VIRTUAL REALITY: THEORY AND PRACTICE

LESSON 2: VR SYSTEM DESIGN AND STEREO VISUALIZATION

GIUSEPPE TURINI - THU 8 FEB 2024 - UNIVERSITY OF PISA

TABLE OF CONTENTS

PART 1: VR SYSTEM DESIGN

This section briefly explores the essential elements of designing a VR system:

- **Hardware Components:** Visualization interfaces (headsets, CAVEs, etc.), VR controllers (gamepads, wands, data gloves, stylus, etc.), tracking systems (inside-out, outside-in, etc.), and audio devices (headphones, spatial audio, etc.).
- **Software Components:** VR engine (graphics, physics, etc.), UI (menus, diegetic interfaces, interactive elements, etc.), content creation tools (3D modeling, texturing, animation, etc.), interaction design (user interactions, navigation, etc.), sound design.
- **User Experience (UX):** Comfort, navigation, feedback, accessibility, testing, evaluation.
- **Content Creation:** Storytelling, narrative design, world building, asset management.

Some guidelines to design an immersive comfortable VR system will also be presented.

HARDWARE: VR VISUALIZATION INTERFACES

A VR system always requires **a stereoscopic system (a stereo visualization interface)**, that is: an optical system capable of generating stereo pairs of a target object, so that the viewer can experience stereopsis (achieve depth-perception).

Traditional stereoscopic systems employ an apparatus that keeps the left and right eye images directed solely at the appropriate eye. For example:

- Head-mounted display (HMD).
- CAVE (CAVE Automatic Virtual Environment).
- Active shutter glasses, and anaglyph glasses.

The *"holy grail"* of stereographics are autostereoscopic systems that do not require special glasses or lighting conditions. For example:

- Parallax barriers, lenticular lenses, or holographies.

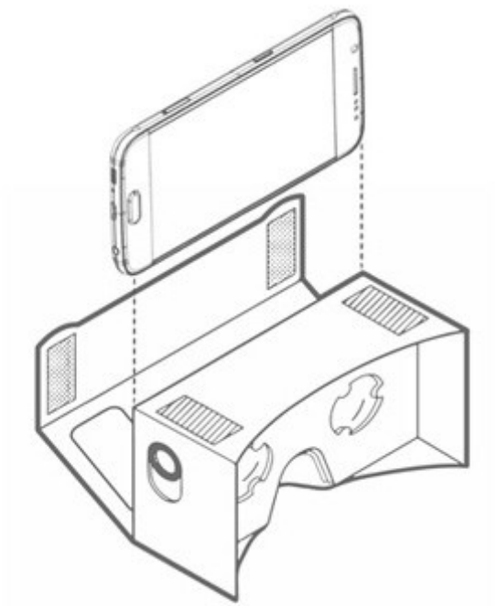
HARDWARE: HEAD-MOUNTED DISPLAY

Exploded view of a modern head-mounted display (HMD), illustrating its main components.

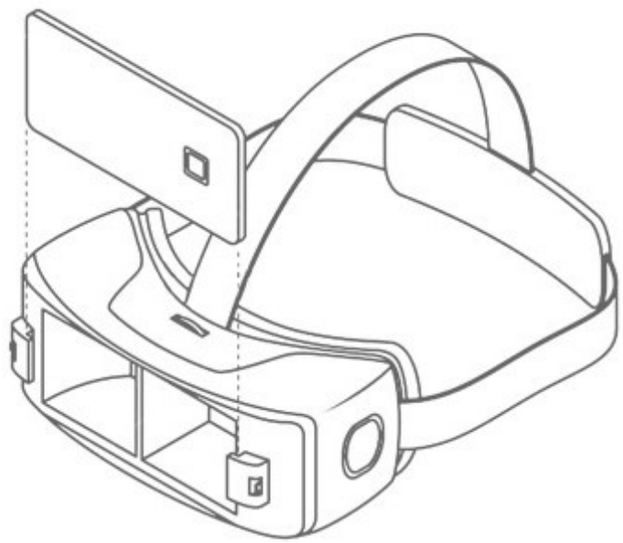
- **Internal display**, usually configured in split-screen modality (half display for each eye).
- **Tracking system** (onboard inside-out, or extenrnal outside-in) for the HMD pose tracking.
- **Internal lenses** (1 Fresnel lens for each eye to allow focus at short distance).
- **Internal processor** and battery (or as an alternative, tethered to a computer).

HARDWARE: HEAD-MOUNTED DISPLAY (2)

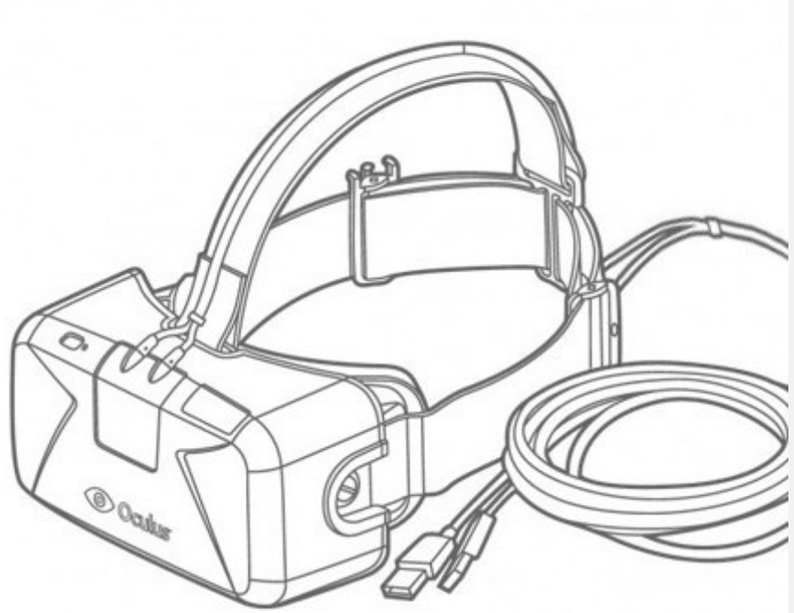
These are just examples of different design for HMD: with/without tracking system, with/without internal display, with/without cabling to an external computer.



Google Cardboard



Samsung Gear VR



Oculus Rift

HARDWARE: HEAD-MOUNTED DISPLAY (3)

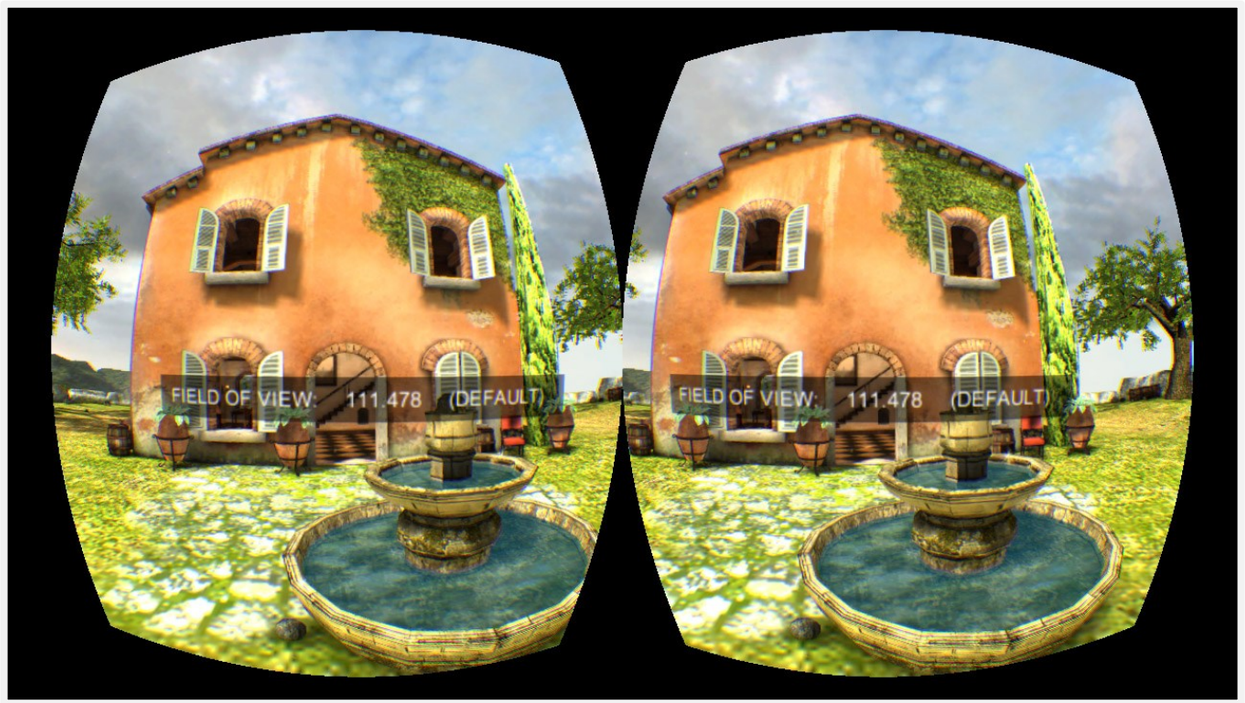
Stereo pair visualized on the internal display of a modern VR HMD in split-screen modality.

HARDWARE: HEAD-MOUNTED DISPLAY (4)

The stereo rendering for a modern VR HMD must consider the split-screen modality used byt the internal physical display.

The virtual camera configuration must match the setup of the physical display and optics of the VR HMD.

The VR software has to process/render the 3D content twice: once for each image of the stereo pair to be generated.



HARDWARE: HEAD-MOUNTED DISPLAY (5)

The horizontal aperture (FOV) of the virtual stereo camera (camera FOV, o CFOV) should match the visible area of the physical display (display FOV, o DFOV).

Display FOV (o DFOV): The area of the physical field of view of the user covered by the virtual 3D content. It is a specification of the VR HMD.

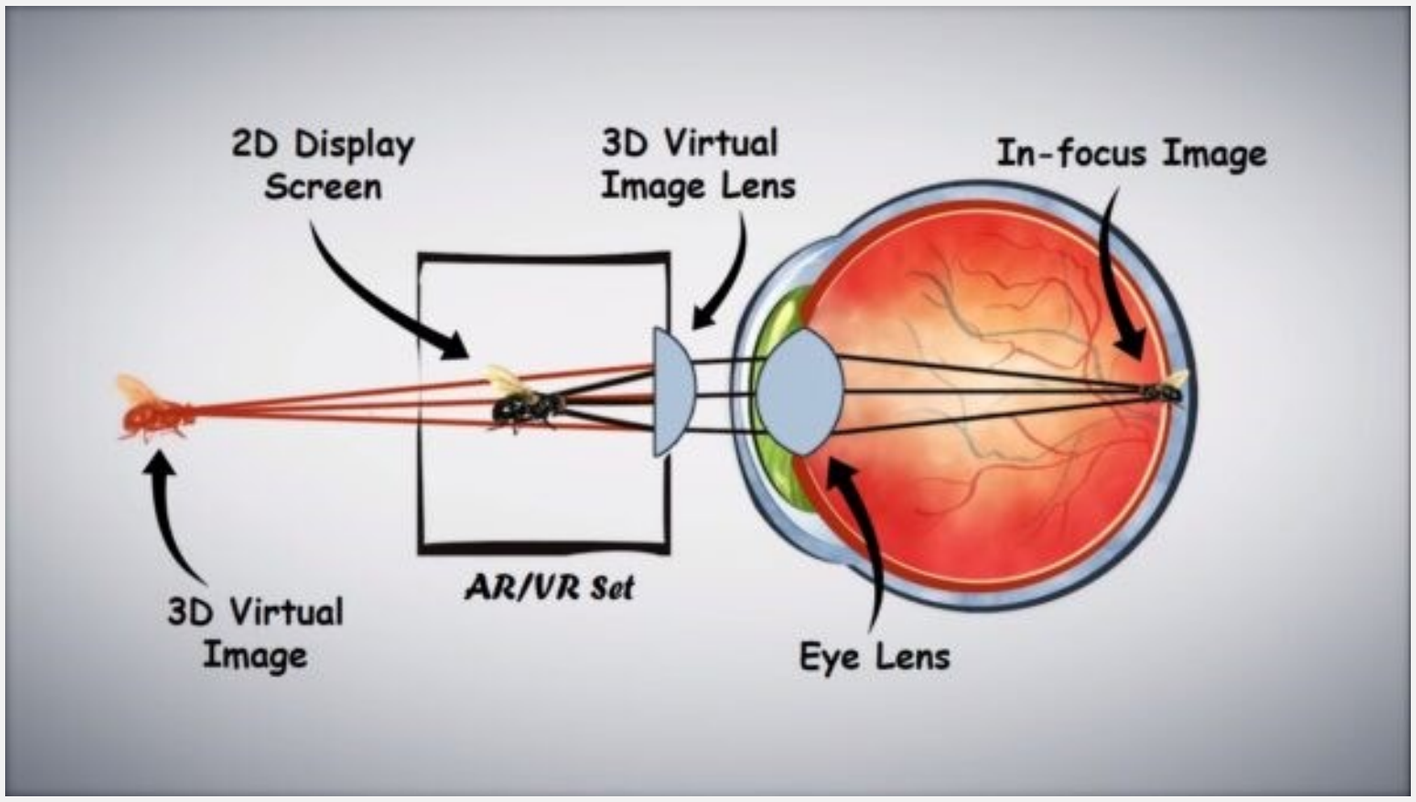
Camera FOV (o CFOV): The horizontal aperture (FOV) of the virtual stereo camera.

