

Intro

Types of VR

HMD, Flat screens

Key elements of VR

Immersion and interactivity

VR definition

Computer generated simulation of a 3d image or environment that can be interacted with by using special electronic equipment.

Artificial environment that is created with software that is created with software and presented to the user in such a way that the user suspends belief and accepts it as real.

Immersive, Artificial, Interactive

AR definition

Combines real and virtual world, interactive in real time, registered in 3D (real and virtual objects are in 3D relation to each other)

AR requirements

Accuracy of tracking

Registration like mentioned above

Precise, accurate, fast and robust tracking and registration

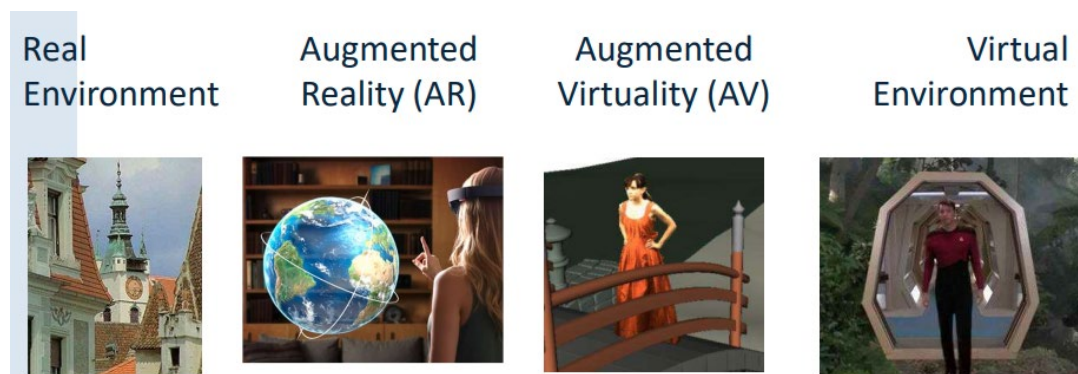
Types of AR

Handheld, Head-mounted and spatial

Mixed reality definition

Any combination of real and virtual world

Milgram's Reality-Virtuality Continuum



History of VR

1956 – Sensorama by Morton Heilig

Mid 90s VR hype

How VR got to the mass market

Mostly advances in other tech, such as:

2000s – thin low cost displays, natural feature tracking and mapping, affordable 2D and 3D cameras (Kinect), CPU/GPU + High speed networking

2012s – Oculus, PrioVR, Cyberith Virtualiser, Perception Neuron MoCap

2016s – Mass market with HTC, Meta, ...

Application Areas VR

Industrial (real time simulation, ergonomics...)

Design studies, Production chain simulation, modelling, simulation...

Visualisation (scientific, medical, information vis)

Entertainment

Training & Education

Rehabilitation & Therapy

Modelling & Design

Visualisation in VR/AR – Why?

Scientific simulations of things too big/small or impossible to do in real world

Abstract content visualised in an understandable way

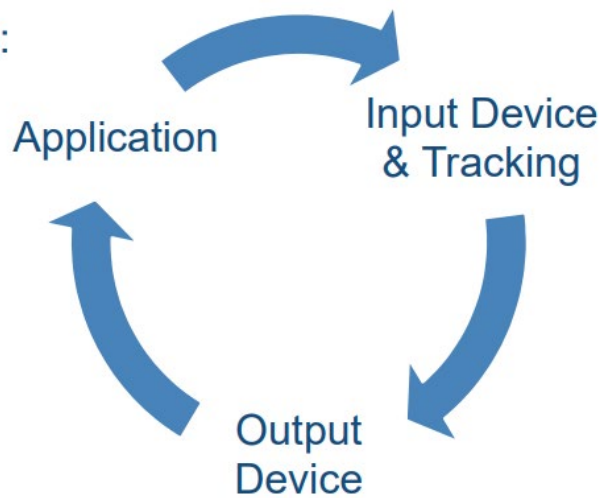
Collaboration on complex 3D data possible

Multiple and different forms of presentations (3D views)

Input devices

The Interface Problem

Dependencies:



There isn't a single ideal solution for every application

Input devices

Are hardware that allows the user to interact with the system, as opposed to interaction techniques which they implement

Classification of IDs

Discrete vs Continuous

Button vs tracker

Continuous is sampled obviously

There are hybrids like mice

But also miscellaneous like speech, gestures and locomotion devices

Degrees of Freedom

2DOF – mouse

3DOF – orientation (rotation relative to coordinate system axes) – roll/pitch/yaw

6DOF – position + orientation on all axes

Active vs Passive

Purely active requires user interaction like mouse

Passive generate data continuously on their own like motion tracker

Technological principle

Classification by operating type – mechanical, magnetic, optical, inertial, time of flight, hybrid

Devicetypes

Desktop

Keyboard, mice, trackballs, joysticks, pens(absolute input as opposed to relative mice that calculate your move on the desk and translate to the screen), 6-DOF device for desktop, haptics

Symbolic

Texts, Numbers in VR – annotations, filenames, labelling, object manipulation, user communication

Keyboards and the like usually don't work in 3D, hence, symbolic is less frequent in 3D than 2D

Gesturebased

By using gloves, computer vision or motion capture

Fast Gesture recognition needed

More vision based approaches today

Speech input

Tracking

Measure position and/or orientation of a sensor usually in 6DOF in space

Most track head and hand and hence give correct viewing perspective and the possibility to interact

Quality factors are:

DOF (3D pos + orientation = 6DOF)

Range and working volume

Amount of space where tracker works and the volume resulting from ranges + time till stability degenerates (drifting of inertial trackers)

Accuracy – Static/dynamic

Static – Maximum deviation from fixed tracker position to fixed reference value – this is influenced by receiver sensitivity, Transmitter s/n ration, A/D converter resolution ...

Dynamic – Accuracy as sensor is moved – influenced by processor type, system architecture, time dependent system components

Time for measurement – phase lag, “real-time”

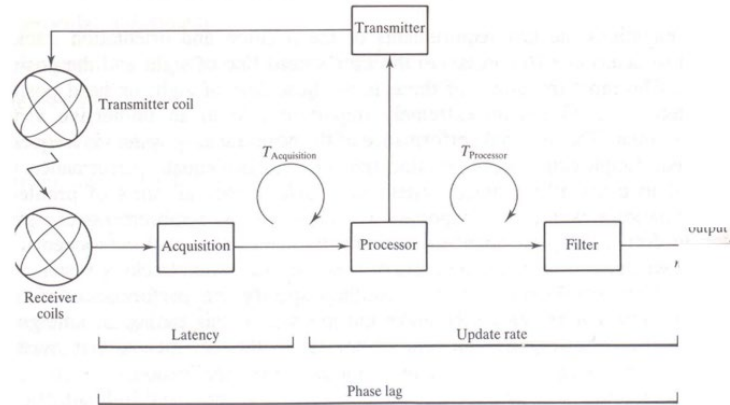


Figure 6.11 Generic architecture for a head tracker system.

The phase lag is the combination of latency and update rate

Latency is the time it takes from receiving a signal to processing it – eg the time to take a picture for an optical system

UR is the time it takes from the start of processing to the output – time to process image & extract data

This leads to the issue that when the phase lag is too high, fast motions can't be accurately tracked, as the tracker needs longer to update than the actual motion is and "misses" it

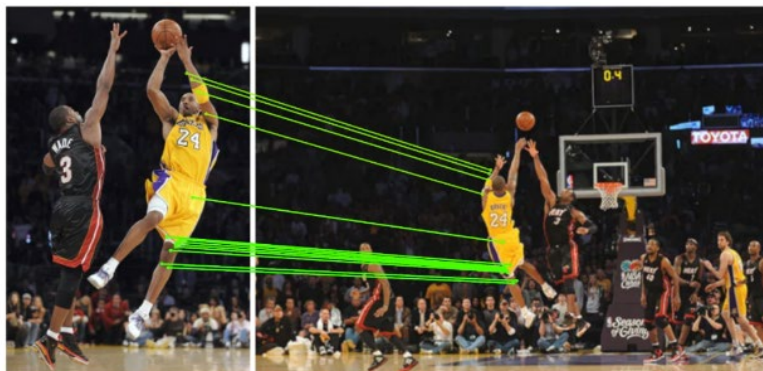
Asynchronous time warp helps with this – it predicts the next frame based on the last one

Update rate (measures/sec)

Signal to noise ratio

Registration

Correspondence – actual position / orientation = reported position / orientation – This is also important with multiple trackers, the pixels in one tracker must correspond with the ones in another to capture a position well enough



These pixel pairs are the **virtual correspondences**, from which the layout of the scene's cameras and geometry can be reconstructed

Good accuracy/resolution alone does not imply good registration

Important for more than one sensor

Important usability factor – can cause cyber sickness

Tracking shouldn't hinder freedom

Effects of inaccurate tracking

Objects appears where they aren't

Proprioceptive conflicts

Static – limb location conflicts

Dynamic – visual delay (lag)

Limb jitter or oscillation

Misregistration

Constant misregistration is OK (immersive, only when non see-through)

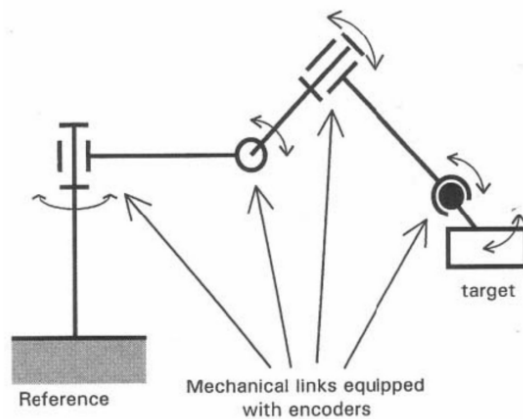
Changing breaks immersion

User has to adapt

Simulator sickness (Cybersickness)

Classifications

Mechanical



Oldest tracking technology

Measure angle of joints

Potentiometers for angular measurements

Position reconstruction using kinematics

+well known, fast, accurate

-Needs ground base -> limited working volume, uncomfortable, exoskeleton affects movement and needs to be adjusted per person

Magnetic

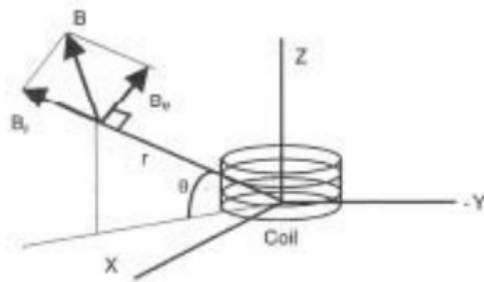
Circulating electric current in coil \rightarrow magnetic field is generated

At distance r , the field has polar coordinates B_θ and B_r

Magnetic flux is created in receiver (magnetic field sensor)

Magnetic flux is a function of distance and orientation relative to the coil

To measure position and orientation of receiver in space, emitters consist of 3 orthogonal coils and receivers of 3 sensors -Y combination of three elementary orthogonal directions



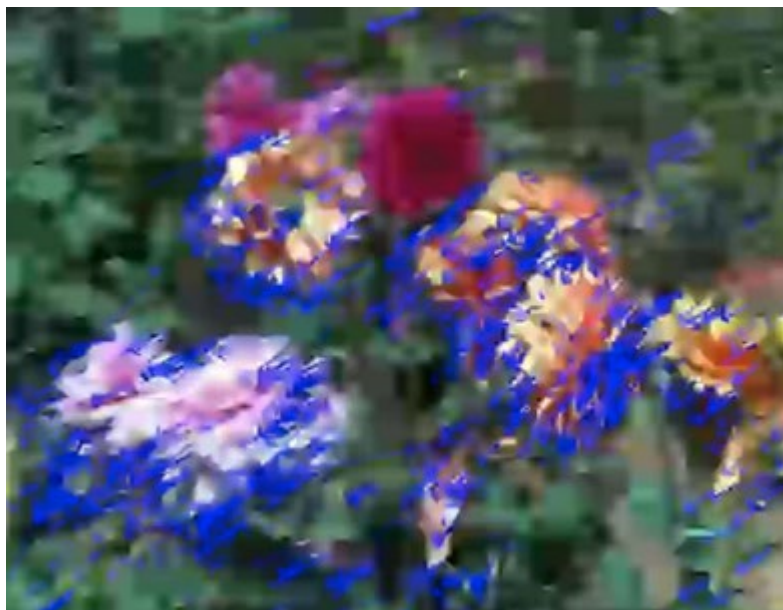
+ No line of sight restrictions \rightarrow continuous data, small and lightweight sensors, wireless versions, off the shelf availability, robust, high update rate, inexpensive

