common symptoms are general discomfort, headache, stomach awareness, nausea, vomiting, pallor, sweating, fatigue, drowsiness, disorientation, and ***apathy.*** *Other symptoms include postural instability and retching.[2] Virtual reality sickness is different from motion sickness in that it can be caused by the visually-induced perception of self-motion; real self-motion is not needed. It is also different from simulator sickness; non-virtual reality simulator sickness tends to be characterized by oculomotor disturbances, whereas virtual reality sickness tends to be characterized by disorientation.*

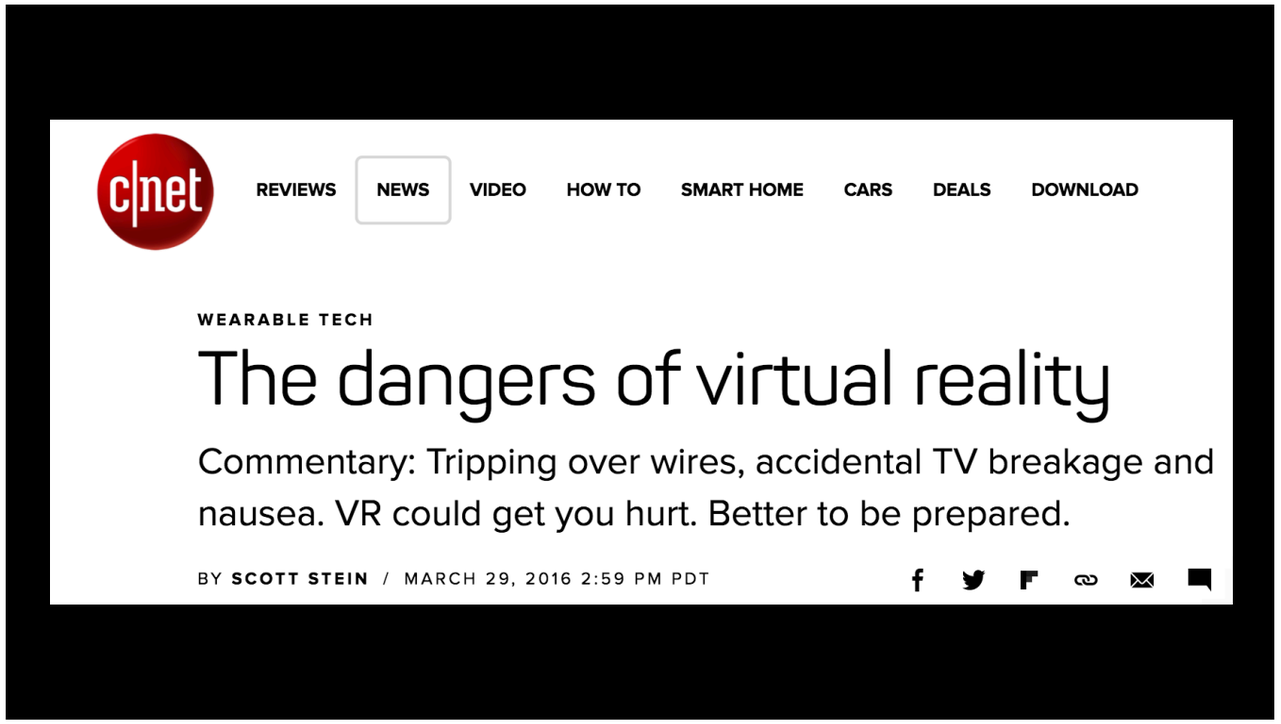
Another trigger of virtual reality sickness is when there is disparity in apparent motion between the visual and vestibular stimuli. Essentially what happens is there is a disagreement between what the stimuli from the eyes send to the brain and what the stimuli from the inner ear are sending to the brain. This is what is essentially at the heart of both simulator and motion sickness. In virtual reality, the eyes transmit that the person is running and jumping through a dimension, however, the ears transmit that no movement is occurring and that the body is sitting still. Since there is this discord between the eyes and the ears, a form of motion sickness can occur.

- individual susceptibility (Some people get it, some don’t)
- "getting your sea legs” seems to help (i.e. the more you use VR, the less you get sick)