In VR, the external environment is not in view, and the virtual environment occupy most of the user peripheral vision. So, it is critical that the CFOV perfectly matches the DFOV.

Scale: The *"scale"* is ratio between CFOV and DFOV, and in VR it should always equal 1.0.

HARDWARE: HEAD-MOUNTED DISPLAY (6)

Diagram illustrating the role of the internal Fresnel lenses in a modern VR HMD.

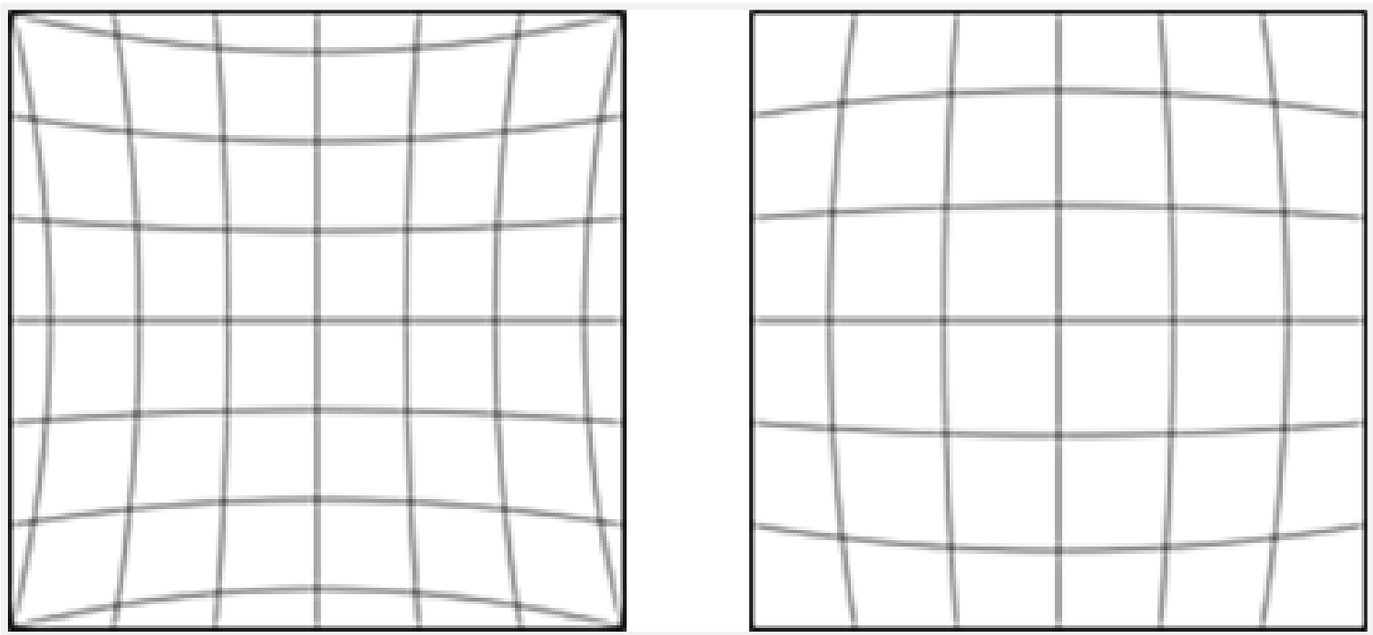


See: *"How Lenses for Virtual Reality Headsets Work"* on YouTube.

HARDWARE: HEAD-MOUNTED DISPLAY (7)

The internal lenses in a modern VR HMD require a correction of the lens distortion.

The internal lenses cause a ***"pincussion distorsion"*** (left). This lens distortion is neutralized rendering the 3D content on the split-screen display applying the opposite distortion: a properly configured ***"barrel distorsion"*** (right).



HARDWARE: HEAD-MOUNTED DISPLAY (8)

The internal lenses in a modern VR HMD cause another issue: a *"chromatic aberration"*.

Chromatic Aberration: Displacement of the colors of an image, caused by lenses.

This effect is due to the refractive index of the lenses, that is variable for each wavelength of the light.

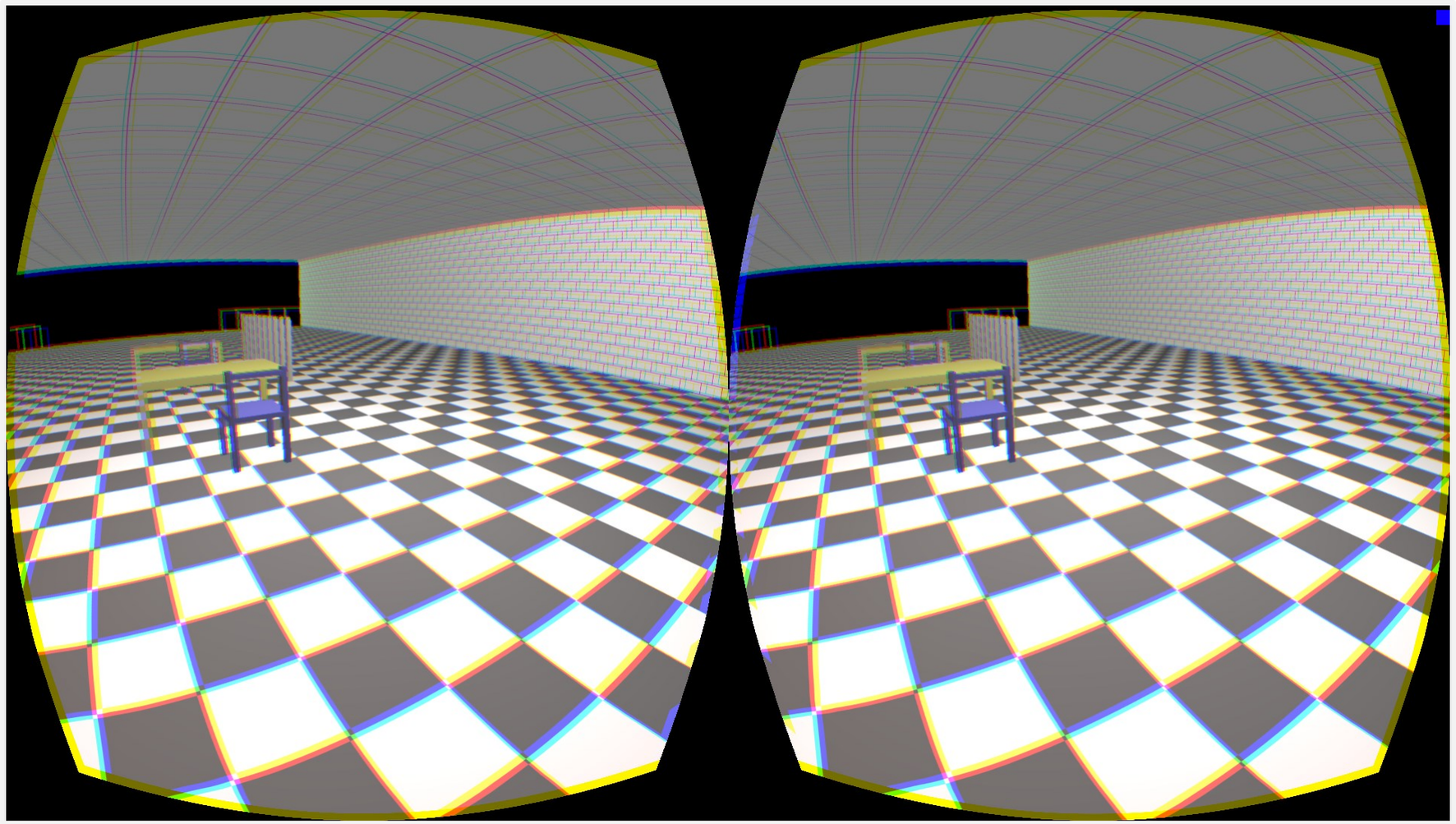
Images visualized on a display are composed by pixels with individual colors (RGB) that are affected by different refractions.

This effect is more significant near the image borders than the image center, and different colors are affected in different ways (the blue is affected less that red, etc.).

Note: Usually, *"chromatic aberration"* is corrected at low level (device driver, or shader).

HARDWARE: HEAD-MOUNTED DISPLAY (9)

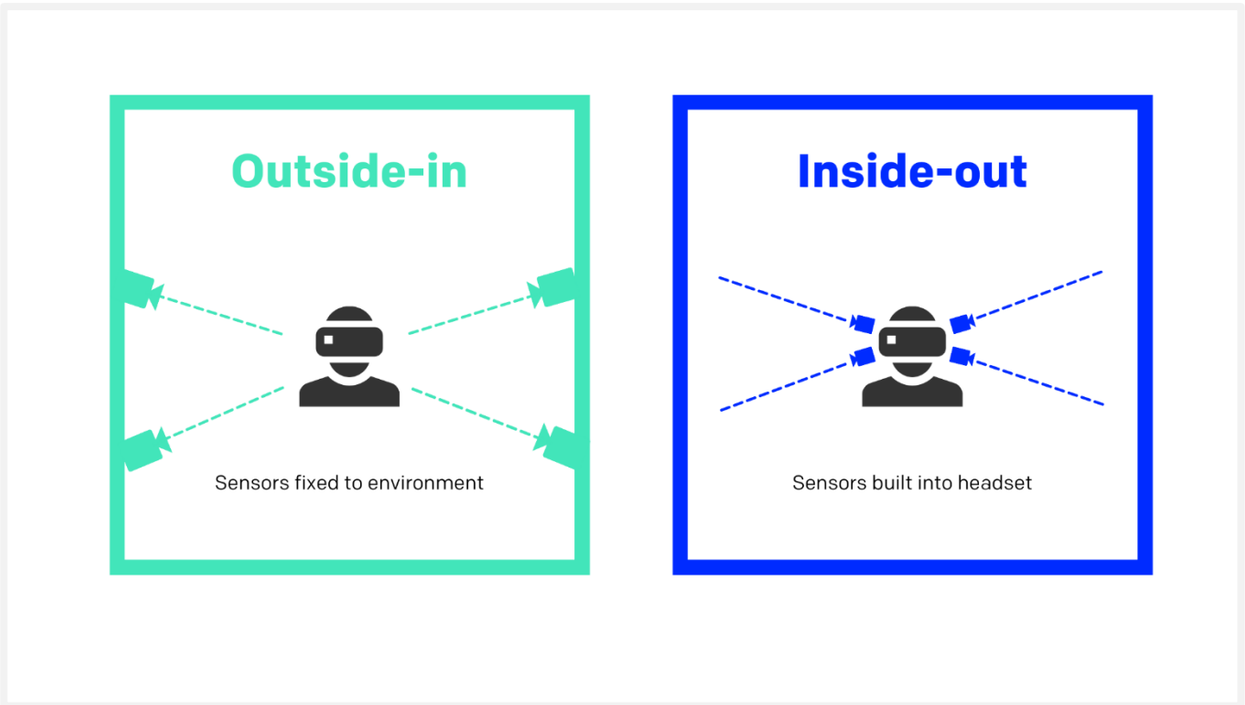
An example of *"chromatic aberration"* due to the internal lenses of a modern VR HMD.



HARDWARE: HEAD-MOUNTED DISPLAY (5)

VR HMDs rely on pose tracking that can be implemented in 2 ways: inside-out and outside-in.

- **Outside-in tracking** relies on external devices (aka base stations or *"lighthouses"*) constantly *"looking"* at the HMD and determining its pose (usually with 1 mm accuracy).
- **Inside-out tracking** relies on sensors and cameras on the HMD, *"looking"* at the surroundings to determine the HMD pose (lower accuracy than outside-in).



HARDWARE: HMD COMPARISON

THE WILD
IMMERSIVE COLLABORATION FOR TEAMS

2022 Business VR Headset Comparison Chart (Q1)

	Meta Quest 2	Pico Neo 3	HP Reverb G2	Valve Index	Vive Pro 2	Vive Pro
						
Official Support in The Wild	☑	☑	☑	☐	☐	☑
Resolution / Eye	1832 x 1920	1832 x 1920	2160x2160	1440x1600	2448 x 2448	1440x1600
Refresh Rate (HZ)	90/120	90	90	144	120	90
Field of View	100°	98°	114°	130°	120°	110°
Weight	503g	620g	544g	570g	850g	563g
Tracking	Inside-out	Inside-out	Inside-out	Base Stations (more equipment = more precise hand tracking)	Base Stations (more equipment = more precise hand tracking)	Base Stations (more equipment = more precise hand tracking)
Type	**Standalone** (no wires, less powerful processor) + **option to wirelessly stream or tether** to a PC with a cable	**Standalone** (no wires, less powerful processor) + **option to wirelessly stream** to a PC	**Tethered** (wired to your PC, more powerful, can run larger models)	**Tethered** (wired to your PC, more powerful, can run larger models)	**Tethered** (wired to your PC, more powerful, can run larger models)	**Tethered** (wired to your PC, more powerful, can run larger models)
Price	**$299**	💼 **$699**	**$599**	**$999**	**$1399** | 💼 **$1599**	**$1199** | 💼 **$1399**
Summary	A great standalone headset for personal or business use. What you lose in processing power you gain in easy setup and freedom of movement. AirLink and the Link cable makes this a great option for running larger models as well.	A fantastic Enterprise standalone (or optional PC-streaming) headset focused on privacy and control, with ability to deploy software through Multiple Device Managers.	An affordable, high-res, tethered headset for running large models from your PC.	A top-of-the-line gaming headset. Base stations and wires require more setup and configuration, but create a smooth and powerful experience in-headset.	A top-of-the-line gaming headset. Base stations and wires require more setup and configuration, but create a smooth and powerful experience in-headset.	An older but still powerful gaming headset. Base stations and wires require more setup and configuration, but create a smooth and powerful experience in-headset.