-Noise from metal and power wires, unwanted eddy currents (conductive material will distort field), field strength $1/d^3$, not best accuracy, limited working volume

Optical

Optical Flow Motion Estimation

Easy to implement watch pixel movement from one to the next image, then you compare the move vectors and indicate the movement direction – this is not very accurate though



Vision based tracking

Marker based

Natural Features

Simultaneous Localisation and Mapping (SLAM)

Vision Trackers

Types

Fiducials(QR-codes), ARTag, Intersense

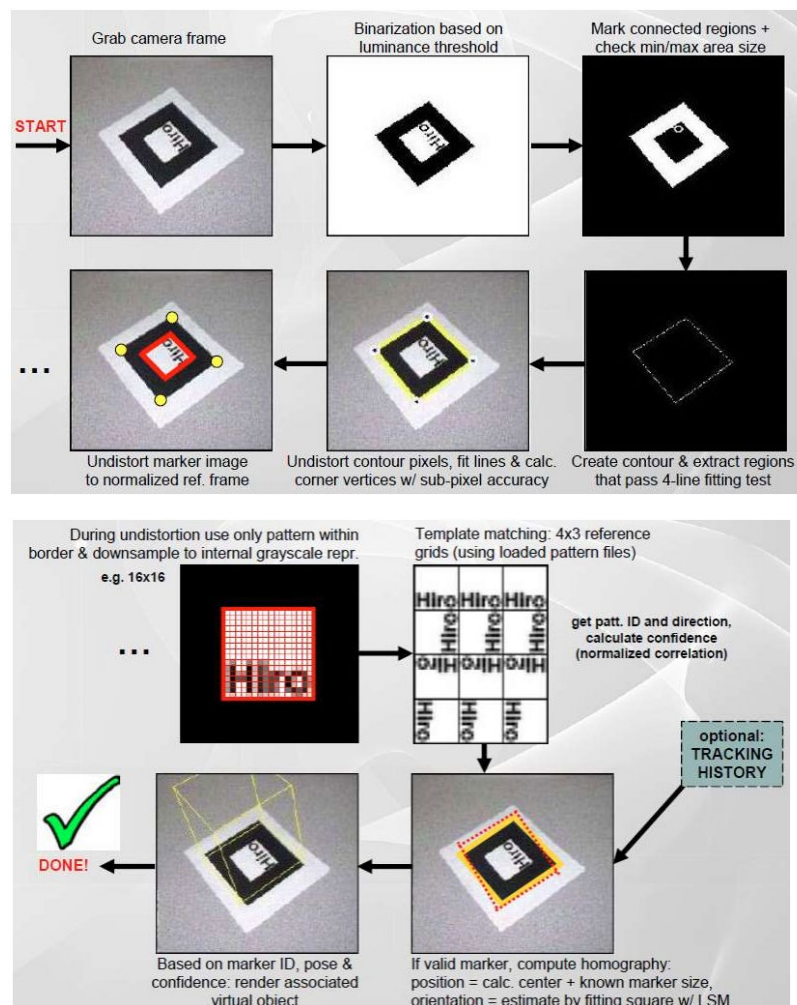
3D-Markers/LEDS

Natural Features

Fiducial Marker Tracking

Usually detects corners in rectangular shape

Pipeline:



The pattern checking uses the four corner points and uses a direct linear transform algorithm

Challenges here are:

Occlusion, unfocused cameras, bad light, jittering, false positives on makers (marker confusion), image noise, reflection...

A solution for this are active markers that emit infrared light or passive markers that reflect light

Inertial

Time of Flight & Frequency Measuring

Hybrids

Inside-Out / Outside-In Tracking

Inside-Out

Sensors are located on mobile target

Higher resolution and accuracy in orientation than outside-in because it produces larger motion in image -> larger displacement of pixels

Outside-In

Sensors (Cameras) are located at fixed reference points

Steps optical Tracking

Camera calibration

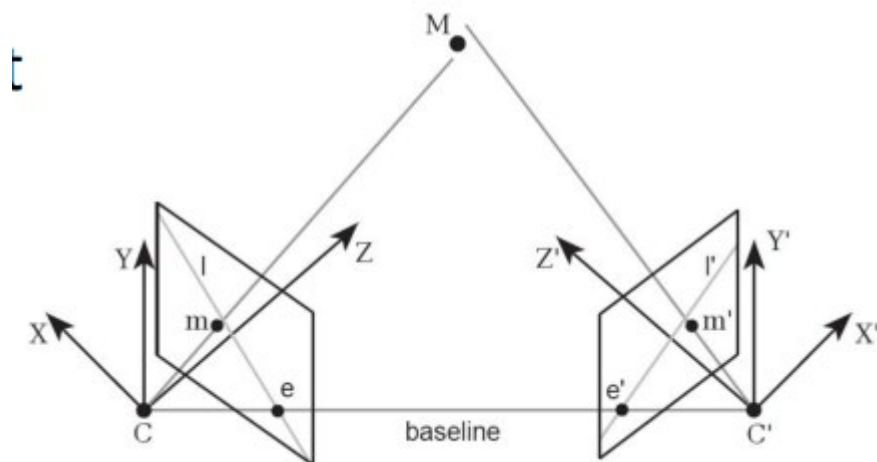
Calculate position and orientation of cameras to each other and points in space

Segmentation & feature detection

Usually done with binary images and contour based

Feature correlation

Projective Reconstruction – acquires 3-DOF marker positions



Rays usually don't perfectly intersect, so you have to estimate

To reduce the error by this, you shoot multiple rays (bundles)

Model fitting

Compute the distance between markers (Euclidean) then select the best hypothesis

Pose estimation

Based on the stored target model, perform a least-squares estimation of rotation and translation

HTC Vive

Inside-Out, 2 line lasers per base station + synch IR LEDs

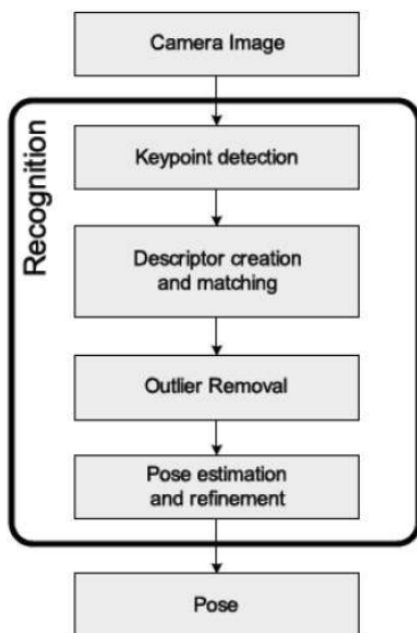
Uses photo diodes on HMD and controllers which take in light and make signals from it. The base station shoots lasers at it, from the time difference the angle to the base can be calculated

Natural Feature tracking

More difficult than marker tracking

Markers are designed for their purpose, the natural environment isn't

Slower than marker tracking



No markers (beautiful) -> more acceptance commercially

Very precise

Suffers from same stuff that markers suffer from + performance heavier

Depth Cameras

Types

Structured Light

Like Kinect

3D Sensor projects infrared pattern onto scene, detects pattern shift in camera image, uses machine learning to estimate body pose

Time of Flight (ToF)

Compute the time of flight of IR pulse for each pixel

Output depth image in addition to RGB

Light Field Camera

Camera that uses a microlens array to capture 4D light field information about scene

Allows reconstruction of the whole light field situation

Motion Capture

Methods

Exoskeleton (mechanical)

Wireless Magnetic Sensors

Wireless Inertial Sensors

Marker based (Optical)

Pure vision based (no markers)

Inertial or Gravimetric

Accelerometers

Linear acceleration with mass on spring in tube, using Piezoelectric (crystals that generate voltage in response to applied mechanical stress)

Rate sensors (Gyroscope)

Use Dreherhaltungssatz to measure angles

Angular velocity

Excentric mass

Gravity sensors

Orientation in gravity field

Vibrating Gyroscope

When rotated, a vibrating element is subjected to Coriolis's effect, which causes secondary vibration orthogonal to the original vibration direction. Sensing this, the rate of the turn can be detected

Ring Laser Gyroscope

Fiber Optic Gyroscope

Inertial Navigation

Relative navigation from known starting point

Inertial Tracking

+very fast, high accuracy in short time, robust

-bias integration results in drift, wireless version require bigger sender

Tracking based on ToF

Acoustic Trackers (Ultrasound)

+small and light weight targets, inexpensive

-line of sight issues, accuracy depends on constancy of velocity of sound, speed of ultrasonic waves varies, ultrasonic noise

GPS/DGPS

Uses 24 satellites in orbit, 4 of them seen simultaneously from any point of earth + 6 monitoring stations 4 ground antennas + 1 master control

Standard service SPS

Precise US government service PPS

Each satellite has atomic clock

GPS receivers need to receive signals from at least 3 different satellites to determine position but receives clock as unknown time bias which results in drift, therefore uses 4 satellites to calculate position and time bias, in practise this usually will take 5 actually (queue song)

+uniform global accuracy, cheap and easy

-direct line of sight needed (because it's a radio signal) -> no GPS indoors, problems in narrow street passages or woods, resolution and precision low

DGPS – differential GPS uses ground stations as reference for higher resolution and thereby accuracy

Radio Frequency – WLAN, RFID, UWB

Phase Difference Time of Flight

Target sends signal, receiver has its own signal with the same frequency, the receiver compares the phase shift of the signals, the modification indicates the relative movement (think Dopplereffect)

+ high data rates possible, as phase shift can be measured continuously

-limited to environmental conditions like temperature and pressure, reflections in the environment are problematic, accuracy

Locomotion Devices

Locomotion = Active Movement

Hence, such devices are: Treadmills, stationary bicycles, walking/flying simulations

Output

Human Sensory Perception

Vision .7

Hearing .2

Smelling .05

Tasting .04

Touch/haptic .01

Different types of displays

Visual Displays, Auditory Displays, Taste Displays, Haptic & Tactile Displays

Immersion

Suspense of disbelief

A simulation is convincing when the artificial reality is simulating human senses

Classification by VR Immersion

Desktop VR

= Window on World system = conventional screen +3D graphics

Fisthtank VR

Tracking

Stereo shutter glasses

Semi-immersive

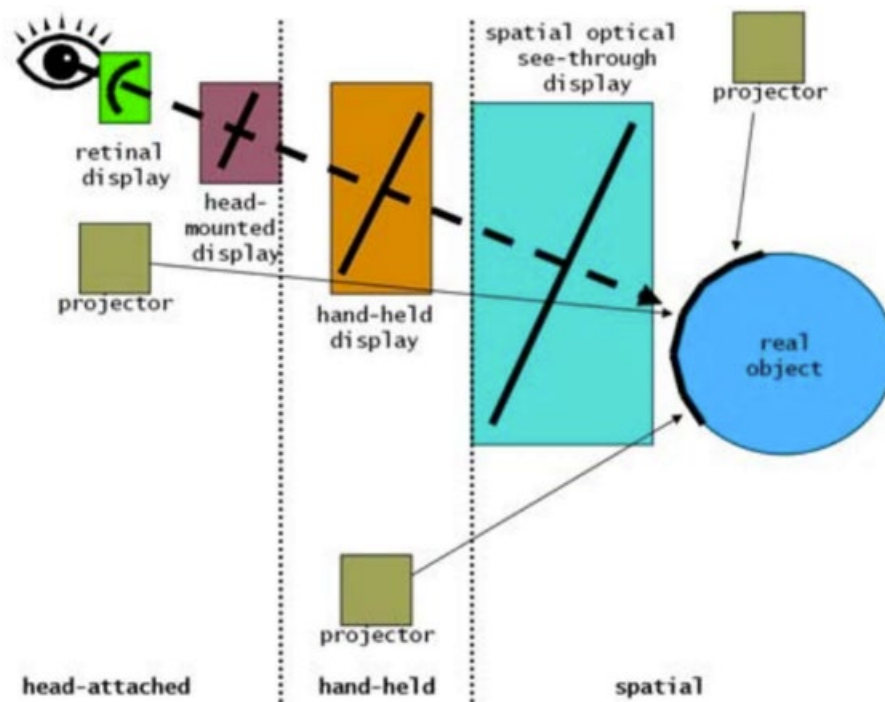
CAVE, Workbench, large stereo screens

Full Immersion

HMD

Audio, haptic interface

Taxonomy of AR displays



Visual Displays

Field of View FOV – for Humans $\sim 200^\circ$

Spatial Res dpi

Screen Geometry (rect, hemispherical...)