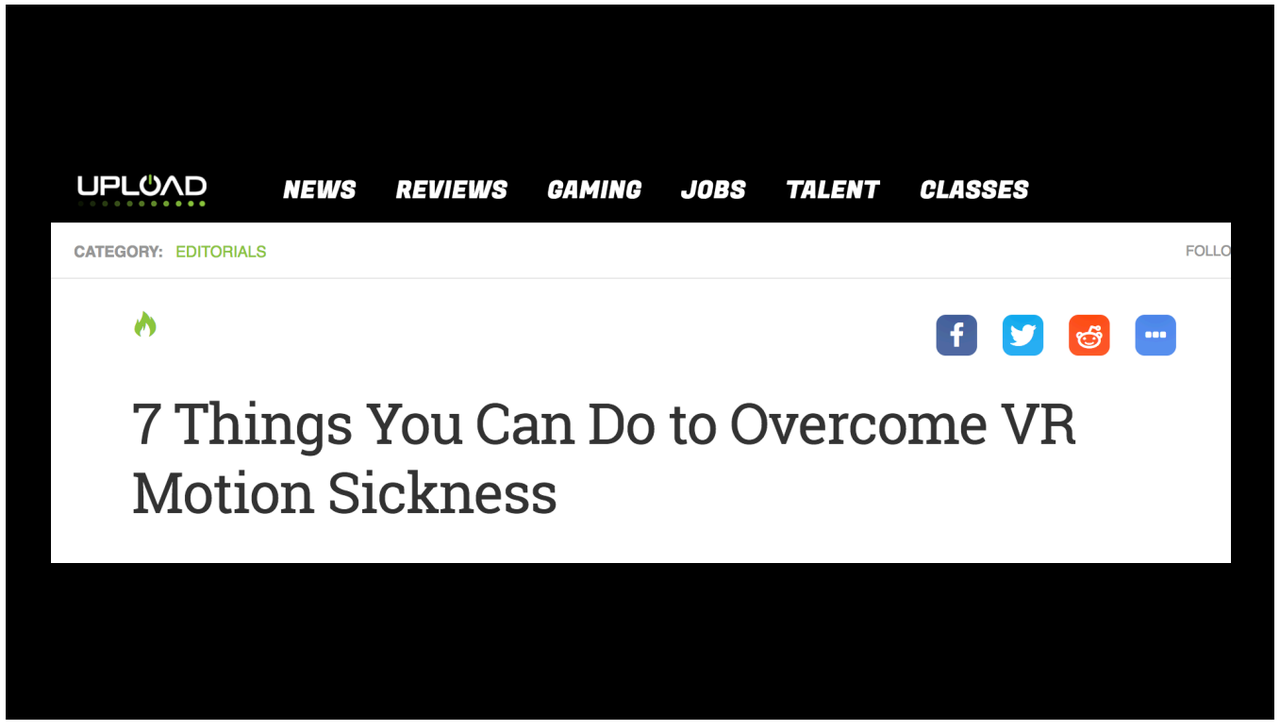
Physiological reaction - Like a poison
On its face, it makes no sense that exposure to motion should bring on disabling nausea and vomiting. But we share this seemingly odd connection between our sense of balance and the gastrointestinal tract with many nonhuman animals, including dogs, monkeys, sheep, birds and even fish. The most often cited explanation is an evolutionary theory put forward by cognitive psychologist Michel Treisman in Science in 1977. Ingesting a poison can also mess with your balance system. So the body interprets the motion reaction as a symptom of poisoning and responds as it would with poison, by vomiting to try to get rid of the harmful substance, he suggested. Although it’s just an idea and has never been tested, it has some intuitive appeal.

More in-depth

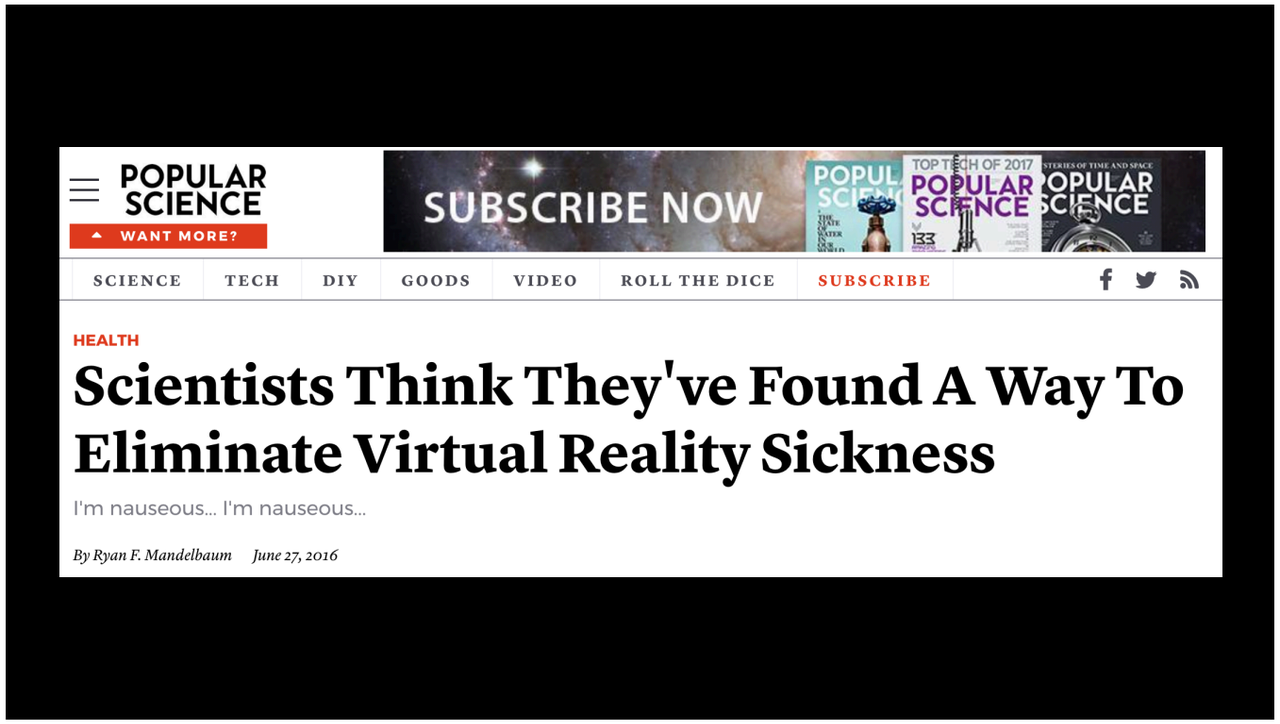
https://www.sciencenews.org/article/virtual-reality-has-motion-sickness-problem


c|net
REVIEWS
NEWS
VIDEO
HOW TO
SMART HOME
CARS
DEALS
DOWNLOAD
WEARABLE TECH
The dangers of virtual reality
Commentary: Tripping over wires, accidental TV breakage and nausea. VR could get you hurt. Better to be prepared.
BY SCOTT STEIN / MARCH 29, 2016 2:59 PM PDT