HARDWARE: HMD TECHNICAL SPECS

This is a summary of the main specs of current HMDs on the market (2022):

- **Cost:** 300-1600 $
- **Display Type:** usually dual OLED.
- **Display Resolution / Eye:** from 1440×1600 to 2448×2448.
- **Refresh Rate:** 90-144 Hz.
- **Field of View (FOV):** 98-130 degrees.

- **Audio:** from mobile audio to 360 degrees noise-cancelling headphones.
- **Latency:** from ~20 ms (smartphones) to ~ 0.01 ms (high-end HMD).
- **Optics:** from fixed Fresnel lenses to adjustable IPD (and varifocal lenses?).

HARDWARE: CAVE

CAVE: A Cave Automatic Virtual Environment (CAVE) is a VR system including a room-sized cube in which 3-6 walls are back-projected in order to create an immersive virtual environment, designed specifically for a multi-user experience.

Typically, CAVE users wear shoe covers and stereoscopic glasses (e.g., shutter glasses), and use VR controllers (e.g., a VR wand).

See: en.wikipedia.org/wiki/Cave_automatic_virtual_environment

HARDWARE: CAVE (2)

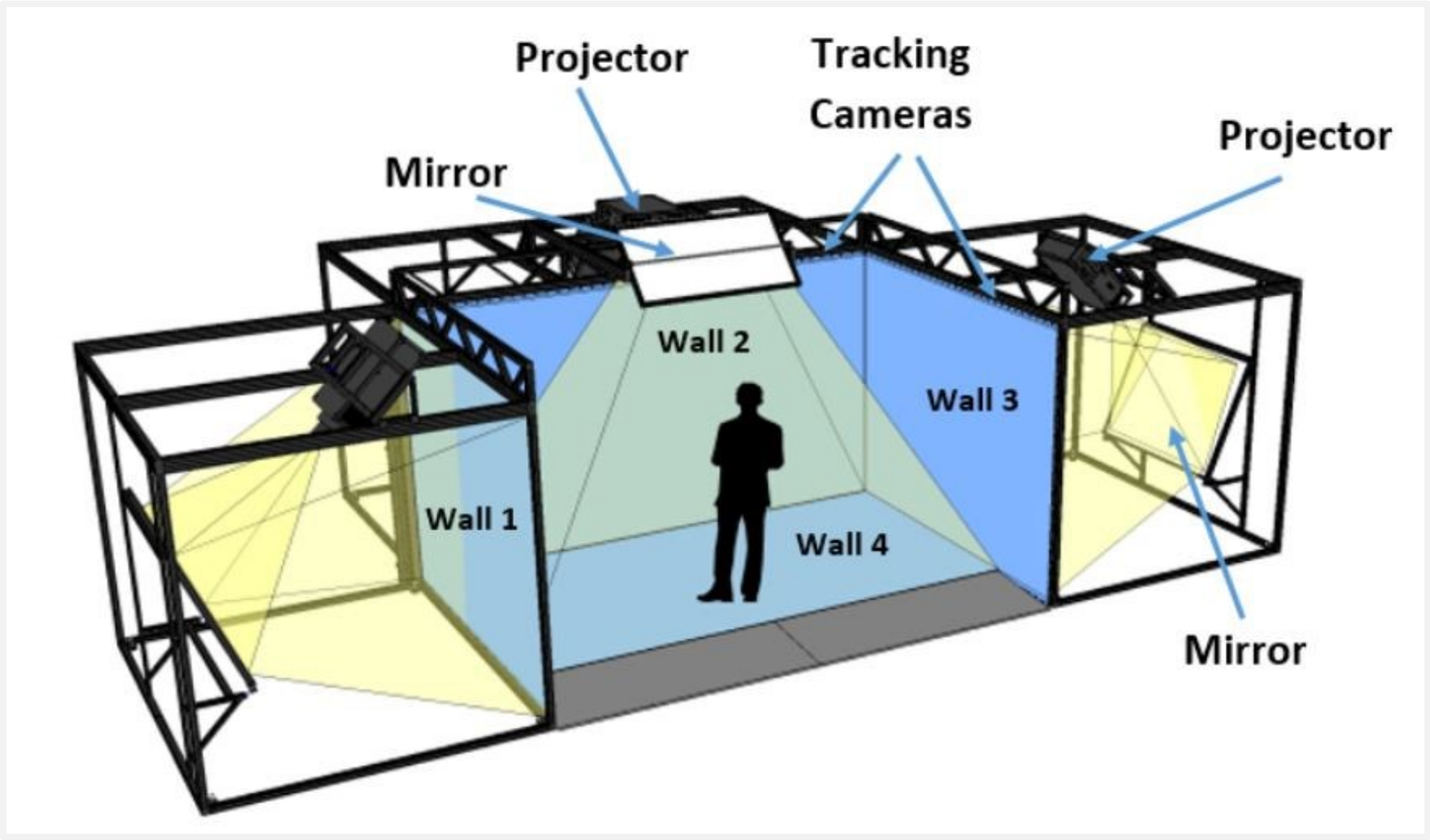
Usually, a CAVE relies on active shutter glasses to render for multiple users.

Active Shutter Glasses

Tethered glasses allowing the visualization of stereo pairs alternating the blocking of left/right images at high frequency, in synch with the visualization frequency and considering the proper eye.

HARDWARE: CAVE (3)

System diagram of a CAVE illustrating its main components (see NASA GRUVE Lab).



HARDWARE: CAVE TECHNICAL SPECS

This is a summary of the main specs of current CAVE systems on the market:

- **Cost:** starting at 50000 $.
- **Dimensions:** starting at 3×3×3 m.
- **Displays:** 4-5 projection screens, used with stereoscopic glasses (shutter glasses).
- **Users:** up to 5 users simultaneously.
- **Interactions:** using VR controllers (wands, data gloves, joysticks, etc.).

HARDWARE: VR CONTROLLERS

Standard input devices (mouse, keyboard etc.) are not user-friendly during a VR session.
So, VR controllers are the standard input devices used to interact with the virtual world.

This is a short list of the most common VR controllers:

- **Wand:** a hand-held joystick (see Nintendo Wii), often including a tracker.
- **VR Gloves (Data Gloves):** special gloves able to track hand-finger movements, often including basic tactile feedback (vibrations).
- **Hand-Finger Trackers:** trackers designed to track hand-finger movements (see Leap Motion), often integrated on HMDs to implement touch-less gestural interactions.
- **VR Controllers:** a pair of hand-held joysticks (see Oculus Touch Controllers), usually integrating both 6-DOF tracking and basic tactile feedback.
- **Haptic Stylus:** a stylus-like joystick attached to a simple robotic arm (see Sensable Phantom Omni), always integrating both 6-DOF tracking and haptic feedback.

HARDWARE: VR CONTROLLERS (2)

VR Controllers

A pair of hand-held joysticks, usually integrating both 6-DOF tracking and basic tactile feedback.

Oculus Touch Controllers

These controllers consist in a pair of handheld units, each featuring: 1 analog stick, 3 buttons, and 2 triggers.

Some versions also include: a dedicated thumbrest, and a system for detecting finger gestures.

The controller ring contains infrared LEDs to allow 6-DOF tracking.

HARDWARE: VR TRACKING

Pose Tracking: In VR a pose tracking system detects the precise pose (3D position and orientation) of HMDs, VR controllers, and other objects or body parts.

Pose tracking is often referred to as 6-DOF tracking, for the six degrees of freedom in which the pose is often tracked.

Note: Pose tracking is sometimes referred to as positional tracking, but it is an error!
Pose tracking tracks both 3D position and 3D orientation of a target.
Positional tracking only tracks the 3D position of the target (and no 3D orientation).

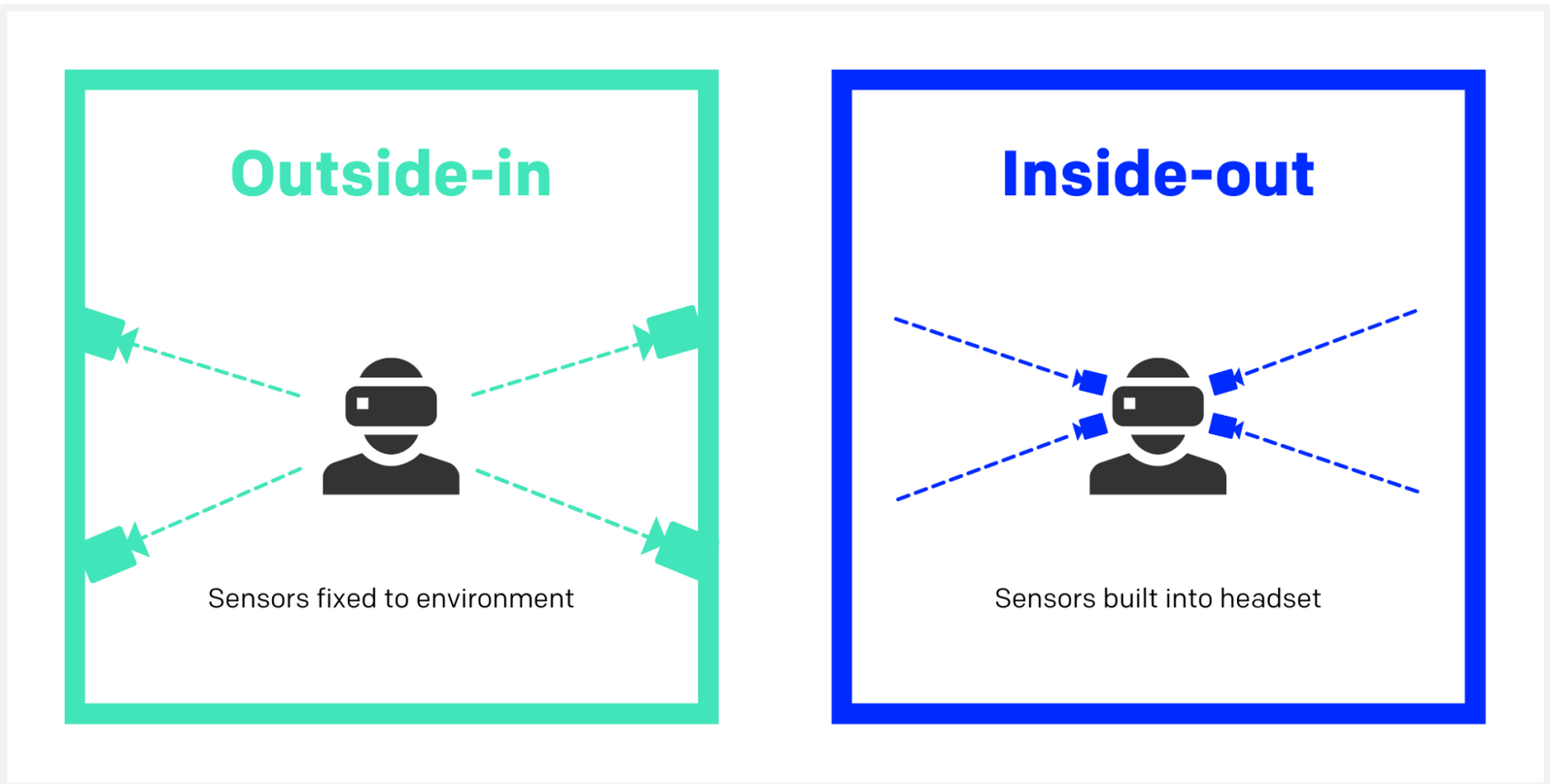
Outside-In Tracking: Tracking cameras are placed in static locations to track the position of markers on the target (HMD, VR controllers, etc.).

Inside-out tracking: Tracking cameras are placed on the target (HMD) and look outward to determine its location considering the surrounding environment.

See: en.wikipedia.org/wiki/Pose_tracking

HARDWARE: VR TRACKING (2)

Comparison between outside-in and inside-out tracking methods for VR.



See: *"Inside a VR Headset: Outside-In Tracking"* on YouTube.

See: *"Inside a VR Headset: Inside-Out Tracking"* on YouTube.

SOFTWARE: VR (GAME) ENGINES

A VR (game) engine provides software developers with a framework for creating a VR app.

Note: Current VR engines are game engines supporting VR devices, with some VR engines supporting also AR-MR app development; at this time there is no engine designed specifically for VR.

Usually, a VR (game) engine should provide:

- Functionality to design/develop/test VR apps.
- Functionality to design/create/edit 3D content for immersive VR experiences.
- Functionality to integrate VR hardware (HMDs, controllers, etc.).

At the moment (2021), the main VR (game) engines on the market are:

- **Unity:** coding in C#.
- **Unreal Engine:** coding in C++.

SOFTWARE: VR (GAME) ENGINES (2)

Nowadays, most VR applications are developed by using game engines.

A game engine is a software framework mainly designed for the development of video games. When VR technology became popular, these engines started including functionalities to support the design and development of VR software applications as well as the integration of VR interfaces (headsets and controllers).

These are some of the functionalities/modules typically included in a modern game engine:

- A 2D/3D **editor** to configure the game content.
- **Code libraries** to support game programming/scripting.
- A **rendering engine** (renderer) to visualize 2D/3D graphics.
- A **physics engine** responsible for rea-time interactive physics simulation.
- A **collision detection** module to manage collision events and collision responses.
- Then other modules for: sounds, animation, VR-AR, AI, networking, etc.