Light transfer Mechanism – front/back projection, direct laser to retina...

Refresh rate (hz)

Ergonomics

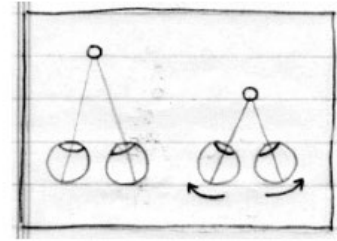
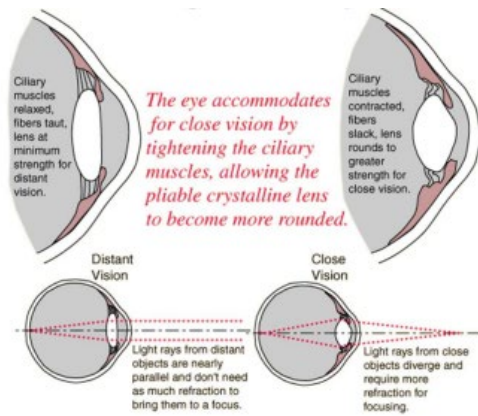
Depth Cues – How to see in 3D

Monocular static cues

Relative size, height relative to horizon, linear perspective, lighting & aerial perspective (blueish and hazy when further away), texture gradient (more detail = closer), Occlusion, shadows, shading

Oculomotor Cues

Derived from muscular tension, accommodation change of eye focal length, convergence eyes looking inwards



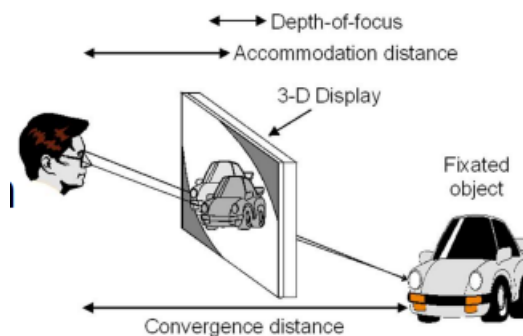
Motion parallax

Closer objects move faster

Binocular Disparity/Parallax

Shift in left/right images

Problem with stereo images – display cue mismatch



Stereo Principles

Active Stereo

Active switching eg shutter glasses

+ least expensive in terms of additional hardware over other output devices, allows usage of many input devices, good resolution, user can take advantage of keyboard and mouse

-not very immersive, user can't move around freely(no motion parallax), does not take advantage of peripheral vision, ghosting, occlusion can avoid IR contact between emitter and glasses

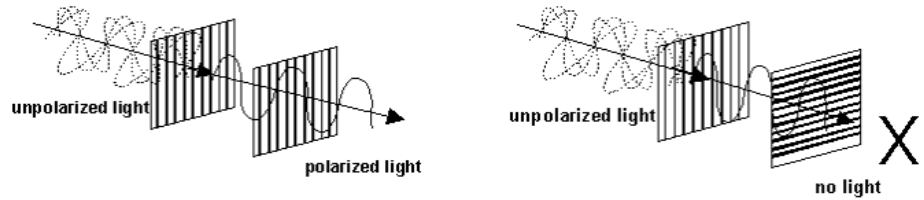
Passive stereo or Polarisation

Eg anaglyph stereo (red/blue), polarized light

Filters create different images for left and right eye

Light is an electro magnetic wave with amplitude (intensity), wave length (colour) and phase

One eye could be polarised with a vertical, the other with a horizontal filter



With linear polarisation, you can't tilt your head, because that would fuck up the alignment of the filter to the light waves. Circular polarisation is better in that regard, you can tilt your head, however, you get more ghosting as is

Head mounted displays HMDs

+immersive experience, easy setup(unless disabled), no restriction of moving around(cable), good high quality HMD affordable, good stereo quality

-limited FOV and res, heavy for neck, no extended use because of cybersickness, AR HMDs have even smaller FOV

Device has one or two screens (LCD, OLED) plus special optics in front of the user's eyes

Provides a stereoscopic view that moves relative to the user

Two versions – seethrough or user cannot naturally see the real world

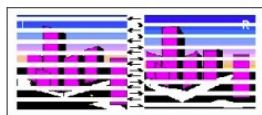
With seethrough there is again two versions – optical seethrough or video seethrough

These are usually used in AR contexts

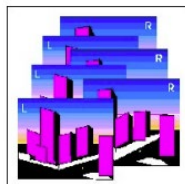
For the stereo image there are three types



Line Interleaved Stereo



Field Sequential



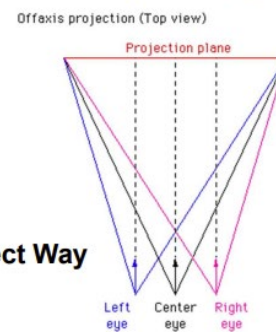
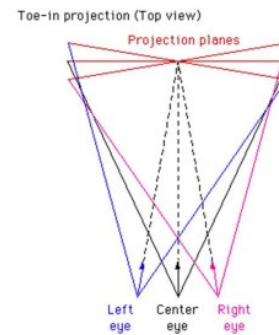
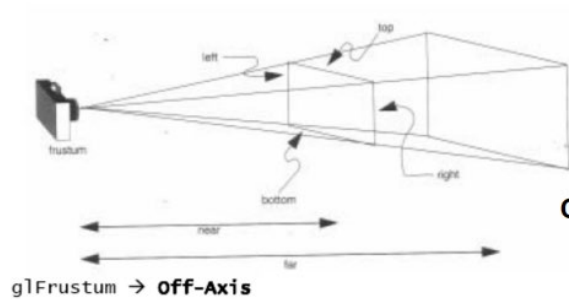
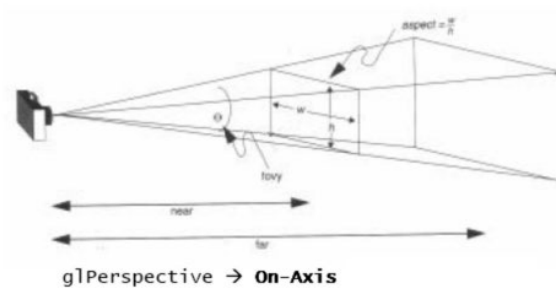
Side-by-Side Stereo



Sometimes: Top/Bottom

Likewise, there are different projection angles

On/Off-Axis Projection



Correct Way

OLEDs

Organic light-emitting diodes

No backlight necessary -> very thin

Deep blacks, high contrast, low power consumption -> all good for HMDs

Oculus Rift

Stereoscopic, FOV 100°, 2 AMOLED fullhd displays @90Hz

Inertial tracker built in

Also IR optical tracking

HTC Vive Pro

FOV 110°

Inertial tracking built in

2 cameras built in (see-through & depth)

Res 1440x1600 per eye @ 90Hz

Vive Cosmos

Integrated SLAM tracking, lighthouse extension

1440x1700

Valve Index

130° FOV

LCD displays with high pixel density

Lighthouse 2.0 tracking

Hololens 2

Small screen 52° FOV

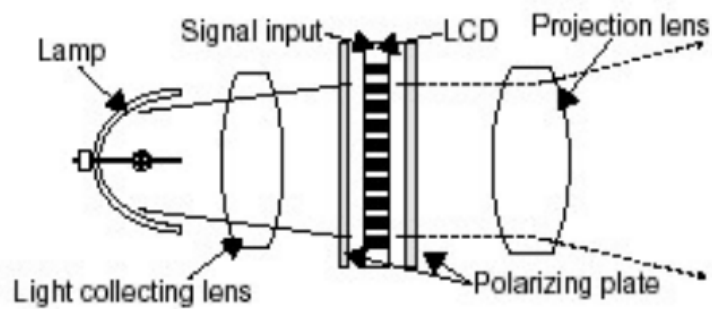
Res per eye 2k

SLAM using 4 cams

Projection Tech and Basic Displays

LCD

Liquid crystal Display



-individual grayscale LCD for each colour, poor contrast, pixel dimension <50micrometers

+ low cost

DLP

Digital Light Processing

Fast switching of micro mirrors for brightness and colour

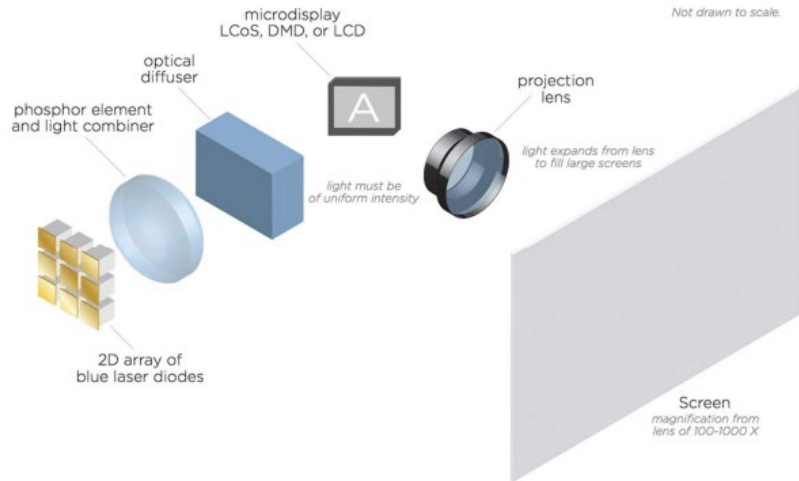
Uses information of several frames for artifact compensation -> leads to delay

High refresh rate > 120Hz

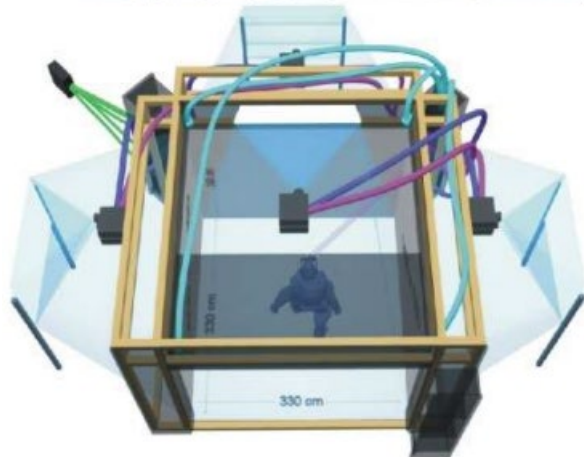
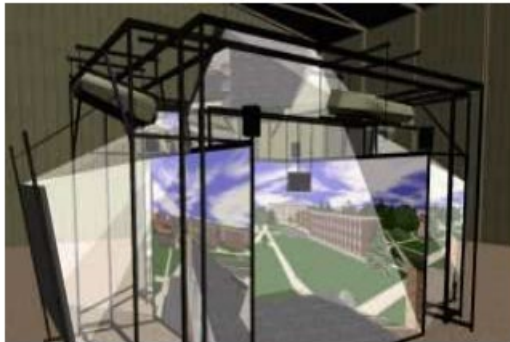
Low cost