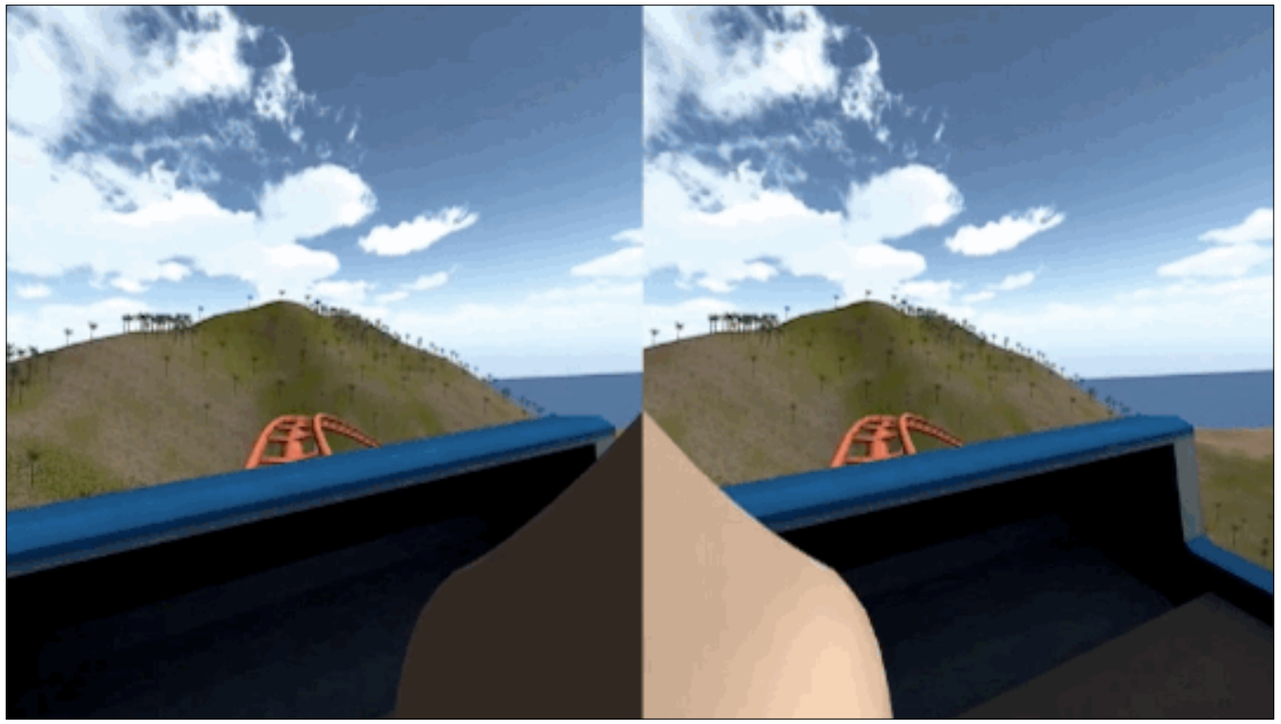

gamesradar+ THE GAMES, MOVIES AND TV YOU LOVE
News Reviews PS4 Xbox Switch Movies TV Magaz
POPULAR Vote for your GOTY! Super Mario Odyssey Guide Assassin's Creed Ori
How the battle to stop VR sickness will change game development forever
By Louise Blain October 14, 2016 News
COMMENTS


UPLOAD
NEWS
REVIEWS
GAMING
JOBS
TALENT
CLASSES
CATEGORY: EDITORIALS
FOLLO
7 Things You Can Do to Overcome VR Motion Sickness

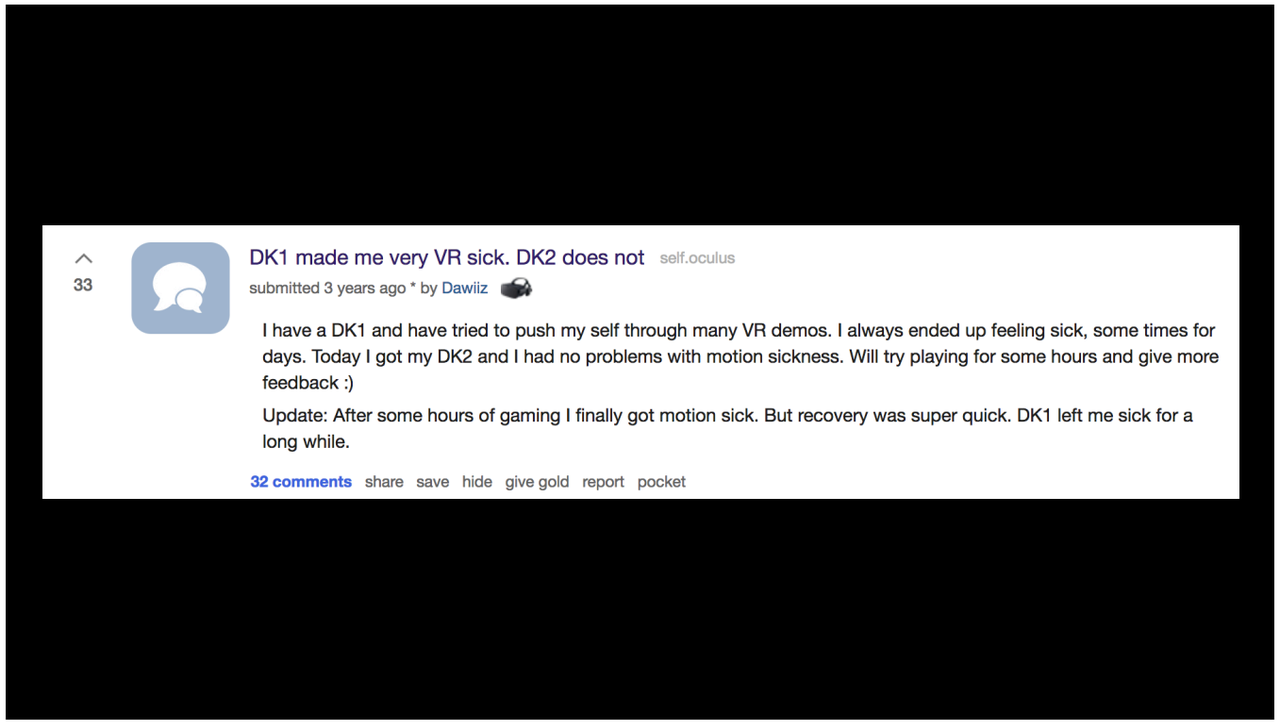

STEAM
STORE COMMUNITY IVAYLO ABOUT SUPPORT
HTC Vive
All Discussions Artwork Videos News Guides Reviews
HTC Vive > General Discussions > Topic Details
Litva Mar 16, 2016 @ 9:53am
Do you feel less motion sick in Rift then Vive ?
I`m planing to play mainly games like War Thunder or other 1st person view games, mainly Simulator games. This motion sickness they do mention more and more about feels like maby its to mutch and best you can do just play this simple games. Teleporting system in game sounds like not the best experiance but keeps you ok in VR. So how will be rift played with joystick if you get motion sick ?