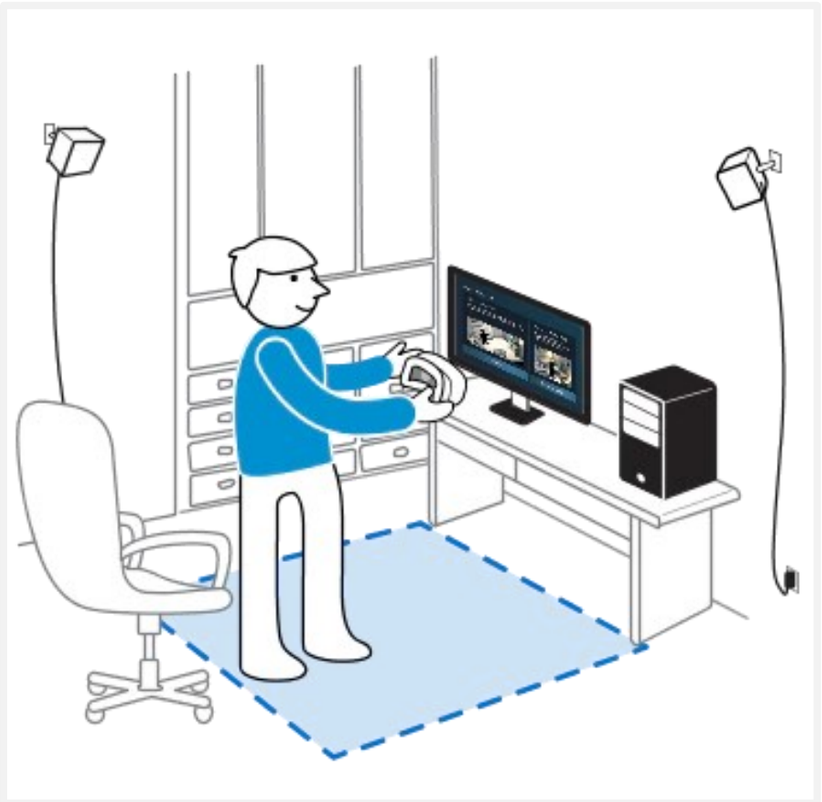
SOFTWARE: VR (GAME) ENGINES (3)

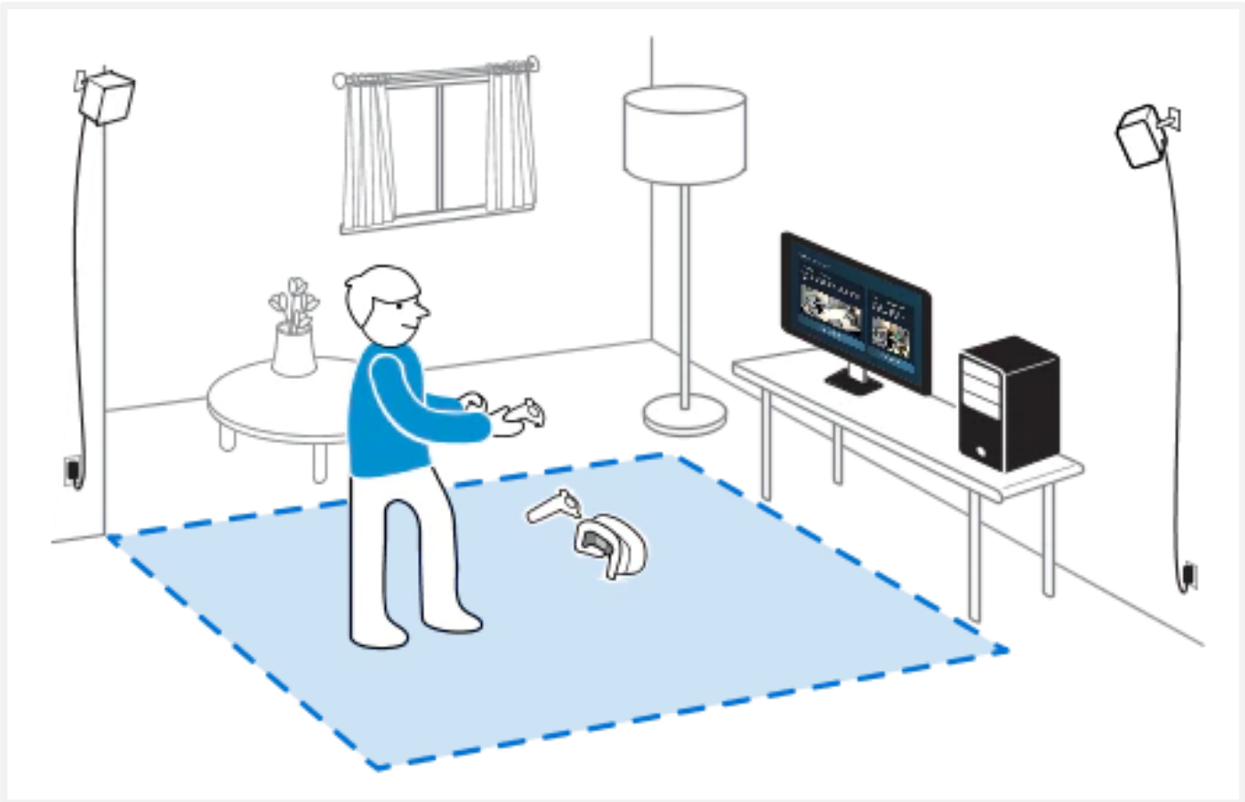
The main game engines on the market today (2024) are: **Unreal**, **Unity**, Godot, etc.

See: The *"Unity Editor"* software app part of the Unity game engine.

DESIGN: VR SYSTEM DESIGN EXAMPLE 1

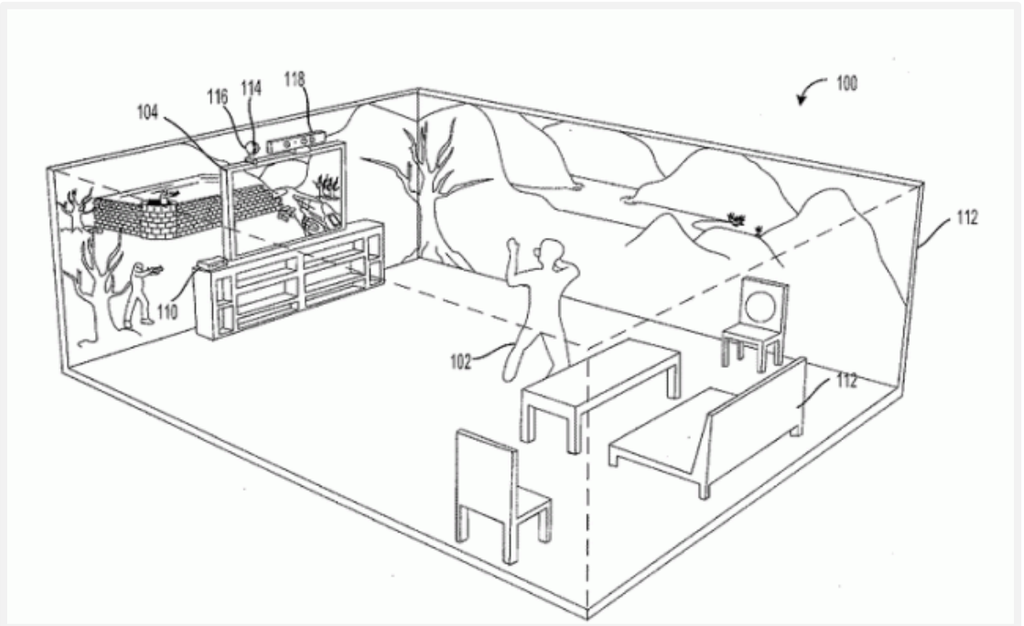
Side-by-side examples of VR system designs: VR workspace for single-desk (left), and VR workspace for full-room (right) (both using outside-in tracking and standalone VR HMDs).

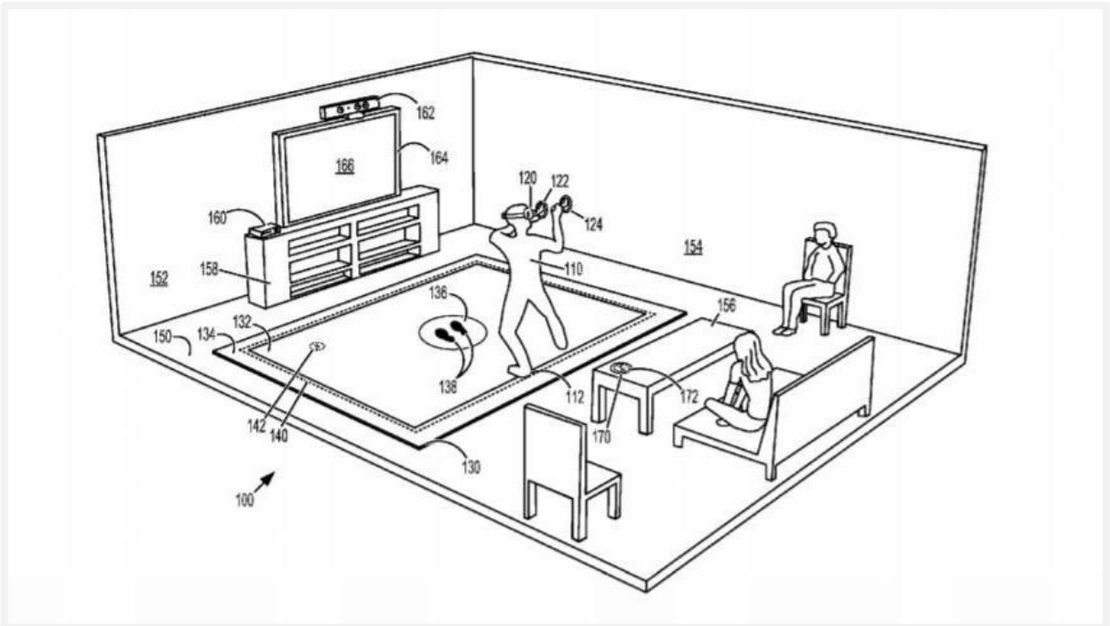




DESIGN: VR SYSTEM DESIGN EXAMPLE 2

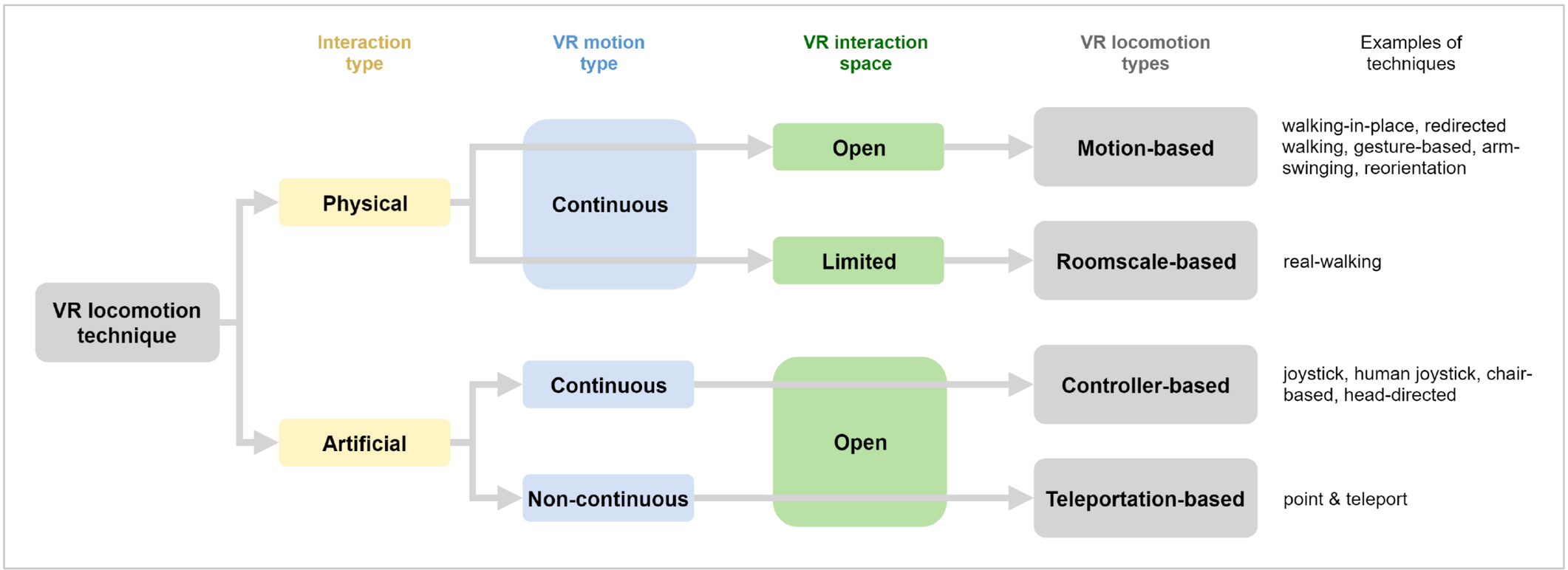
Side-by-side examples of VR system designs: VR workspace full-room CAVE-like (left), and VR workspace for full-room with gaming-mat (right) (both from Microsoft patents).