Laser Light Source

One or multiple (RGB) laser light sources
Longer life of light source



CAVE



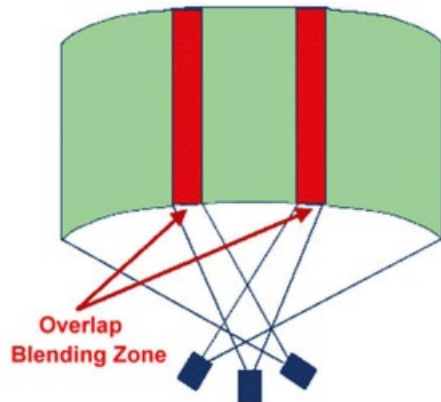
+ high res and large FOV, peripheral vision, only lightweight shutter glasses needed for stereo, freedom to move, space to place props, real and virtual objects can be mixed, group of people can inhabit the space

-expensive, large amount of physical space, projector calibration must be maintained, only 1-2 users can be head tracked, no direct interaction possible, physical objects can get in the way of graphical ones

Curved Displays

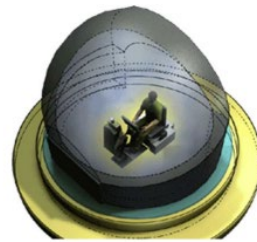
Cylindrical or hemispherical screens that require distortion correction

When multiple used, requires to correct blending zone

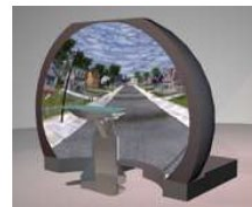


Vision Dome

Vision Dome



Spherical Display

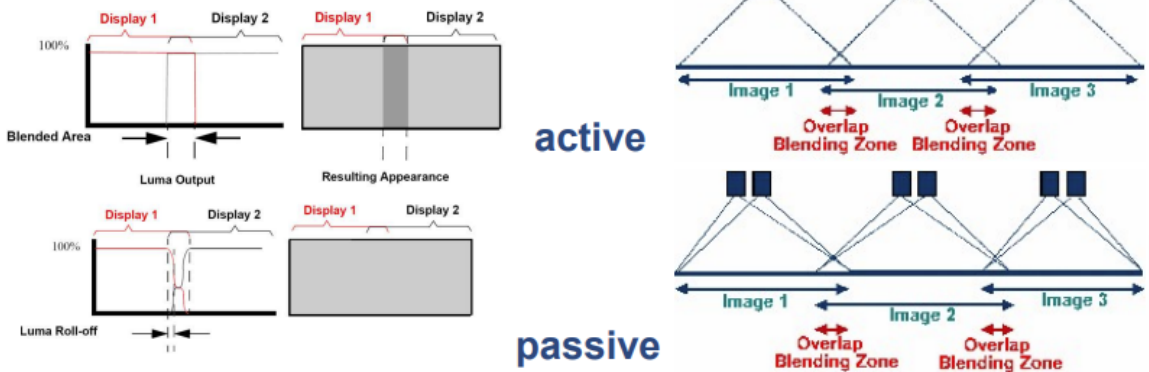


VisionStation

Multi-Projector walls

Active or passive stereo, multi-projector setup, overlap or edge blending through driver support or on GPU

– Warp & Blend can be done on GPU



Multi Screen Displays

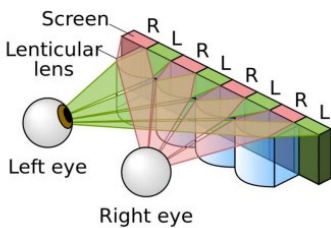
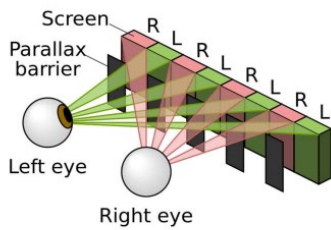
Needs to be synchronised – through multi head graphics or multiple workstations
genlock/Framelock

Genlock – exact synchronisation of vertical synch (electron beam of CRT) – refreshes each pixel synchronously

Framelock – Synchronising frame buffer swap – begins redrawing at the same time

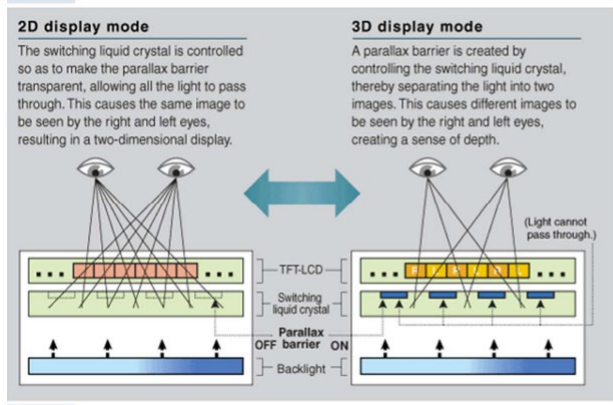
Auto-stereoscopic display tech

Stereo without glasses, basically just blocks the left image from the right eye with lenses and stuff and vice versa



Types

Parallax barrier



SHARP 3D display + stereo cameras



Nintendo 3DS



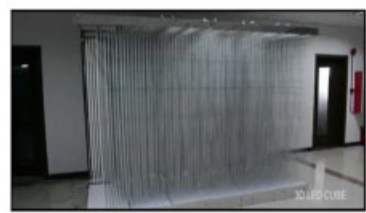
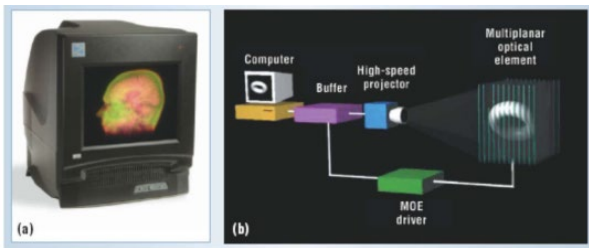
Lenticular

No glasses, various components

Volumetric

Swept volume – plane or helical surface

Stack of planes



Holographic

Auditory output

Localisation creates spatialised sound (3D Sound sampling)

Sonification communicates information via sound (non verbal through tempo changes and the like)

Ambient effects – realism

Sensory substitution

Annotation and help (without speech)

Different types of setups, can be super powerful (adaptive game music), tells user where to look

Olfactory displays – scent

Sense of smell or olfaction displays

Haptics

Touch/tactile feedback relies on sensors in and close to skin

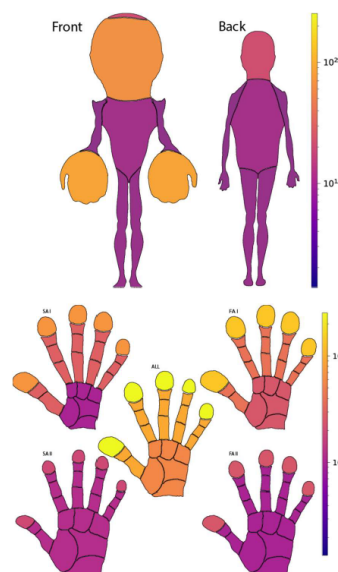
Mostly sensors on hands, conveys information about contact surface:

Geometry, Roughness, Slippage, Temperature

Easier to implement than force feedback

Tactile Sensitivity of Human Body

- ~230,000 tactile afferent fibers across whole body
- 15% innervate palmar skin of both hands
- 19% in region surrounding face and lips
- ~60% of all tactile fibers are slowly-adapting the rest are fast adapting



Force feedback

Conveys information such as contact surface but also object weight and inertia

Actively resists the user's motions

More difficult to implement

Interface Challenge

Naturalism

Make VE & Interaction work exactly like real world

Magic

Give user new abilities like perceptual, physical or cognitive ones

Will the target users be able to learn to navigate and interact with the environment in an effective manner?

Will the cognitive overhead required to use the interface distract users from the intended tasks and goals?

A VE is effective when

Users' goals are realised easily

User tasks done better, easier and faster

Users are not frustrated

Users are not uncomfortable

There is some measurable gain in targeted real world performance

3D Graphics

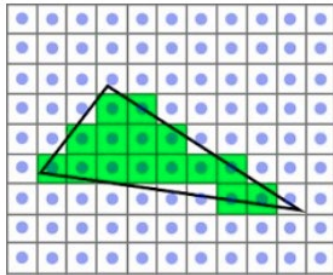
Rendering Algorithms

Rasterisation

Real time rendering VR/AR

Hardware acceleration on GPU

Rasterises triangles into pixels



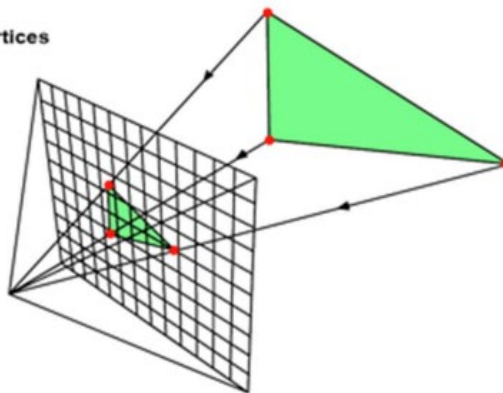
First you project each triangle to the screen

Draw each pixel which belongs to a triangle (rasterise triangle)

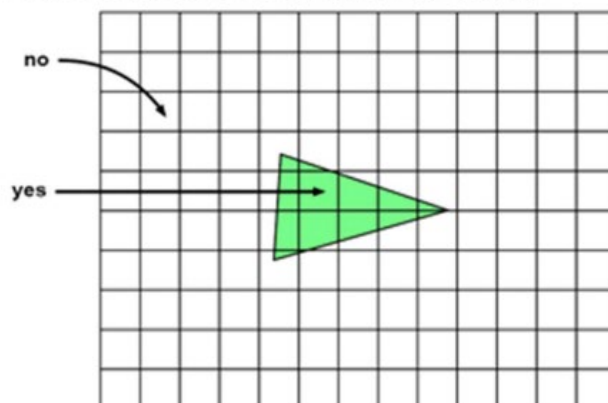
Solve visibility by Z-buffer algorithm

Compute per pixel colour

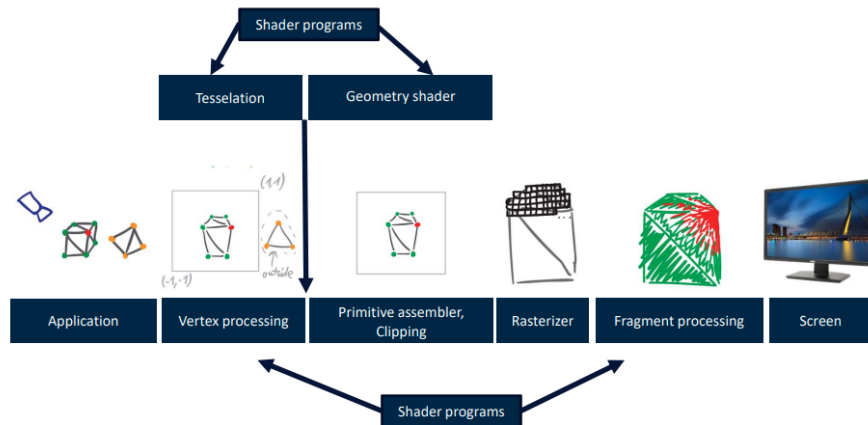
1) Project vertices



2) Loop over pixels. Does the pixel lie in the triangle?