Nasum Virtualis



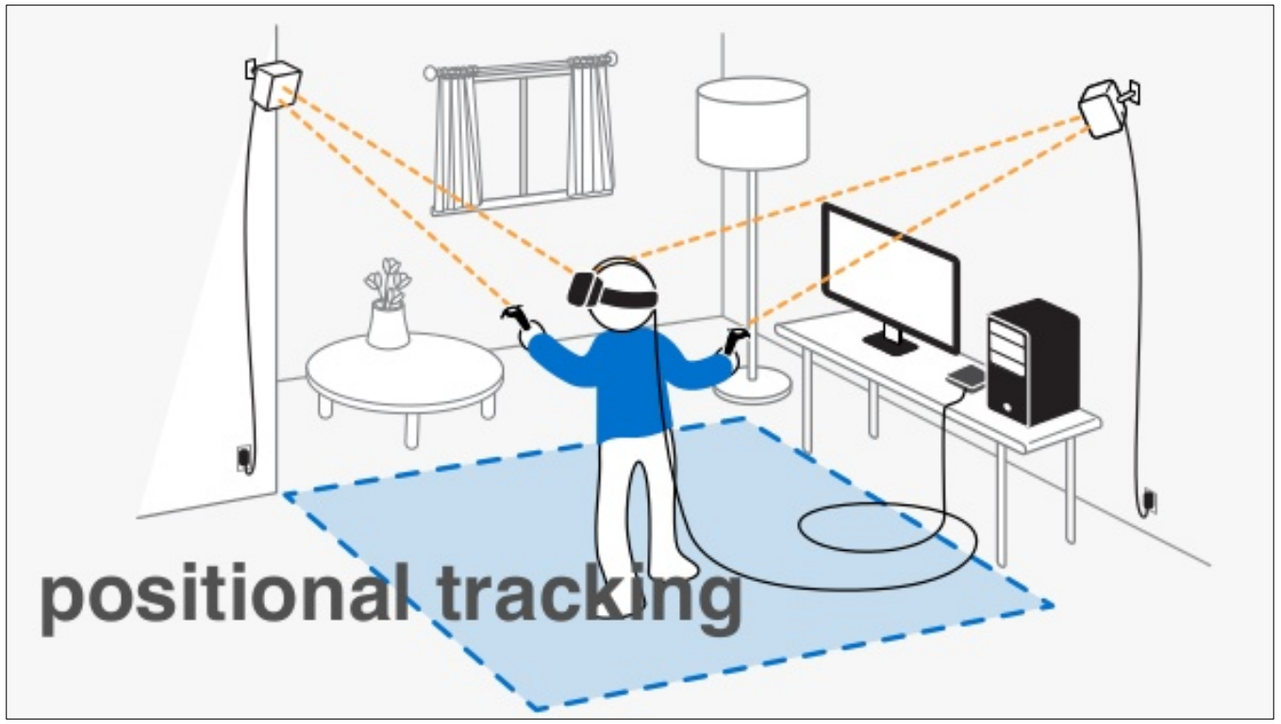
Nasum Virtualis - putting a nose in as a frame of reference.



This helped, but also the tech is just getting better too.

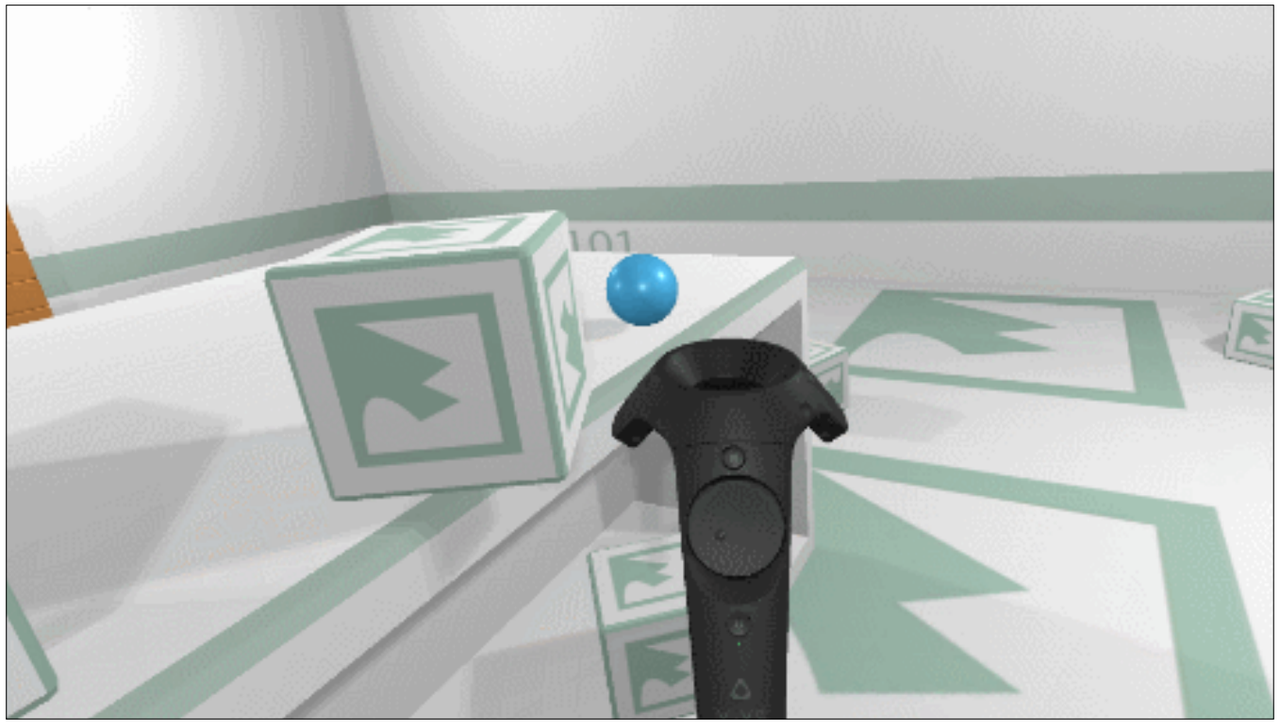
First Oculus dev kit (left) didn’t have positional tracking. DevKit2 added a IR webcam.