DESIGN: VR LOCOMOTION DESIGN

A proposed typology for VR locomotion including 4 distinct types of VR locomotion.



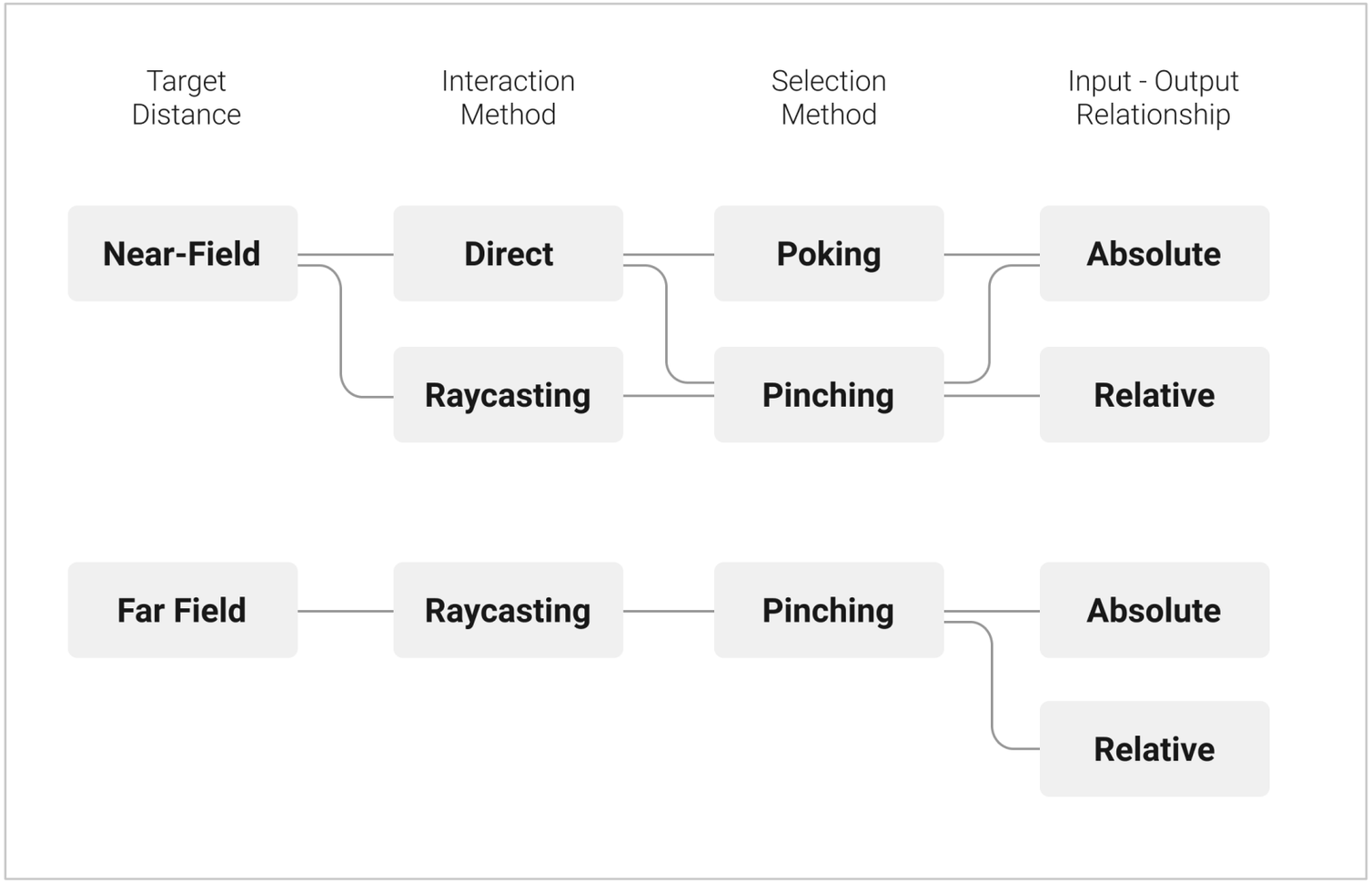
See: *"The New Era of Virtual Reality Locomotion"* by C. Boletsis.

DESIGN: VR INTERACTIONS/UI DESIGN

Interaction design in VR involves creating intuitive and engaging ways for users to interact with a 3D virtual environment.

Key considerations: 3D spatial interactions, affordances, signifiers, feedback, safety and comfort.

The diagram (right) shows **best practices for hand/finger interactions**.



DESIGN: VR INTERACTIONS/UI DESIGN (2)

The design and implementation of user interfaces (UIs) for VR includes:

- **Menus/Panels:** for specific views, in-game, integrated in 3D (diegetic), etc.
- **Heads-Up Displays (HUDs):** minimaps, timers, etc.
- **Controls/Interactions:** keyboard-mouse, point-and-click, joypad, etc.
- **Interaction Feedback:** visual (highlighting), audio (clicks), tactile (vibration), etc.

Designing UI for VR means:

determining which information/interactions should be available to the user,

and how these will be implemented

(usually maximizing comfort and usability).

DESIGN: VR INTERACTIONS/UI DESIGN (3)

Every interactable object (UI element) should *"hint"* at how to interact with it.
Moreover, in VR (usually) there is no/limited tactile feedback to confirm user actions.

These two issues are usually solved by reinforcing on all interactions the communication of affordances through clear signifiers and continuous feedback.

Affordances: What you can do with an object.

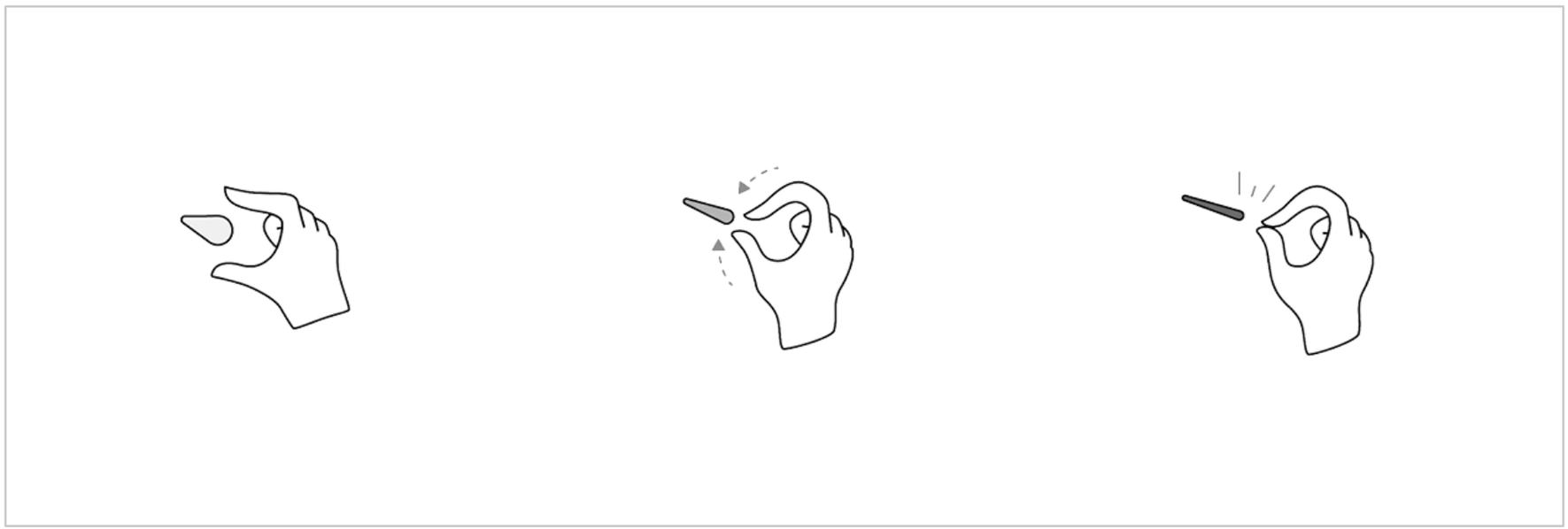
Signifiers: Communicate the relative affordances to the user.

Feedback: Confirm to the user the interaction/user state throughout the interaction.

Example: Seeing a door, the user assumes it affords opening (affordance); the handle (signifier) communicates this affordance to the user; then, during the interaction, the door/handle opening/turning (interaction) is continuously confirmed to the user just by looking at them (continuous feedback).

DESIGN: VR INTERACTIONS/UI DESIGN (4)

VR Example: When a user wants to interact with the VR system via pointing (affordance), the user can achieve a *"poiting state"* (left) that activates the visualization of a pointer (signifier, left). The pointer is squishable so that the user understands that **a *"click interaction"* is achieved by squishing the pointer enough.** Continuous feedback is provided by the squishing/color animation of the pointer through the point-and-click interaction.



DESIGN: VR INTERACTIONS/UI DESIGN (5)

A non-diegetic UI in a traditional videogame (first-person action shooter).

DESIGN: VR INTERACTIONS/UI DESIGN (6)

A VR UI designed for hand-finger interactions.

DESIGN: UI FOR VR (3)

These are some guidelines to design and implement UI for VR:

- Integrating UIs in the virtual environment (diegetic UIs) is the best design.
- Visualize reticles/crosshairs on the target (not at a fixed distance).
- Hide virtual instruments when not in use.
- Be careful with inconsistencies of virtual avatars.
- No standard input device is ideal in VR, the best option is a gamepad.
- Use familiar input devices (remember that the user does not see the controllers in VR).
- Be careful with the intense use of gestures and gaze (it can easily cause fatigue).
- Locomotion is still unexplored in VR (it can create issues never seen before).

DESIGN: VR BEST PRACTICES

This is a summary of the *"best practices"* to design and implement VR systems:

- Do not neglect monocular depth cues (e.g., illumination, details, etc.).
- The most comfortable depth range in VR is 0.75-3.50 m.
- Ensure that text elements in VR are easy to read and avoid small unnecessary elements in areas where the user focuses.
- The most comfortable VR experiences do not integrate any *"self motion"* for the user, excluding head/body movement to look around.
- If *"self motion"* is necessary in VR, slow user movements are the most comfortable.
- Ensure that any form of acceleration (of the user or other elements) is as hort and unfrequent as possible.

DESIGN: VR BEST PRACTICES (2)

This is a summary of the *"best practices"* to design and implement VR systems (continued):

- User head movements and virtual camera movements should always match!
- Do not integrate *"head bobbing"* in the VR avatar movements.
- To achieve comfortable VR experiences, minimize backward and lateral movements.
- Careful with situations inducing strong vection (visual feeling of motion), for example climbing stairs, accelerating, etc.
- Minimize (if possible) both *"lag"* and latency.
- If latency is unavoidable, a variable *"lag"* is worse than a constant *"lag"*.

Follow these guidelines to maximize comfort and usability of a VR experience, minimizing: visual fatigue, disorientation, nausea, etc.

PART 2: STEREO VISUALIZATION

This section briefly explores the essential elements of designing a VR system:

- **Hardware Components:** Visualization interfaces (headsets, CAVEs, etc.), VR controllers (gamepads, wands, data gloves, stylus, etc.), tracking systems (inside-out, outside-in, etc.), and audio devices (headphones, spatial audio, etc.).
- **Software Components:** VR engine (graphics, physics, etc.), UI (menus, diegetic interfaces, interactive elements, etc.), content creation tools (3D modeling, texturing, animation, etc.), interaction design (user interactions, navigation, etc.), sound design.
- **User Experience (UX):** Comfort, navigation, feedback, accessibility, testing, evaluation.
- **Content Creation:** Storytelling, narrative design, world building, asset management.

Some guidelines to design an immersive comfortable VR system will also be presented.

THEORY: BINOCULAR VISION

FOV: The field of view (FOV) is the extent (described as a solid angle) of the observable world that is seen by the user or a sensor.

Binocular Vision: A type of vision system of a human/animal that includes 2 eyes capable of facing the same direction (share the FOV) to perceive a single 3D image of the virtual/real world.

Monocular Vision: A type of vision system of a human/animal that includes only 1 eye or multiple eyes not facing the same direction (do not share the FOV).