On the GPU the pipeline looks like this:

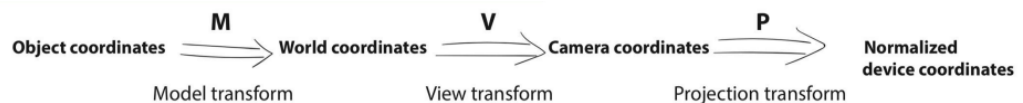


It consists of programmable parts (shaders), fixed parts and configurable parts (backface/frontface culling, z-buffer, alpha blending)

In the first step, the application contains the model-view-projection transforms comprising the 3D scene

Secondly, applying the MVP matrix, execute the vertex shaders (Vertex processing and primitive assembler clipping from the picture) – This manipulates colour and position per vertex on the GPU(shader) eg for height maps or colour animations

MVP = Model view projection



The clipping from the third box in the image gets rid of everything outside of the current view – this is not programmable

Then we go on to the last three steps which are rasterising, meaning scanning through object surfaces and generating fragments (pixels with depth information) this is important for the z-buffer that defines what is in front of what

Then we go to the fragment processing where linear interpolation (like colour mixing and UV coordinates) of vertex data to fragments happens – this is programmable and adds camera noise

Finally, everything is output on the screen

Shaders in depth

There are Vertex and Fragment shaders, they are parallelised

Z-buffer in depth

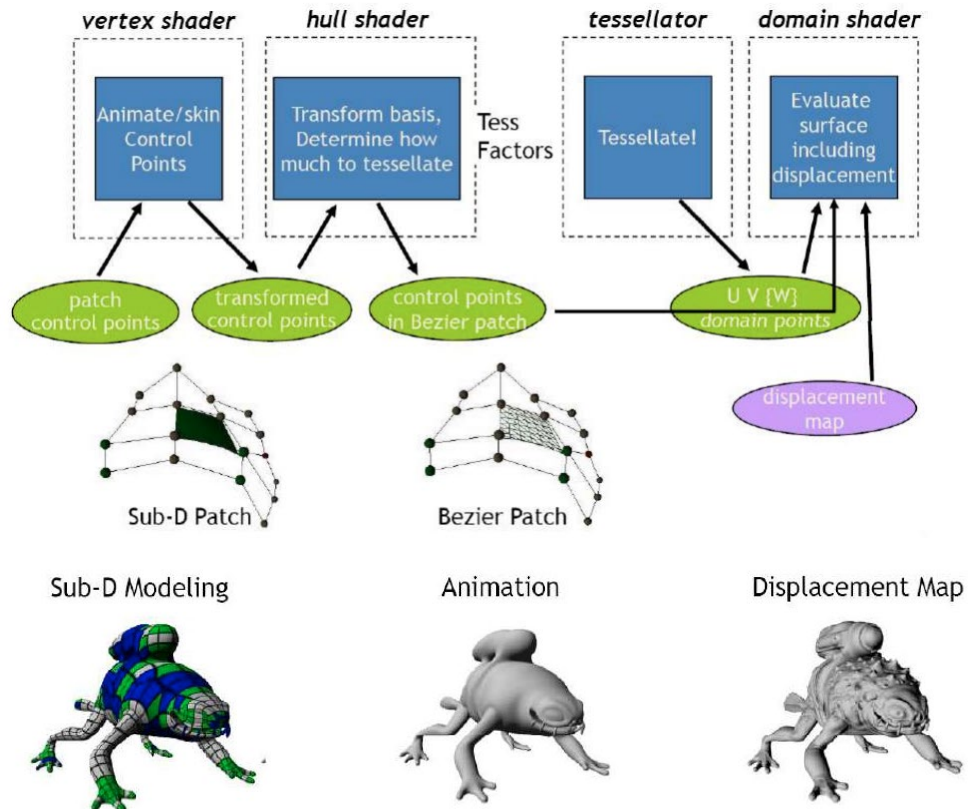
Like the colour buffer, the pixels also have a depth buffer, this is the Z-buffer that determines which pixel is drawn (others behind can be disregarded)

Culling is used in addition to the z-buffer, it is an optimisation for removing hidden surfaces. It identifies surfaces with normal facing away from the camera

Vertex indices specify whether the triangle is facing the camera or not, which is usually the case when its three vertices are clockwise after the projection, this can be determined using the cross product

Tessellation and geometry (still shader programs)

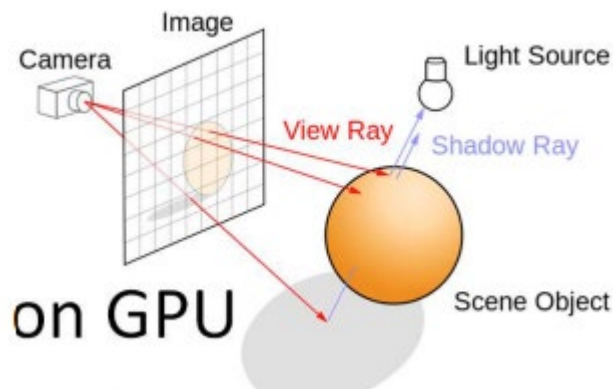
Tessellation is a stage used to subdivide and modify geometry (eg smoothing or displacement)



Geometry shaders process geometric primitives such as triangles, lines and points. They can add, modify or remove primitives, which can be used for sprite expansion, dynamic particle systems and shadow generation

Ray-tracing

High quality rendering, super expensive



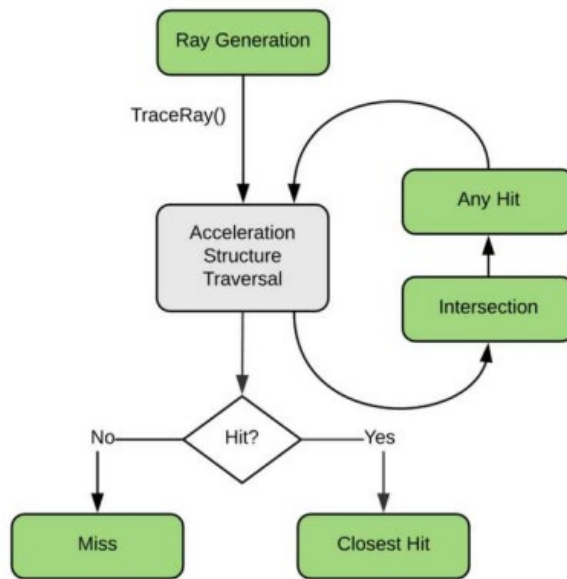


Figure 1. The ray tracing pipeline

Cast a ray from a light source to everywhere, those rays are reflected off the surfaces (adding the colours of the surface hit) and again shooting it everywhere, depending on the angle the reflection differs and what not (using the normals) but essentially, we trace it till it hits a camera, where we see which pixel it correlates with. As this would be super unlikely to hit the tiny camera, we do it in reverse, so we start with the light source from the camera and wait to hit a light.

There are algorithms based on ray-tracing such as whitted ray-tracing for only specular reflections, path tracing which traces all light paths from the camera, photon mapping which goes from a light source and irradiance caching which interpolates between sparse samples.

Raytracing creates noise (because we don't have infinite rays), so it must be denoised with eg AI

Parallel Graphics Hardware

We need this because all the mentioned effects take up immense computing resources

There are different types:

- On-chip/graphics board (standard)

- Multiple boards (former graphics super computer) with multi GPUs

- PC Clusters offline rendering standard network in distributed environments or real-time render with PC clusters with special hardware

Advanced AR Techniques

Coherence

Spatial Coherence

Tracking and registration

Visual Coherence

Photometric registration

Common Illumination (Light capturing/estimation)

Camera simulation

Light interaction

Stylised AR

Occlusion

Steps for visual Coherence

Acquisition

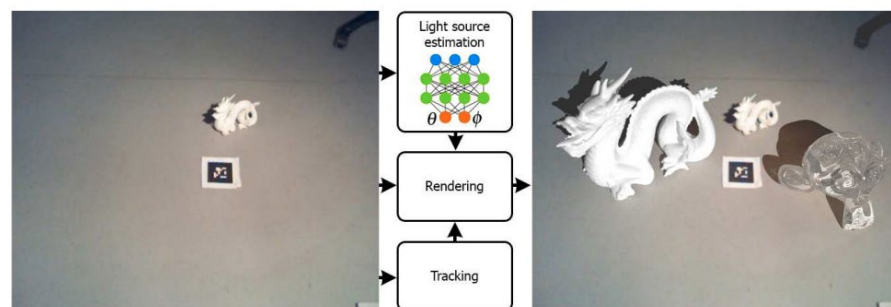
Illumination capturing with fish eye camera(light probe camera) or mirror sphere or with the main camera during motion

Illumination estimation from main camera image

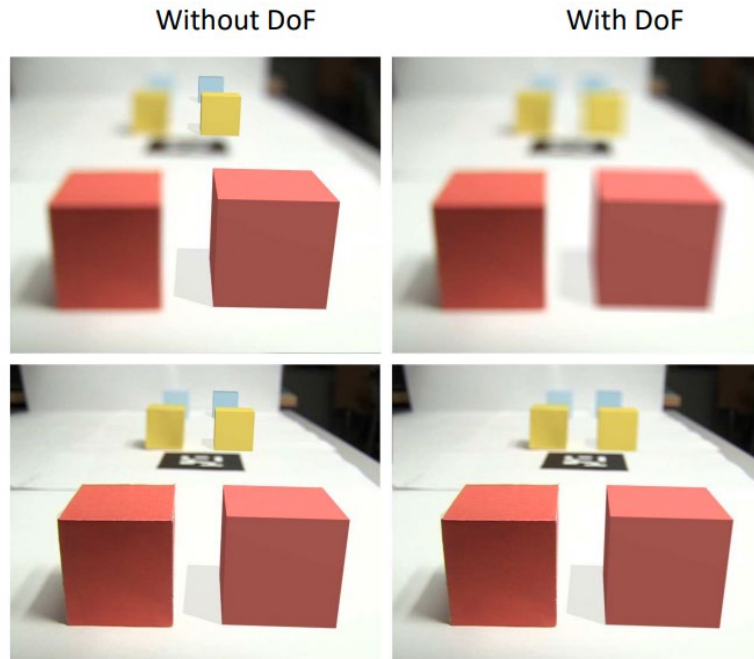
Also accumulation of images with HDR

Registration

Light source estimation

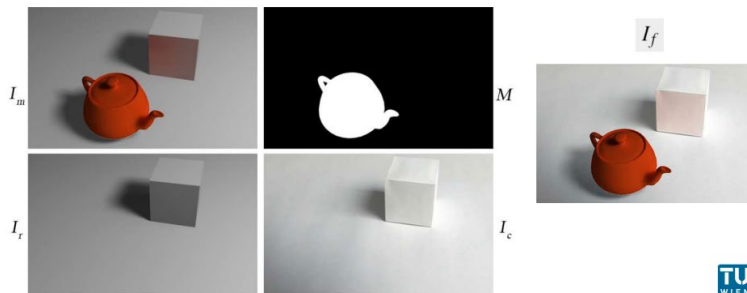


Also physically based depth of field



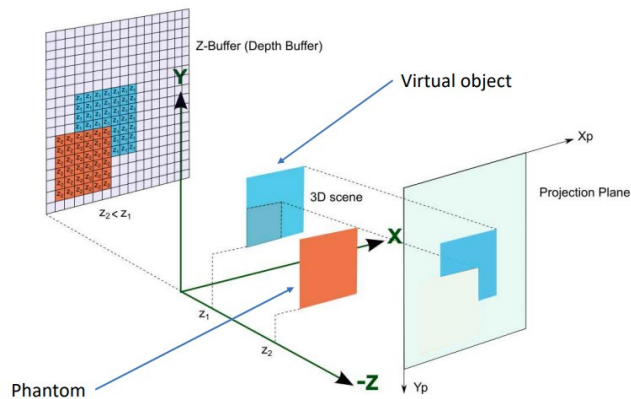
Composition

$$I_f = M \odot I_m + (1 - M) \odot (I_c + I_m - I_r)$$



Occlusion

Problem here is that video does not contain depth information, hence the solution for AR is drawing the shape of real objects into the Z-buffer to prevent virtual objects which are supposed to be behind the real object from overdrawing



This is called phantom rendering – Phantom objects represent the invisible geometry of real objects.