IR webcam was great step forward, but there were still other technical problems (for example, the refresh rate on the internal display)

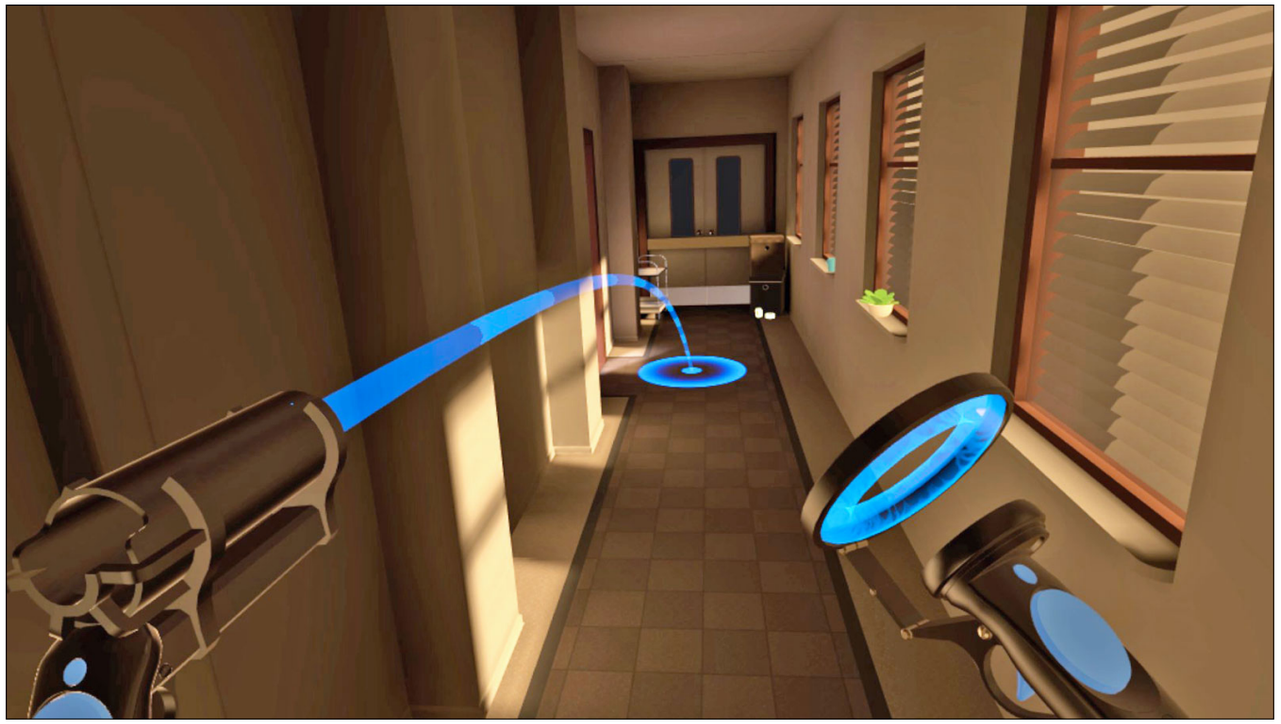


Positional tracking is HUGE step forward for avoiding sickness - avoiding disconnect between visual perception and sense of movement.

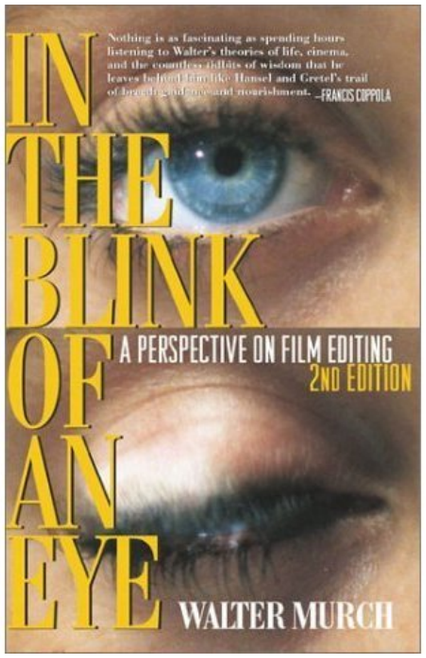
Cockpits are great ways to have frames of reference and avoid sickness…



Teleporting allows us to virtually move without *physically moving.*



Why doesn’t teleport freak us out?


Nothing is as fascinating as spending hours listening to Walter's theories of life, cinema, and the countless tidbits of wisdom that he leaves behind him like Hansel and Gretel's trail of bread crumbs and nourishment. –FRANCIS COPPOLA
IN THE BLINK OF AN EYE
A PERSPECTIVE ON FILM EDITING
2ND EDITION
WALTER MURCH

> “So it seems to me that our rate of ***blinking is somehow geared more to our emotional state and to the nature and frequency of our thoughts than to the atmospheric environment*** we happen to find ourselves in.”