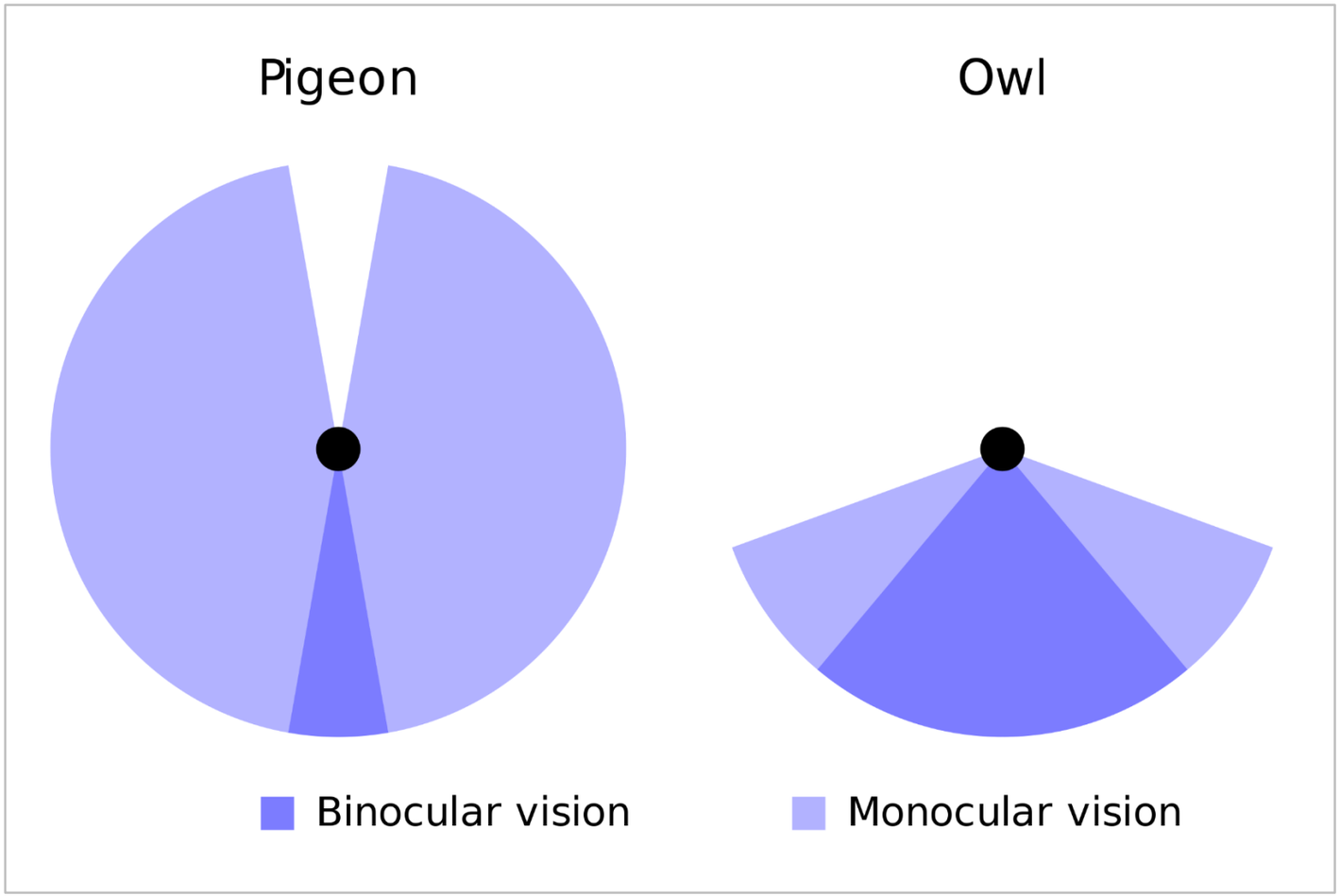
Example: The vision system of a horse is mainly monocular, its eyes are positioned laterally on its head, this arrangement provides the horse with increased field of view, but poor depth perception.

THEORY: BINOCULAR VISION (2)

The FOV of a pigeon (left) and of an owl (right).

Pigeons have 2 eyes arranged laterally on their head, whereas owls have 2 eyes positioned frontally on their head.

Different colors show FOV areas where binocular vision is enabled and where only monocular vision is available.



THEORY: DEPTH PERCEPTION

Depth Perception: The human ability to perceive distance (depth) to objects in the real/virtual world using a visual system. It is a major factor in perceiving the real/virtual world in 3D.

Depth Sensation: The animal (non-human) ability to perceive distance (depth) to objects in the real/virtual world using a visual system.

Note: It is known that non-human animals can perceive depth, but it is not known if it it the same perception as humans (vision systems could be different, etc.).

Stereopsis: The process performed by the human brain (visual cortex) allowing the depth perception when a scene is viewed with both eyes (binocular vision).

THEORY: DEPTH CUES

Depth perception is essential in VR systems,
and it is based on the correct generation of stereo pairs!

To understand how depth perception works we must consider that there are several *"depth cues"* utilized by our brain. These are the **(monocular) depth cues available in 2D graphics:**

- **Perspective:** objects (projections) become smaller if the distance from the point of view increases.
- **Relative Dimensions:** we expect certain dimensions of familiar objects.
- **Detail:** we can see more details of closer objects in respect to distant targets.
- **Occlusion:** occluding objects appear on the foreground in respect to occluded objects.
- **Illumination:** near objects are brighter than distant objects.
- **Relative Movement:** distant objects move slower than near objects.

THEORY: DEPTH CUES (2)

These are the **(binocular) depth cues only available in 3D graphics:**

- **Binocular Disparity:** difference between the left and right images of a stereo pair.
- **Accomodation:** muscular tension needed by our eyes to focus on a target.
- **Convergence:** muscular tension needed to rotate our eyes toward the focal point.

Binocular disparity is considered the dominant depth cue dominante!

All other depth cues, if incorrect, can have a severe detrimental effect!

THEORY: DEPTH CUES (3)

To achieve a correct depth perception, the stereo pair must be properly visualized so that the visual cortex can *"fuse"* it.

If a stereo pair is generated with conflicting depth cues:

- Any depth cue could become dominant (accidentally).
- Depth perception could be exaggerated or diminish (in respect to the correct one).
- The 3D content could be uncomfortable to view.
- The visual cortex could fail *"fusing"* the stereo pair (resulting in double vision).

If a stereo pair is generated correctly:

- Both binocular disparity and convergence are correct!
- Accomodation will always be inconsistent, but usually this issue is tolerated...

THEORY: STEREOSCOPY

Stereopsis: Depth perception produced by the reception in the brain (visual cortex) of visual stimuli (stereo images) from both eyes in combination (binocular vision).

Stereo Pair: A pair of views of a 3D object designed to be viewed independently by the left and right eyes of the viewer. Any type of stereoscopic system involves visualizing a stereo pair to the viewer in some way.

IPD: The inter-pupillary distance (sometimes referred as pupillary distance, or PD) is the distance in millimeters between the centers of each pupil.

Stereo Rendering: The computer graphics process to visualize 3D content by rendering a stereo image pair in order to induce the viewer to achieve stereopsis.

Stereoscopic System: A system (hardware and software) that presents the viewer with an image of a 3D object such that it appears to have *"real"* depth (enabling the viewer to have depth perception of virtual content).

THEORY: HORIZONTAL PARALLAX

Horizontal Parallax: The pixel-distance (in screen space) between the left and right projections in a stereo pair of an individual 3D point.

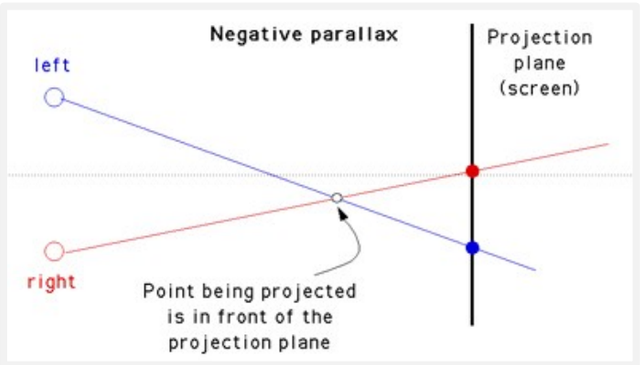
In a stereo pair, considering an individual 3D point, we have a positive horizontal parallax when the **left/right projections appear on the same side of the relative eye**.
The max positive horizontal parallax is associated with an indivitual 3D point at infinite distance from the camera.

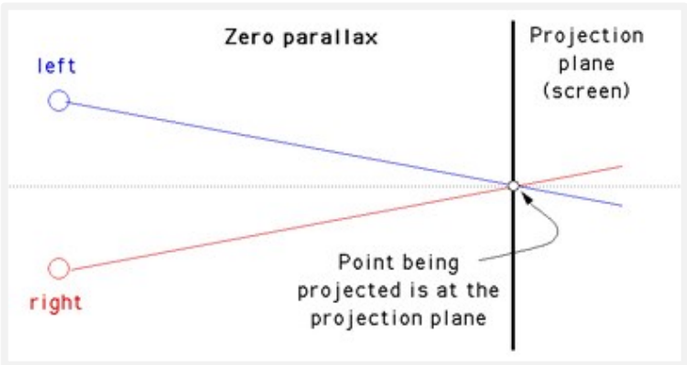
In a stereo pair, considering an individual 3D point, we have a negative horizontal parallax when the **left/right projections appear on the opposite side of the relative eye**.
The negative horizontal parallax equals the IPD if the individual 3D point is halfway between the projection plane and the eye center.

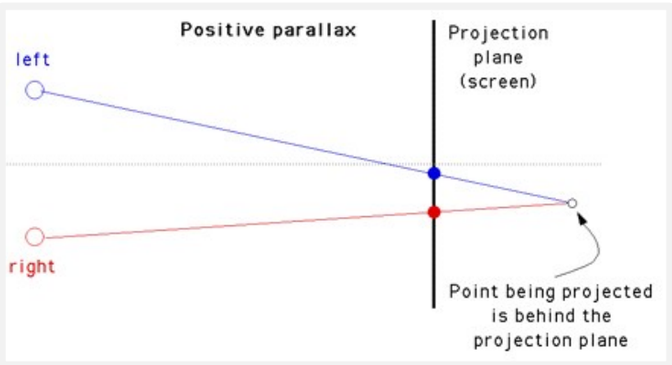
In a stereo pair, considering an individual 3D point, we have a zero horizontal parallax when **the individual 3D point is on the projection plane so that its left/right projections coincide**.

THEORY: HORIZONTAL PARALLAX (2)

Top-down views of left/right projections of an individual 3D point in a stereo pair with: negative horizontal parallax (left), zero horizontal parallax (center), and positive horizontal parallax (right).







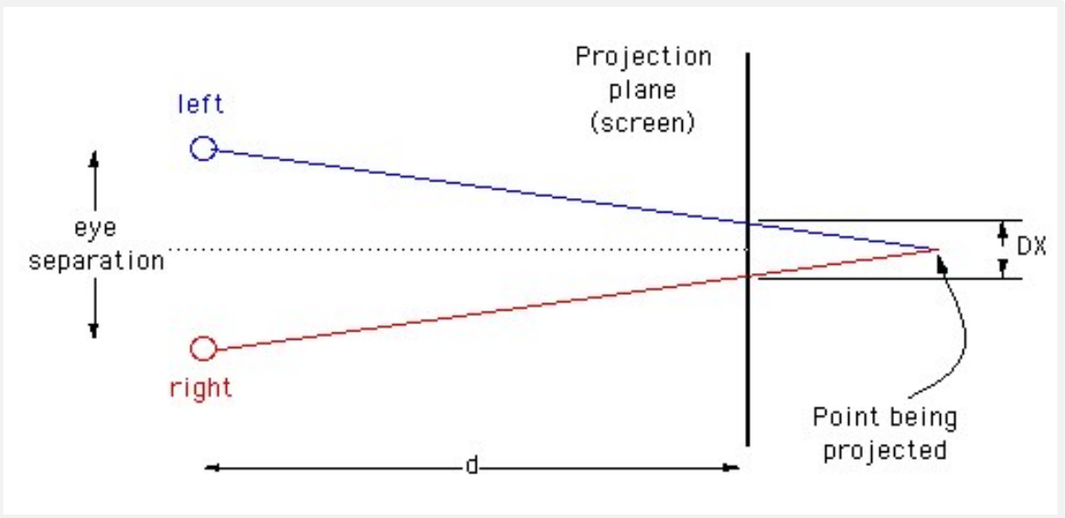
Note: A correct stereo pair should always present a zero vertical parallax for all 3D points. If a non-zero vertical parallax is present, then the stereo rig (configuration of the stereo virtual camera) is incorrect!

THEORY: HORIZONTAL PARALLAX (3)

In discussing horizontal parallax, it is useful to define the horizontal parallax angle.

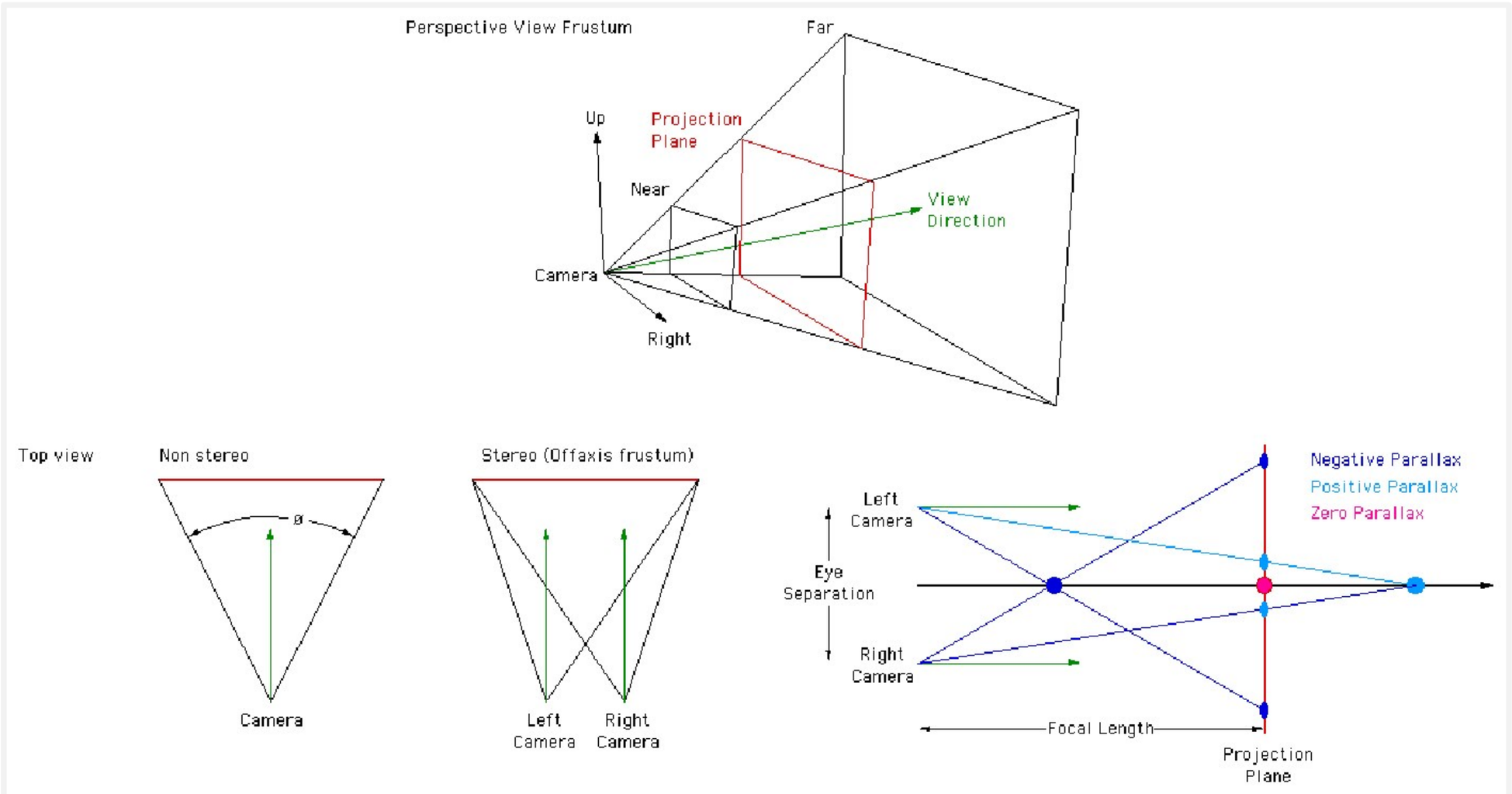
$$\text{horizontal parallax angle} = \theta = 2\,\text{atan}\left(\frac{\text{DX}}{2\text{d}}\right)$$