Algorithm:

Draw video image to colour buffer

Disable writing to colour buffer

Render phantoms of the real scene to Z-buffer

Enabling writing to colour buffer

Draw virtual objects

There are variations of this technique:







Model-based – target needs to be tracked and of known shape

Probabilistic – use alpha blending in image areas with uncertain occlusion (eg around hand)

Model -free – Using depth sensors (eg Kinect)

3D Interaction Techniques

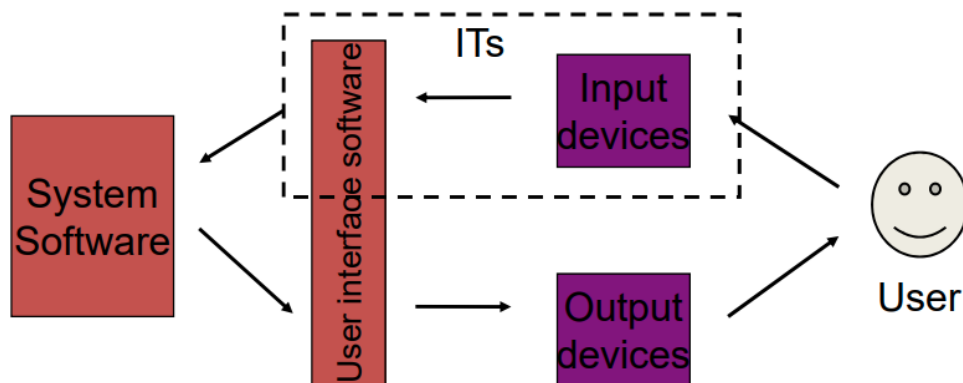
Gestures

Gesture	Posture		Tracked DOFs
point			position (3 DOF) + direction (2 DOF)
pick			pose (6 DOF) + distance between two fingertips
release			pose (6 DOF) + distance between two fingertips

3D interaction techniques are methods used to accomplish a given task via the interface

Hardware components are input&output devices

Software components are control display mappings translating information from input devices to system actions -> display to user



Again you have the Interface Challenge

Naturalism – make VE & Interaction work exactly like real world

Magic – give user new abilities

Goals of interaction design

Performance

Efficiency

Accuracy

Productivity

Usability

Ease of use

Ease of learning

User comfort

Usefulness

Users focus on tasks

Interaction helps users meet system goals

Difficulties

Spatial input, Lack of constraints, lack of standards, lack of tools, lack of precision, layout more complex, fatigue

Universal Interaction Tasks

Selection – picking objects from a set

Goals are indication of action on object, make object active, travel to object location, set up manipulation

Manipulation – modifying objects

Goals of Manipulation are object placement – Design, Layout, Grouping as well as Tool usage

Navigation – travel, wayfinding

System control

Symbolic input

Isomorphic vs nonisomorphic

Isomorphic – strict, geometrical 1:1 correspondence between physical <-> virtual world, most natural, imitates physical reality and its limitations

Nonisomorphic – magic virtual tools that extend working volume or arm length, depends on application, majority of manipulation techn. nonisomorphic.

Basic Techniques and Extensions

Basic techniques are pointing or grabbing and also speech.

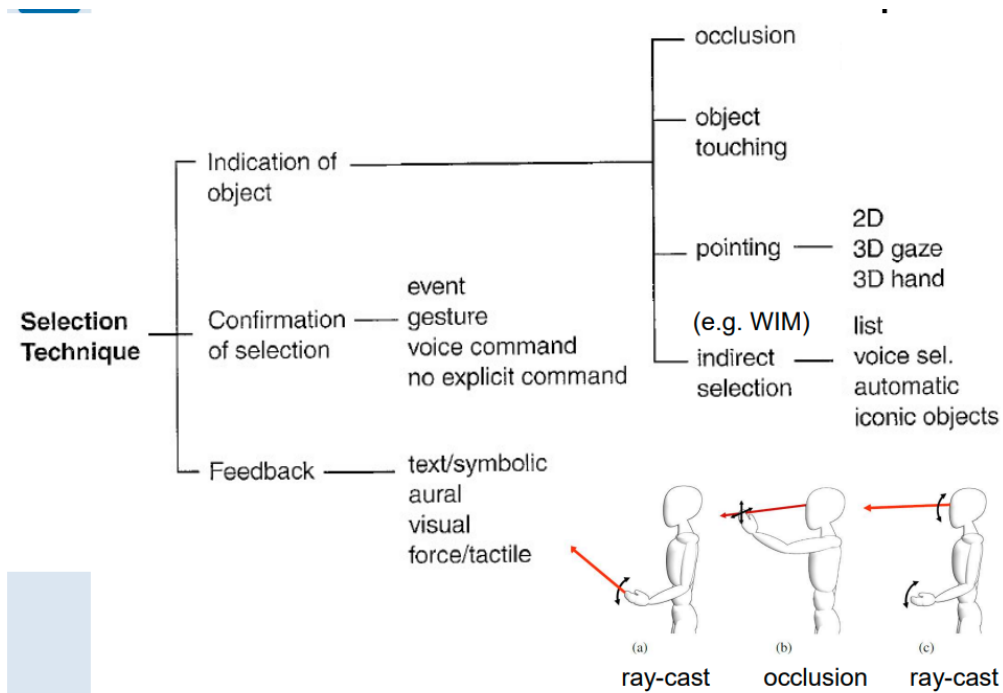
Extensions can be:

Go-Go techniques – non linear mapping between physical and virtual hand position

Fishing-reel technique (additional device . distance)

World in miniature where you select icon-like objects

Classifications of selection techniques



Comparison of Techniques

Ray-casting and image-plane usually more effective than Go-Go except for very small objects where the ray barely hits it ever

Ray-casting and image-plane techniques result in the same performance

Manipulation metaphors

Simple virtual hand

Natural, easy placement

Limited reach, fatiguing

1:1 position mapping

Ray Casting

Little effort required

Exact positioning and orienting very difficult (lever arm effect)

Indirect depth control (eg mouse wheel)

Infinite reach, no tiring

Not natural, separates DOF

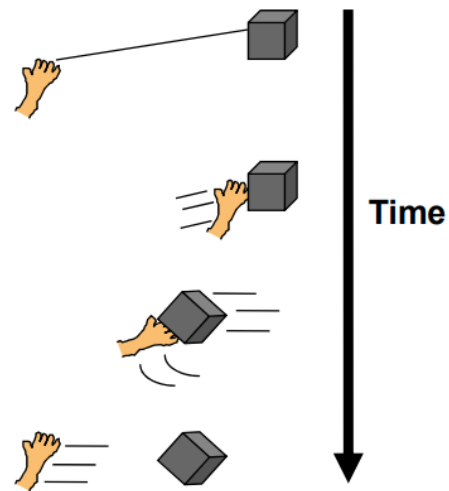
HOMER technique

Hand-Centered Object Manipulation Extending Ray-Casting

Select: ray-casting

Virtual hand moves to object

Manipulate: hand



Scaled world grab technique

At selection scale user up or down so that virtual hand is actually touching selected object

Voodoo dolls

Indirect manipulation

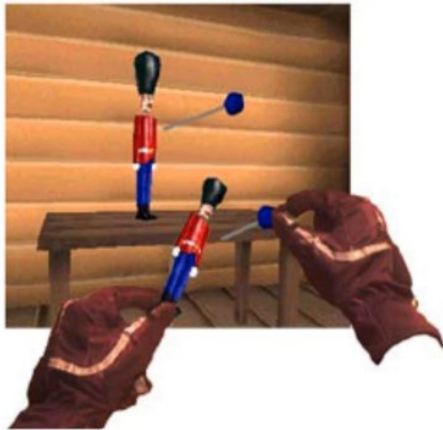
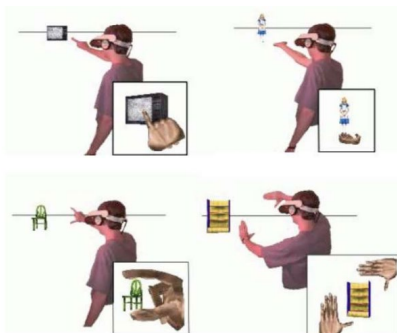
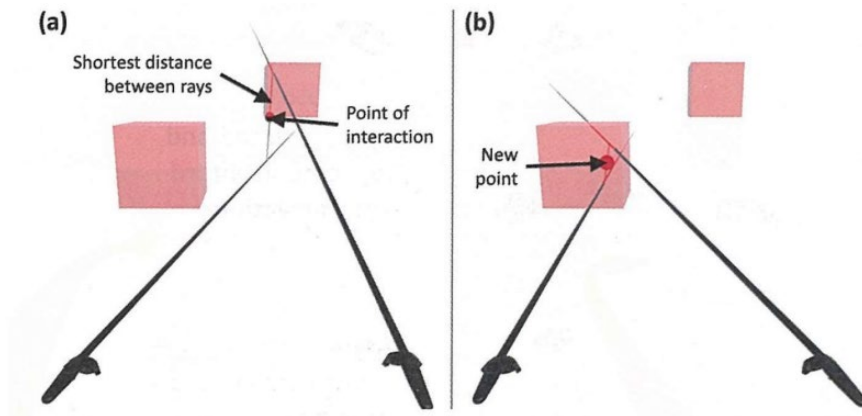


Image plane interaction

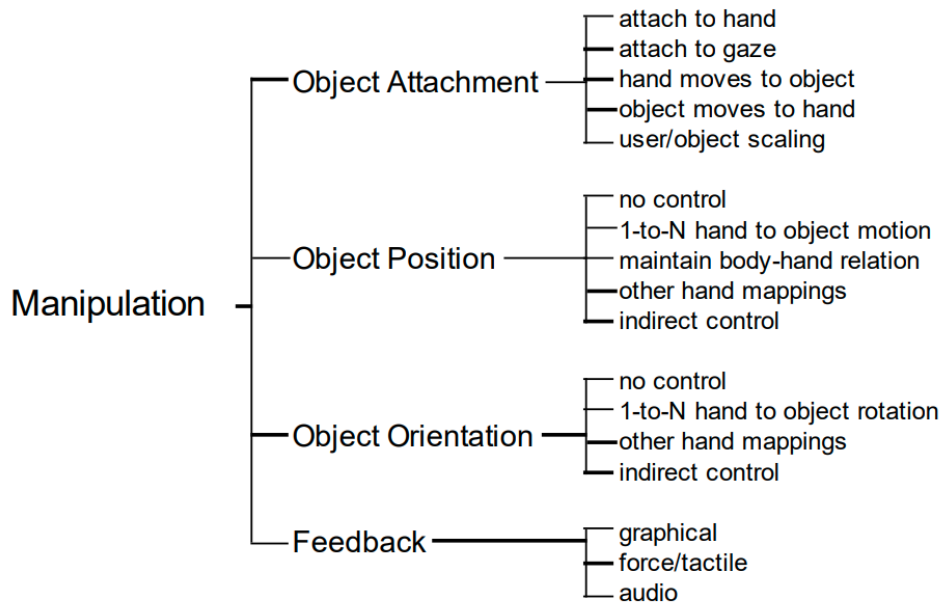


Symmetric Bimanual Technique



Two raycasts

Classification of manipulation techniques



Comparison Positioning task

Raycasting works if object is repositioned at a constant distance

Scaling techniques (HOMER, scaled world grab) difficult in outward positioning of objects eg picking an object located within reach and move it further away