The blink is either something that helps an internal separation of thought to take place, or it is an involuntary reflex accompanying the mental separation that is taking place anyway.

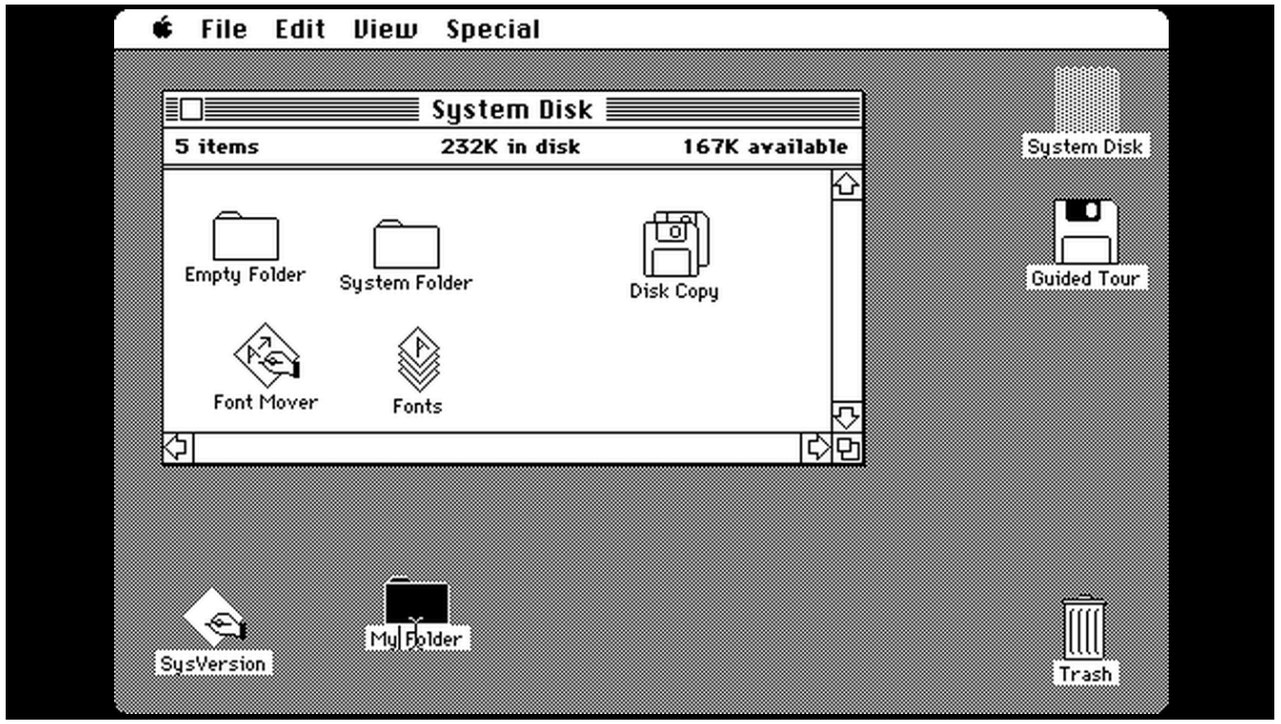
n. - A particular mode in which something exists or is experienced or expressed.

Basically, the ***way*** that you do something.

———

M EIFLER, AKA BLINKPOPSHIFT.

http://elevr.com/studio-metaphor-an-embodied-software-paradigm/



Do you keep this metaphor of files/folders?
Do we make little filing cabinets and sheets of paper.

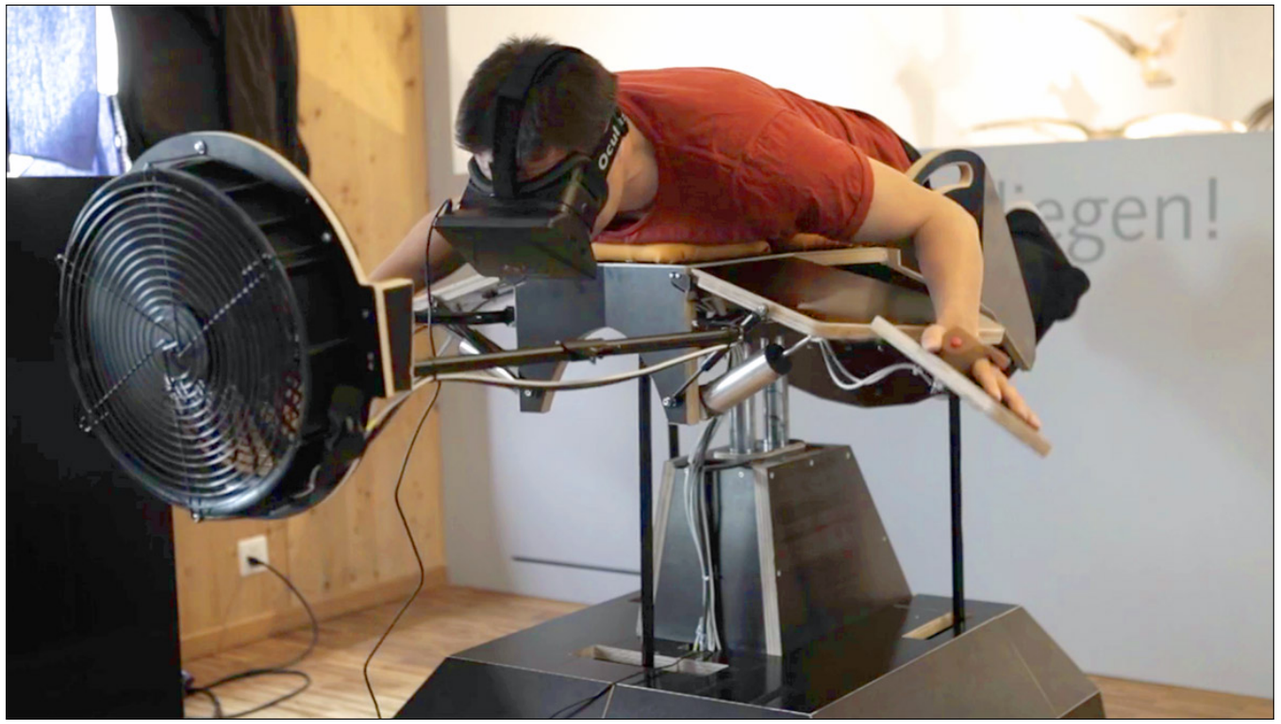
Move existing interfaces into new spaces?