- theta is the horizontal parallax angle,
- DX is the horizontal parallax (positive in the left figure),
- d is the distance between eyes/cameras and projection plane.



Note: The horizontal parallax angle is positive for 3D points "behind" the projection plane, and negative for 3D points in front of the projection plane.

THEORY: VR STEREO CAMERA



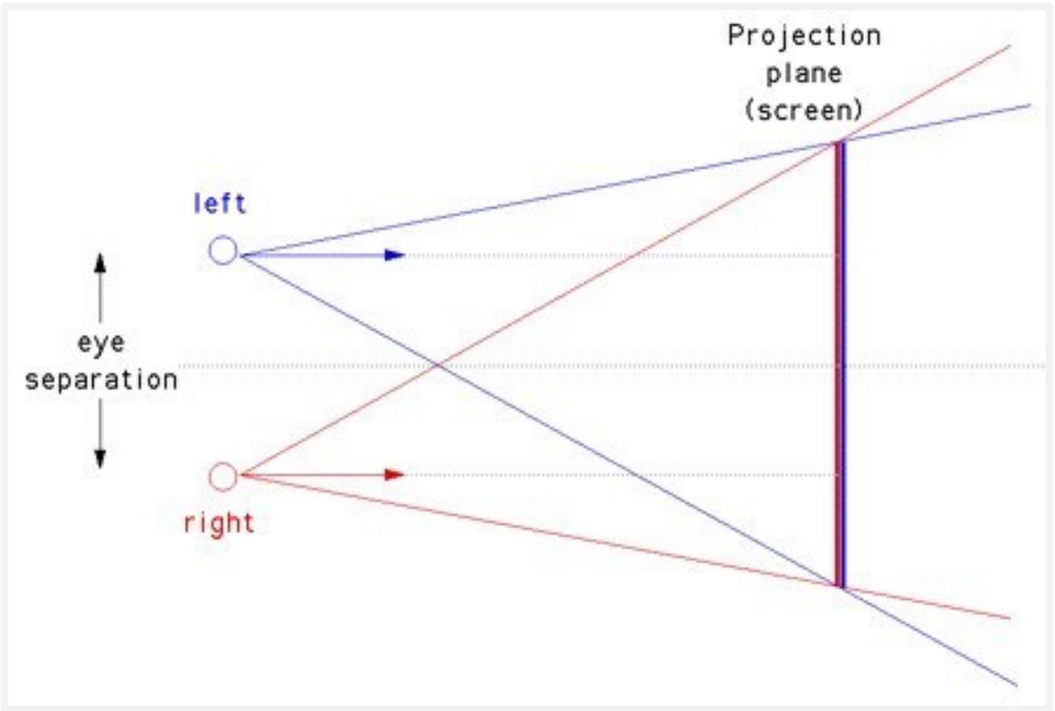
THEORY: VR STEREO CAMERA (2)

Off-Axis: VR stereo camera configuration optically correct.

The left/right cameras needs an asymmetric horizontal aperture, and are directed toward different focal points (pointing straight forward).

Stereo pairs created with an off-axis VR stereo camera do not include any non-zero vertical parallax (because the left/right projection planes are coplanar).

The off-axis VR stereo camera configuration needs asymmetric viewing frustums for the left/right cameras.



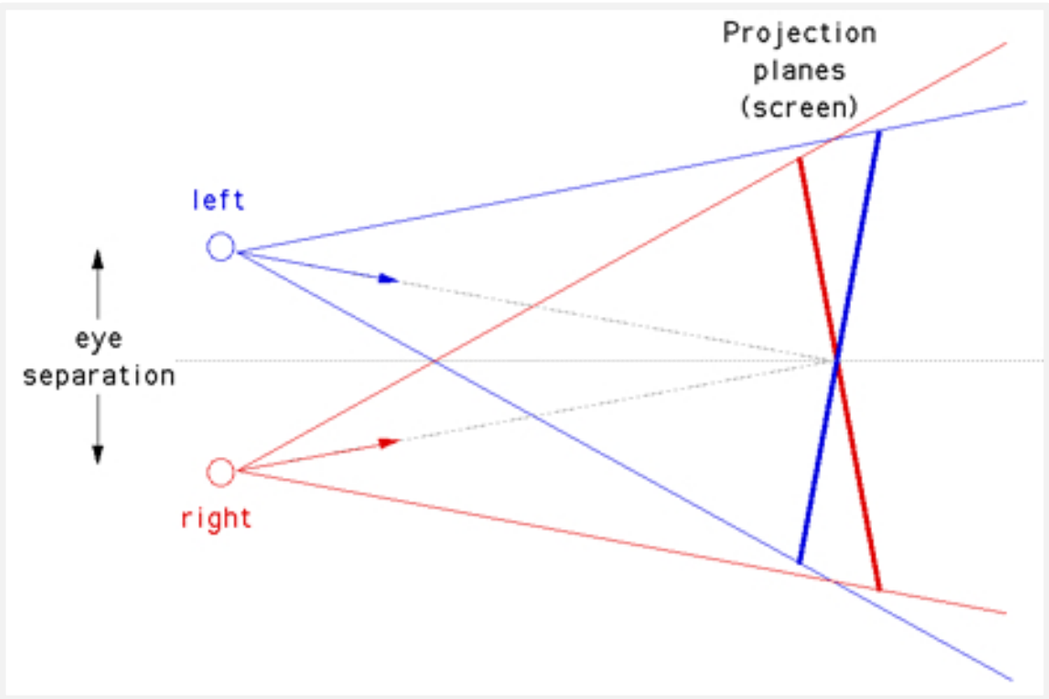
THEORY: VR STEREO CAMERA (3)

Toe-In: VR stereo camera configuration optically incorrect. However, this configuration is useful whenever asymmetric viewing frustums are impossible.

In the toe-in configuration, the left/right cameras use a symmetric horizontal aperture, and are directed toward the same focal point (pointing inward).

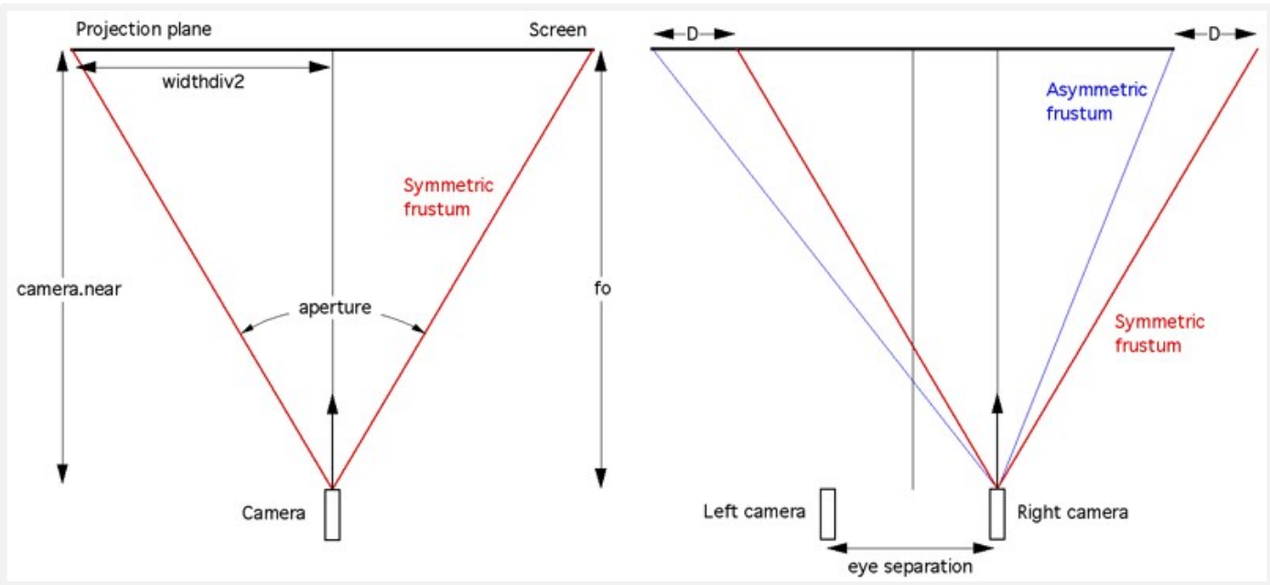
Stereo pairs created with a toe-in VR stereo camera cannot avoid non-zero vertical parallax (because the left/right projection planes are not coplanar).

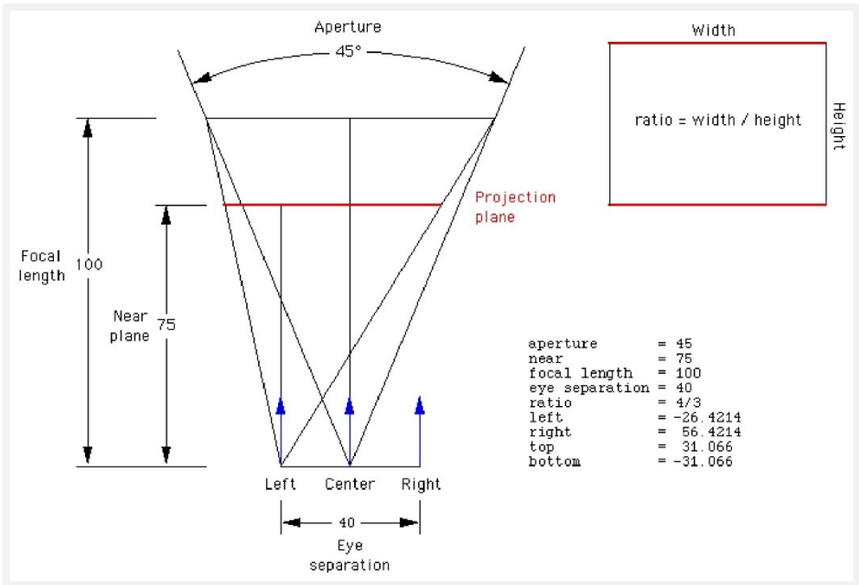
The vertical parallax increases from the image center toward the border, and if the camera horizontal aperture increases, and make the stereo pairs generated uncomfortable to view



THEORY: VR STEREO CAMERA EXAMPLE

An example of off-axis VR stereo camera configuration.





THEORY: STEREO RENDERING GUIDELINES

Summary of the main concepts for a correct comfortable stereo rendering:

- 3D objects in front of the projection plane will appear in front of the physical display.
- 3D objects behind the projection plane will appear *"inside"* the physical display.
- Usually, it is easier to fuse stereo pairs of 3D objects appearing *"inside"* the physical display; so, the focal point should be placed nearer to the virtual stereo camera than the target 3D objects.
- To facilitate fusing stereo pairs, the absolute value of the horizontal parallax angle (theta) should be smaller than 1.5 degrees for all individual 3D points.

THEORY: STEREO RENDERING GUIDELINES (2)

Summary of the main concepts for a correct comfortable stereo rendering **(continued)**:

- The magnitude of depth perception depends on both (1) the distance between the virtual camera and the projection plane, and (2) the left/right camera separation.
- A left/rigth camera separation too high (aka hyperstereo) could cause the generation of stereo pairs that are difficult to fuse (discomfortable viewing).
- A good approximation for left/right camera separation is 1/20 of the distance between the camera and the projection plane (this can also be considered the max camera separation for a comfortable stereo viewing).
- It is a good practice to ensure that the negative horizontal parallax never exceeds the left/right camera separation.