If outward positioning is not needed then scaling techniques might be good

Design guidelines for manipulation

There is no single best manipulation technique

Naturalism is not always the goal

Consider grasp-sensitive object selection

Use pointing techniques for selection and grasping techniques for manipulation

Use existing techniques

Consider the trade-off between technique and environment – design your environment for optimal manipulation if possible

Navigation

Combination of Wayfinding (cognitive component of navigation) and Travel (motor component of navigation)

Tasks of Travel

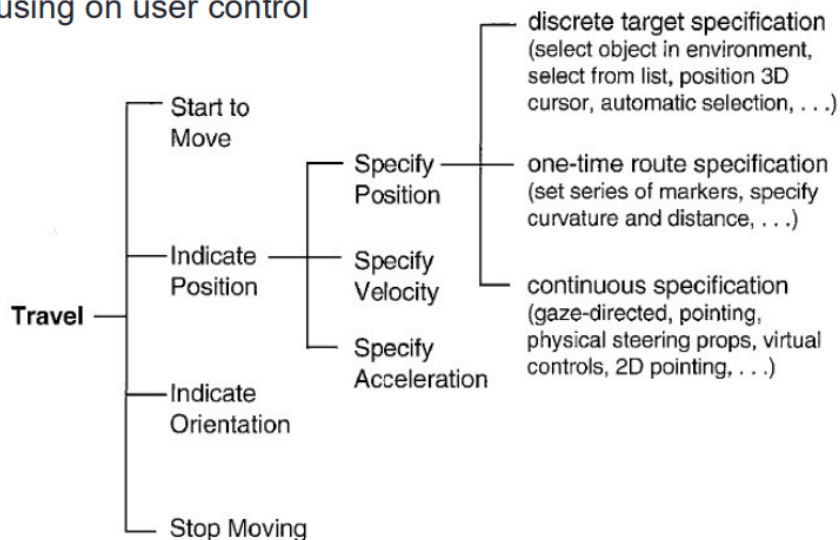
Exploration – travel without target, build knowledge on environment

Search – move to target location, naïve when target position not known, primed when position known, build layout knowledge

Manoeuvring – short precise movements, travel to position the viewpoint for a task

Movement Process

Focusing on user control



Traveling Metaphors

Steering – Continuous specification of direction via gaze or pointing/device

Target-based – point at object (teleport), choose from list or enter coordinates

Route planning – one time specification of path, placing markers in world, moving icon on map

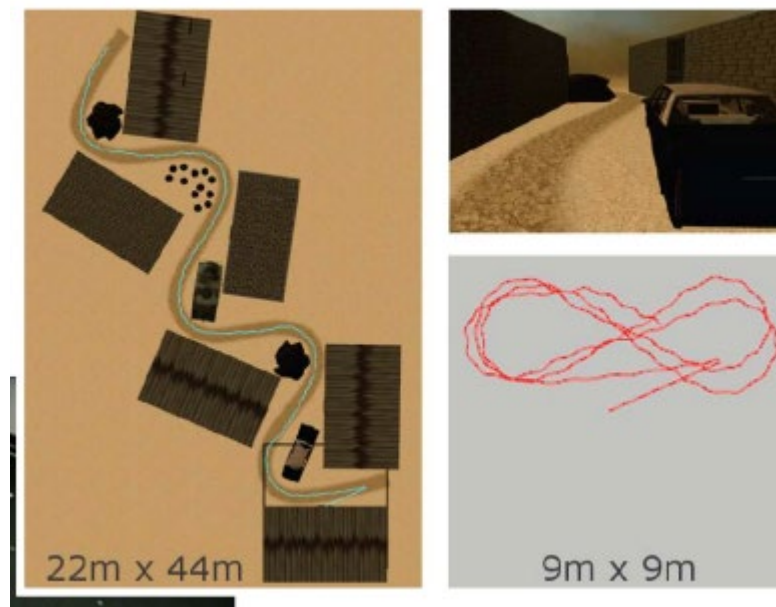
Manipulation – camera in hand, fixed object manipulation like film camera

Natural Travel Metaphors

Walking techniques, treadmills, bicycles, magic carpets, river raft ride, simulation of flying

Real walking – enhances sense of presence, enhances perception of size and distance, focuses attention, improves task performance but limits size of virtual environment to size of tracking space

Redirected walking through waypoints, distraction ...



Can be done through change blindness where changes are applied while user is distracted (cyclic paths possible)

Same goes for flexible spaces with overlapping rooms, real walking, focus on virtual constant and bigger distance between the rooms meaning more overlap

Spatial Perception in VR

Self-overlapping rooms

You can layout rooms in S-shapes to create bigger walkable spaces essentially

Evaluation of methods

Teleportation leads to disorientation

Environment complexity affects information gathering

Travel IT and user's strategies affect spatial orientation

There is no optimal travel technique

A natural technique will not always be better

Desktop 3D, workbench and Cave applications can't all use the same HMDs and techniques

Design guide for navigation

Match the travel technique to the application

Use an appropriate combination of travel technique, display device and input device

The most common travel should require minimum effort from the user

Use physical locomotion technique if user exertion or naturalism is required

Use target based techniques for goal oriented travel and steering techniques for exploration and search

Choose travel techniques that can be easily integrated with the other interaction techniques in the application

Maps

Rules for good maps:

Provide you are here marker, provide grid, choose either north up or forward up map, try mixing local and global maps

System control techniques

Launching application, changing system settings, opening file

Make commands visible to user and support easy selection

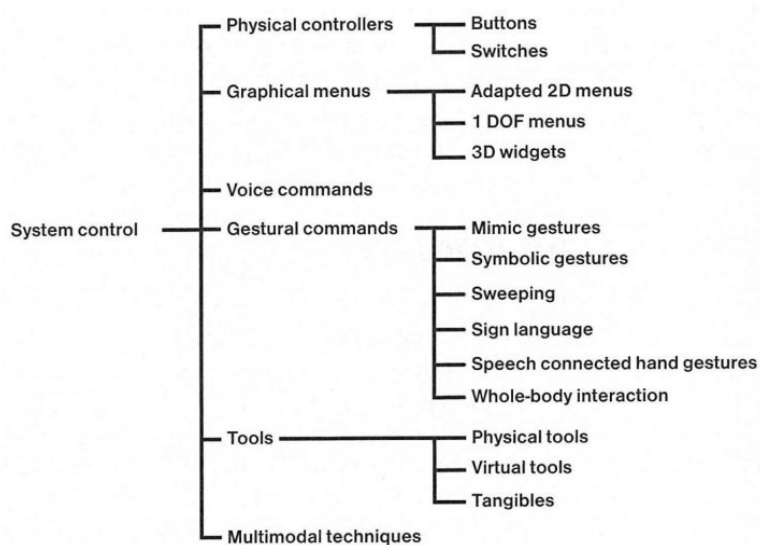
Types

Menu systems

Voice commands

Gestures/postures

Implicit control (pick up new tool to switch modes)



Floating Menus

In 3D they can occlude environment

using 3D selection for a 1D task

can be difficult to find

better than heads-up-display

better if menu follows user

Tulip menu

Attaches to fingers

Pen & Tablet interaction

Tablet = real object that can be put away and allows for handwritten input, can be used as clipboard, constrained surface for input, usability good for people that are used to 2D input

Combines 3D and 2D interaction

Use any type of 2D interface not menus

Pen = direct manipulation tool

Voice input

Implementation – Wide range of speech recognition engines available

Factors to consider – recognition rate, background noise, speaker dependent/independent

Design issues – voice interface invisible to user, no UI affordances, overview of functionality

Limited commands using speech recognition

Complex application

Philosophie of interaction design

Artistic approach

Intuition about users, task

Heuristics, metaphors

Aesthetics

Adaptation

Scientific approach

Formal analysis
 Formal evaluation
 Performance requirements

Comparison interaction techniques VR AR

	Virtual Reality / 3DUI	Augmented Reality
Selection	Raycasting, virtual hand, world scaling	same
Manipulation	Everything can be manipulated.	Distinction between <i>real / virtual</i> objects
Navigation	Viewpoint can be controlled freely.	Only <i>passive</i> hints
System control, Symbolic input	Menus, voice, gestures	same

Interaction techniques for smartphones

Relative to target 2D moving

Relative to world AR 3d

Conclusion

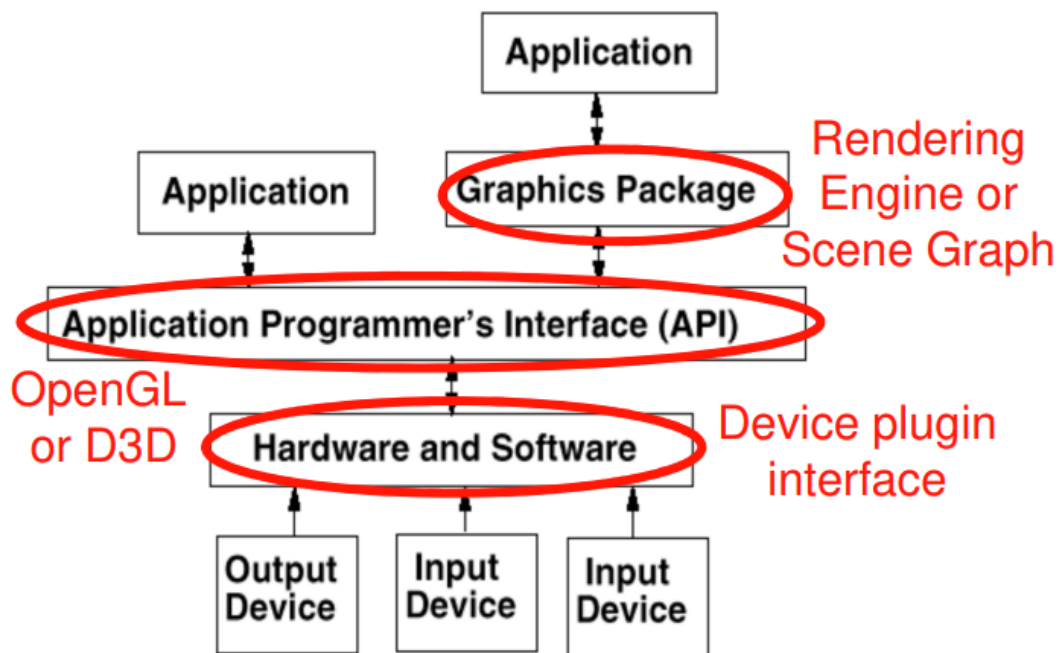
User interface is key for good VR experience – Need 3D user interface techniques

Design for – selection, Manipulation, Navigation, system control

Follow good design guidelines – cannot just implement 2D techniques in VR

High Level Graphics programming & VR System Architecture

Application Programmer's View



High-Level Graphics APIs

Rendering Engine (Unity3D, Unreal Engine, CryEngine) or Scene Graph (Vulkan, OpenInventor, OpenSceneGraph)

Object oriented

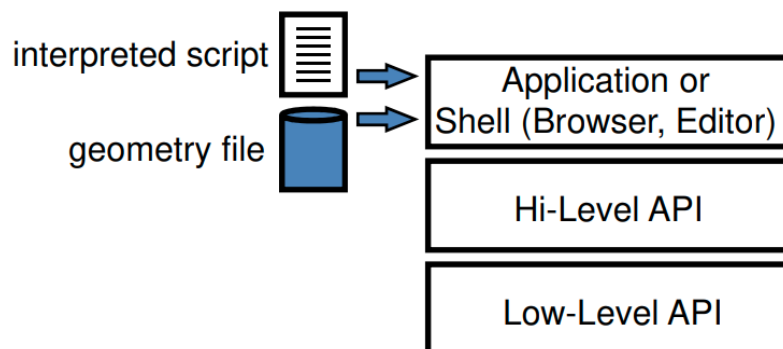
Scene Objects – “Objects, not Drawings”

Interactivity – Event-model for 3D

Why (high level) API?