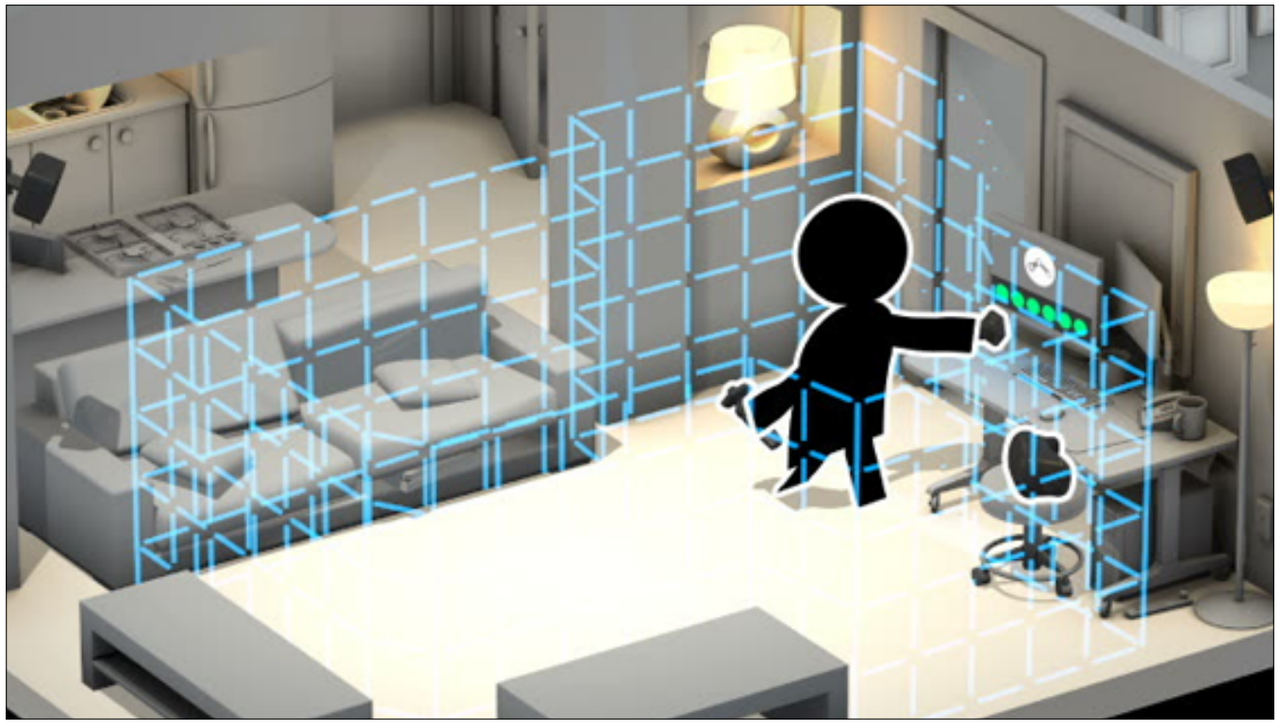
Create idealized spaces?

Create idealized spaces?

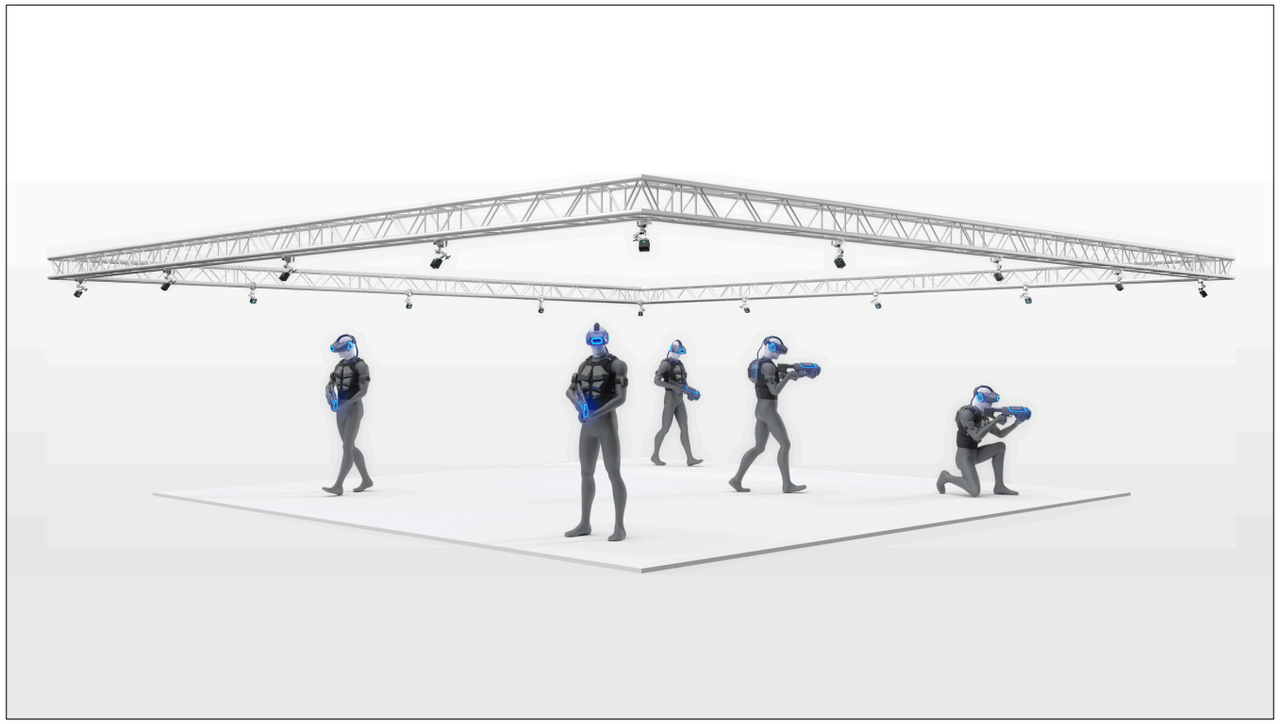
Good example: uses space and natural interaction for the tools you have available (handheld remotes)