THEORY: STEREO RENDERING GUIDELINES (3)

Summary of the main concepts for a correct comfortable stereo rendering **(continued)**:

- The aperture of the virtual camera should equal the *"sweet spot"* of the VR headset.
- Properly select the focal distance (associated with individual 3D points with zero horizontal parallax) considering the 3D content, to avoid 3D object too close to the virtual camera (nearer than half the focal distance).
- Using a left/right camera separation similar to the average IPD (human): (1) we can achieve a realistic sense of scale and distance in VR, (2) we could obtain weak dept perception for 3D content distant from the virtual cameras.

PART 3: VR SW DEVELOPMENT IN UNITY

"Unity is so much more than the world's best real-time development platform – it's also a robust ecosystem designed to enable your success. Join our dynamic community of creators so you can tap into what you need to achieve your vision."

UNITY TECHNOLOGIES

Applicazion Fields: videogames, architecture, automotive, movies, XR, etc.

"Create once, deploy across 25+ leading platforms and technologies to reach the largest possible audience."

UNITY TECHNOLOGIES

See: unity.com

REFERENCES

- These slides are available online at: github.com/turinig/vrphd
- Unity website: unity.com
- *"Virtual Reality"* S. M. LaValle: lavalle.pl/vr
- *"Calculating Stereo Pairs"* P. Bourke: paulbourke.net/stereographics/stereorender
- *"Best Practices for Immersive VR"*
- *"How to Design for Virtual Reality: Basics and Best Practices for VR Design"*
- *"Virtual Reality: Introduction"*
- *"Getting Started with VR for Your Architecture & Design Team in 2022"*
- *"Pose Tracking Methods: Outside-In Versus Inside-Out Tracking in VR"*
- *"VR Design Best Practices"*

APPENDICES

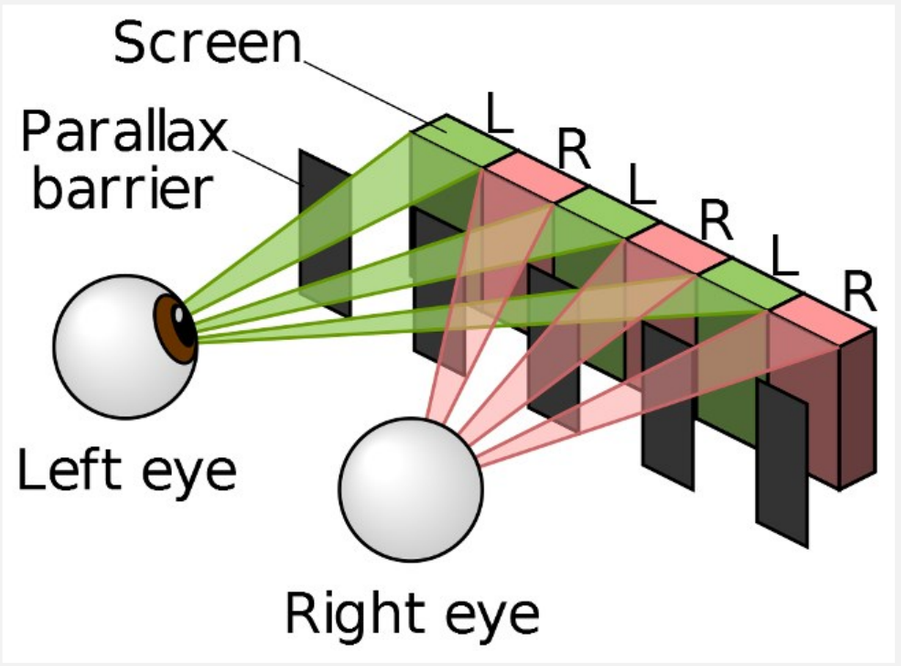
- Additional VR Hardware: Anaglyph Glasses, Parallax Barriers, and Lenticular Lenses
- Topics Related to VR Systems: Serious Games, Real-Time Interactive Simulators

HARDWARE: ANAGLYPH GLASSES

Anaglyphs rely on the encoding of a stereo pair into a single image exploiting color filters, and then using special glasses (**anaglyph glasses**) including the proper color filters to block/enable (decoding) the proper left/right image to the relative eye.

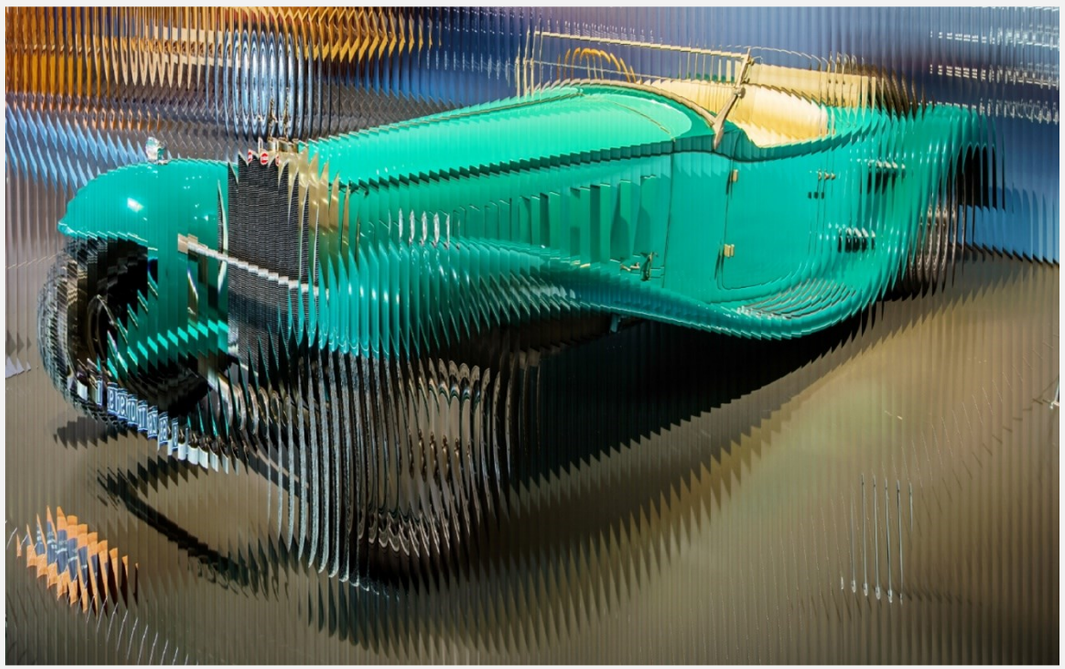
HARDWARE: PARALLAX BARRIER

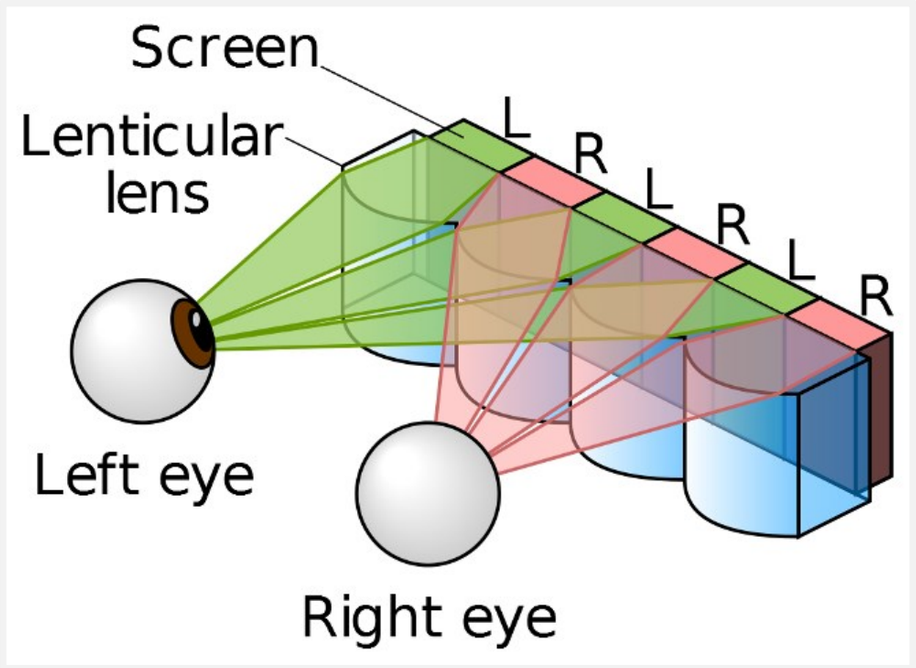
A **parallax barrier** is an autostereoscopic system allowing the visualization of stereo pairs without the need of additional optical devices. The image visualized integrates the stereo pairs interlacing the left and right images, that are delivered them to the proper eye using a semi-transparent barrier exploiting horizontal parallax.



HARDWARE: LENTICULAR LENS

Lenticular lenses are an autostereoscopic system allowing the visualization of stereo pairs without the need of additional optical devices. The image visualized integrates the stereo pairs interlacing the left and right images, that are delivered to the proper eye using a semi-transparent lens sheet to steer the view properly.





SERIOUS GAMES VS INTERACTIVE SIMULATORS

Serious games are games that have an *"...explicit and carefully thought-out educational purpose and are not intended to be played primarily for amusement..."*, but this *"does not mean that serious games are not, or should not be, entertaining..."*

CLARK C. ABT. SERIOUS GAMES. 1970.

Real-time interactive simulators are serious games consisting in 2D/3D graphics-oriented simulations that can realistically replicate real-world events, while providing a high degree of interactivity and *"real-time"* system reactions.

Note: In most cases there is not a clear distinction between a serious game and a real-time interactive simulation. For example: a serious game can integrate (and usually does) elements of real-time interactive simulation, and viceversa.

A SERIOUS GAME EXAMPLE

See: *"Vital Signs: Emergency Department"* medical interactive trainer by BreakAway Games

REAL-TIME INTERACTIVE SIMULATION

A real-time interactive simulation is a digital simulation, usually graphics-oriented, designed to be interactive and performed with discrete-time. In particular:

- These are **graphics-oriented** simulators, including 2D or 3D interactive visualization.
- These systems can **realistically simulate** natural phenomena and real-life events.
- These simulators are **highly interactive**, including complex user interfaces.
- These softwares can react in real-time, so **performance optimization** is critical.
- These simulators integrate elements of: **game development, AI, HCI, networking**, etc.
- These systems can be designed for **single-player or multi-player** environments.

Real-time interactive simulation involves a multidisciplinary approach, drawing from: computer science, computer engineering, mathematics, and physics.

REAL-TIME INTERACTIVE SIMULATION (2): DISCRETE TIME

In a real-time interactive simulator the simulation is performed with discrete-time.

Usually, the simulation discrete-time step is constant, that is: time moves forward in steps of equal duration, and this is commonly known as fixed time-step simulation.

However, in some cases, a variable time-step simulation can also be used.

Fixed time-step simulation is preferrable in some situations, whereas variable time-steps simulations provide better results in others.

Example: In an interactive video game integrating some physics simulation, the graphics visualization runs at a variable time-step (frame rate) while the physics updates run at a fixed time-step (usually ~50 Hz).

Example: In a real-time interactive simulator integrating physics and haptics, the physics updates run at a fixed time-step (usually ~50 Hz) while the haptic rendering runs at a different fixed time-step (usually ≥1 KHz).

REAL-TIME INTERACTIVE SIMULATION (3): ONLINE OR OFFLINE

To solve the mathematical equations at a given time, each simulation variable is solved successively as a function of variables of the previous time-step.

In a discrete-time simulation, the time required to solve all mathematical equations at a give time may be shorter or longer than the duration of the simulation time-step.

If the computing time to solve all mathematical equations (update time) is shorter than the simulation time-step, the simulation is called accelerated simulation (not real-time).

If the computing time to solve all mathematical equations (update time) is longer than the simulation time-step, the simulation is called offline simulation (not real-time).

If the computing time to solve all mathematical equations (update time) equals the simulation time-step, the simulation is called real-time simulation (or online simulation).

REAL-TIME INTERACTIVE SIMULATION (4): MULTITHREADING

In a real-time interactive simulator the simulation is usually performed in a multi-threaded environment (*i.e.*, multiple software modules executed concurrently), allowing the decoupling of simulation tasks with different requirements.

In particular:

- Multithreading allows **optimal use of multi-core CPUs**.
- The **graphics rendering** thread usually runs at a variable frequency (frame-rate, >30 Hz), depending on the content visualized, and that should be constantly maximized.
- The **physics simulation** thread usually runs at a constant frequency (physics updates, >50 Hz), depending on the mathematical equations solved.
- The **collision detection** thread is usually integrated in the physics thread.
- The **haptics feedback** thread usually runs at high constant frequency (>1 kHz), to provide appropriate tactile/force feedback.

REAL-TIME INTERACTIVE SIMULATION (5): -IN-THE-LOOP

The capability to solve simulation equations in real-time enables different types of testing:

- **Hardware-in-the-Loop (HIL):** A testing technique for complex real-time embedded hardware modules, by enabling their interactions with a simulated system.

  For example: An hardware-in-the-loop test can simulate a car engine interacting with the ECU (engine control unit) with real inputs/outputs to enable the ECU testing.

- **Human-in-the-Loop:** A testing technique for complex real-time systems, by using interactive simulation involving always a human user that influences its outcomes.

  For example: A human-in-the-loop test can simulate an airplane cockpit interacting with the pilot with real inputs/outputs to enable the pilot testing/evaluation/training.

- **Software-in-the-Loop (SIL):** A testing technique for software algorithms, by simulating their interaction with a software environment/platform.

  For example: A software-in-the-loop test can simulate a self-driving car interacting with an AI algorithm with real inputs/outputs to enable the testing of its code.