Rapid prototyping

Rapid application development (RAD)



Scenegraph – Structure

Scenegraph – graph containing 3D scene data

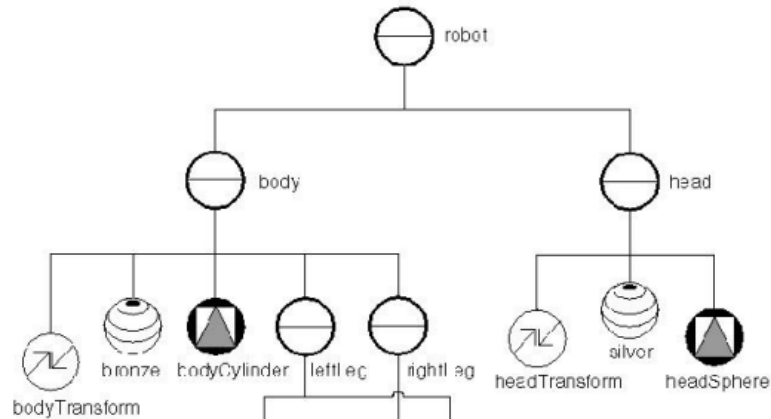
Scenegraph consists of Nodes

Directed graph! (Head/Tail) Directed edges -> hierarchy

Use of the hierarchy

Semantic Hierarchy eg car (parts)

Geometric hierarchy eg puppet/jointed doll



Scene Graph Nodes

Nodes consist of data and methods

Nodes are of a specific type which determines behaviour (reaction to certain events which are in turn generated by the user -> interactivity)

Nodes are instances of a class

Flexibility – Choose node(type), compose scene graph with nodes

Extendible – New nodes can be implemented

Scene Graph Fields

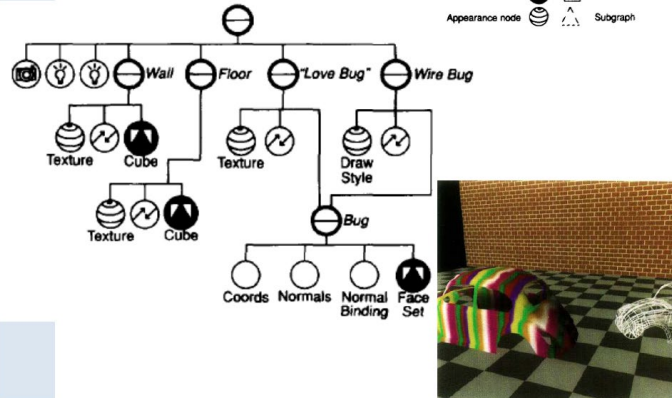
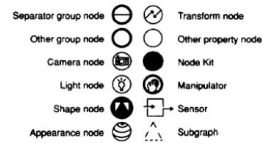
Attributes (member variables) in nodes are called fields

They set/get, detect changes, connect fields across nodes

Fields are objects by themselves – float, string ...



Example



Graph Traversal

Data structure (scene graph) is processed

Foreach node a number of methods can be implemented

Rendering, BoundingBox calculation, Transformation matrix calculation, Handel events (like picking), search nodes, write to file, execute user callback

Dependency Graph

Another structure modelling dependencies

Higher Level Programming

Unity3D

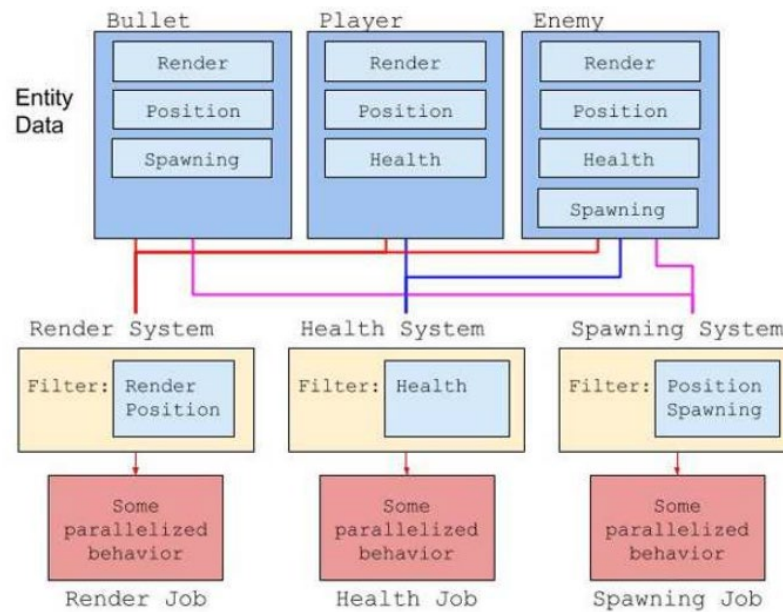
Based on game objects. Game objects are not suitable for scenes with massive amounts of objects, however, entity component system addresses the scalability problem

Entity Component System (ECS)

Entity – a general purpose object eg every game object

Component – a container object, raw data for one aspect of the object and how it interacts with the world

System – Process which acts on entities. Each system runs continuously (as though each system had its own private thread) and performs global actions on every entity that possesses a component of the same aspect as that system – eg physics system or rendering system



Entities are simple compositions, not complex inheritance trees

Simple object lookups

System could – render components, detect collisions, compute physics, manage health

Memory efficient, scalable, extendable

High performance possible

OpenXR

API aimed at developers and a device layer aimed at the VR/AR hardware – abstraction interface with the device

High performing cross platform access to hardware

Distributed Shared Scene Graph

Shared Memory (SM)(ha!) Multiple CPUs access the same memory – very simple, may need mutual exclusion (locks etc)

Distributed shared memory (DSM) – SM on top of standard message passing

Distributed Shared Scene Graphs – DSM semantics added to a scene graph library

Long Distance Distribution Requirements for AR Applications

Main Challenges

Robust application replication

Reliable network communication over long distance

Options what to send

Input data – eg. Tracking data

Output data – application content

Intermediate high level metadata for regenerating correct application state

Evaluations & Human Factors in VR/AR

Why Evaluation?

Identify usability problems and change design

Iterative process design <-> evaluation

Ideally leads to design guidelines

Even better performance models for UIs (eg how fast can a user position a pointer over a target area)

Collecting Data

Preliminaries with participants

Quantitative techniques

- Benchmark tasks Measuring time on task, number of errors, movements in space (tracking data) etc.
 - Quantitative measures such as timing can be valuable even with paper prototype, though not very precise
- Questionnaires (Likert scale)
 - Motivation, usability, presence, cybersickness, self-reported performance, ...
- Biofeedback measures
 - heart rhythm, blood pressure, skin conductivity, respiration rate, ...
- User satisfaction scores

Qualitative Techniques

Verbal protocol taking

- Participants think aloud, talking while performing tasks
 - Can be intrusive, but effective
 - Some participants not good at talking
 - Evaluator/facilitator sits in room with participant to collect data
- Can be used for both timed and un-timed tasks
 - Studies show it can be done with minimal effect on performance time
- Answer questions about what to do with a hint, not a direct answer

Observational techniques

Structured Interviews, Co-discovery (multiple participants with think aloud)

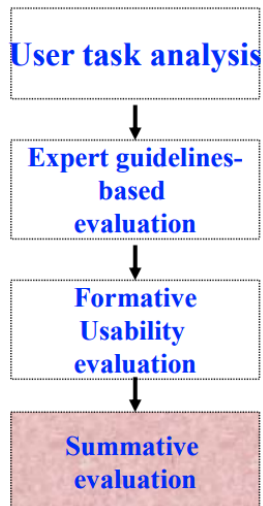
Audio recording, video taping

Usability Engineering

A subclass of human factors research to determine the ease of use of a given product

Usability studies are product-oriented part of dev cycle

No clear standards



Effects of VR on users

There are direct and indirect effects

Direct – involve energy transfer at the tissue level and are potentially hazardous

Mostly on visual system but also auditory, skin, musculoskeletal...

Lasers for eyes

Absence state from flashing lights

Bright lights coupled with loud pulsing sounds migraines

Loud noises

Haptic feedback effects on muscle and skin

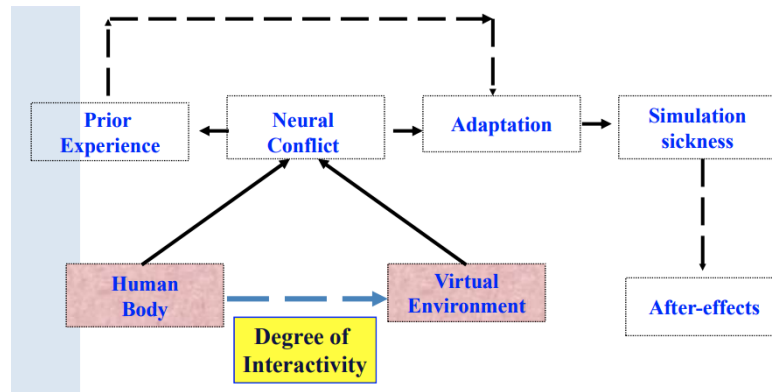
Indirect – are neurological, psychological, sociological or cybersickness

Aftereffects

Such as sensation of self motion, headaches and spinning, diminished hand-eye coordination

Side effects

Simulator sickness (cyber sickness)



Reasons for this can be tracker errors, system lag, HMD image resolution or FOV

But also age, general health, pregnancy, susceptibility to motion sickness

Neural conflict occurs when simulation body and sensorial feedbacks conflict

Type I: two simultaneous conflicting signals (A and B) – example Information from a moving platform does not coincide with the motion of waves seen on an HMD.

Type II: Signal A is present and B is not – example looking at a roller coaster simulation, without a motion platform;

Type III: Signal B is present and signal A is not – flight simulation in fog (instrumented flight). Motion platform moves, but visual feedback is unchanged.

Since more information from the simulation results in more conflict, it is logical that neural conflict induced **cyber sickness grows with the duration of immersion** in the VE.

Ergonomics in VR/AR

Techniques

Anthropometry – Measurement of body dimensions

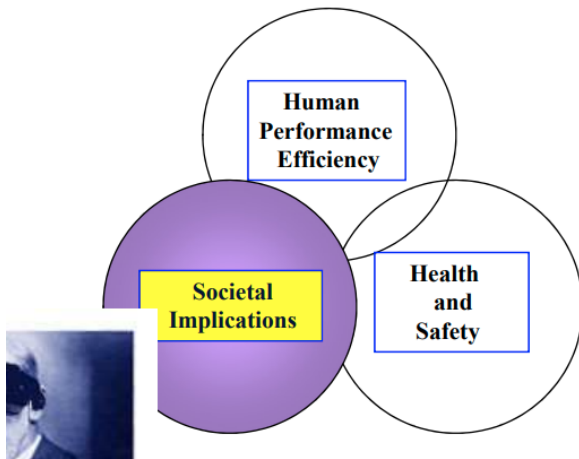
Musculoskeletal – Strain muscular and skeletal systems

Cardiovascular – actions that increase stress on heart

Cognitive – information over/underload

Psychomotor – response to stimuli with physical movement

Human factors VR/AR



Presence Definition

Presence is a state of consciousness where the human actor has a sense of being in the location specified by the displays

A high degree of presence should lead to the participant experiencing objects and processes in the virtual world as (temporarily) more the presenting reality than the real world

Social implications

Health care quality

Synthetic and distance learning using VR

May increase individual isolation