Not just what you are doing in the experience! Think about **how** the user uses it.





Finally, think of scale!



Back into Unity

- Is everyone on Visual Studio 2017?

Vectors

In programming terms, you can think of Vectors as a way to store 2, 3, or 4 values in one easy-to-use package:

```
Vector2 someNumbers = new Vector2(1.0, 2.2);
Vector3 someOtherNumbers = new Vector3(5.3, 2.6, 12.0);
Vector4 evenMoreNumbers = new Vector4(7.4, 2.1, 12.0, 9.8);
```

Vectors

We can use vectors to:

- Store multiple numbers in one variable
- Describe the position of something in our world
    - For example: (2.1, 8.9, 7.4) represents the point in space 2.1 units along the X-axis, 8.9 units along the Y-axis, and 7.4 units along the Z-axis.

Vectors

We can use vectors to:

- Describe a direction
  - For example: (0.0, 1.0, 0.0) represents a point 1 unit directly above (along Y) the origin.
  - If we a drew an arrow from the origin to this point, it would point straight up.
  - It doesn’t matter how long the Vector is:
    - (0.0, 1.0, 0.0) and (0.0, 5.2, 0.0) are different points, but they both describe the same *direction* (straight up).

Vectors

Unity has some built-in direction shorthands:

```
Vector3 example = Vector3.up;
```

is the same as:

```
Vector3 example = new Vector3(0.0, 1.0, 0.0);
```

Vectors

Other shorthands:

- `Vector3.up` (pointing along Y-axis)
- `Vector3.forward` (pointing along Z-axis)
- `Vector3.right` (pointing along X-axis)
- `Vector3.one` (Equal to `(1.0, 1.0, 1.0)`

RayCasting

RayCasting is when we shoot an invisible line into our scene to see if we hit something in that direction.

To understand RayCasting, you must understand **Vectors**.

RayCasting

`Physics.Raycast()` is a function built in to Unity. There are many, many different forms it can take. Here is the easiest:

```
Physics.Raycast(Vector3 originOfTheRay, Vector3 directionOfTheRay);
```

All this function actually does is return `true` or `false` to answer “did this Ray hit anything?”

See the example script on the github for more details:
https://github.com/ivaylopg/Tech421Tech3706/blob/master/Session17/ScriptExamples/RayCaster.cs

RayCasting

To store information about *what* was hit, and more importantly *where* the hit is in space, we have to do two things:

1. Declare a variable of the type `RaycastHit` to store the information about the hit point.
2. Use a slightly different version of `Physics.Raycast()` to pass the hit info out of it:

```
RaycastHit hitInfoVariable
Physics.Raycast(Vector3 originOfTheRay, Vector3 directionOfTheRay, out hitInfoVariable)
```

See the example script on the github for more details:
https://github.com/ivaylopg/Tech421Tech3706/blob/master/Session17/ScriptExamples/RayCaster.cs

RayCasting

So if wanted to Raycast from a GameObject (for example a Vive tracker or the user's headset POV):

We want to shoot a ray from:
gameObject.transform.position

in the direction of:
gameObject.transform.forward

(**gameObject.transform.forward** is the **local Z-axis** of the *object*, which may be different from the *world* Z-axis, which is Vector3.forward)

See the example script on the github for more details:
https://github.com/ivaylopg/Tech421Tech3706/blob/master/Session17/ScriptExamples/RayCaster.cs

RayCasting

```
void Update() {
    RaycastHit hit;
    if ( Physics.Raycast(gameObject.transform.position, gameObject.transform.forward, out hit) ) {

        Debug.DrawLine(gameObject.transform.position, hit.point, Color.red);
        Debug.DrawRay(hit.point, hit.normal, Color.green);

    }
}
```

RayCasting

the **hit** variable that stores information about the result of the Raycast has a few useful properties:

hit.point (The coordinates of the collision as a Vector3)

hit.normal (A Vector3 direction that describes the direction coming *straight out* of the face of the hit object)

See the example script on the github for more details:
https://github.com/ivaylopg/Tech421Tech3706/blob/master/Session17/ScriptExamples/RayCaster.cs

RayCasting

These visual Debug functions help you see what's going on. They will draw lines in your *Editor,* but never in the actual *Game* view:

```
Debug.DrawLine(Vector3 lineStartCoordinate, Vector3 lineEndCoordinate, Color color);

Debug.DrawRay(Vector3 lineStartCoordinate, Vector3 lineDirection, Color color);
```

See the example script on the github for more details:
https://github.com/ivaylopg/Tech421Tech3706/blob/master/Session17/ScriptExamples/RayCaster.cs

Design For Humanity - Parts 4, 5, 6, 7

http://bit.ly/1TOgJ6E

Thank you!