Module IV
VR HARDWARES & SOFTWARES

Human Factor-Introduction

Before proceeding with the technology of VR it will useful to explore aspects of our selves that will be influenced by this technology-this area is known as *human factors* and covers a wide range of complex topics.

Human factors covers much more than the physical aspects addressed in this chapter. It embraces issues such as:

•Human learning in VEs.

•Spatial cognition, spatial memory, spatial ability in VEs.

•Individual differences in VEs

•Cognitive models for VEs

•Multi-user VE design methods.

•Social interaction within VEs.

These topics, unfortunately, are beyond the scope of this text as researchers around the world are still addressing them, so I will just look at our senses.

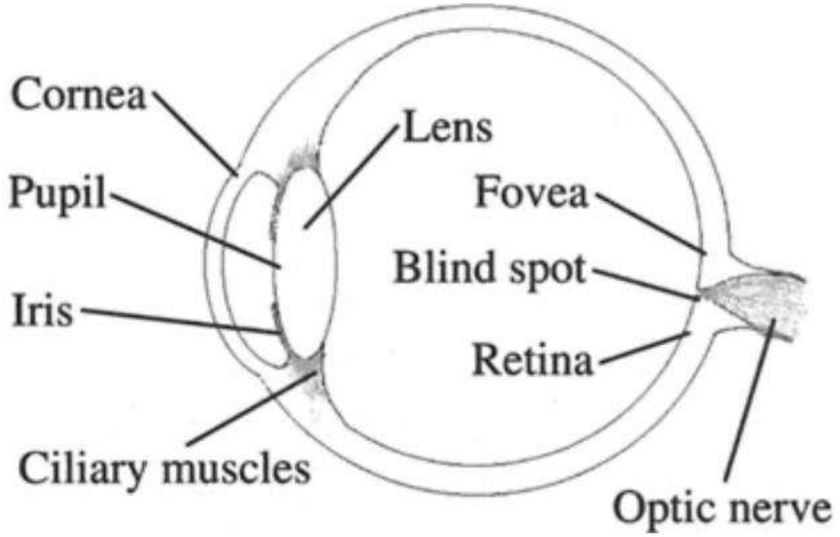
The senses for vision, hearing, smell, taste and equilibrium are the normal *special senses;* whereas the senses for tactile, position, heat, cold, and pain are the *somatic senses*. In this chapter however, we will look only at some relevant facts related to the sensory systems exploited by VR technologies. These include vision, hearing, tactile and equilibrium.

Vision

It is obvious that our visual system is an essential human factor to be taken into account when designing VR hardware and software. The eye and the mechanism of vision is complex, and we have still a long way to go in understanding how the eye and brain work together, not to mention the act of seeing. Fortunately we do not need to know too much about how the eye functions-just enough to understand how it can be best exploited without being abused.

The eye

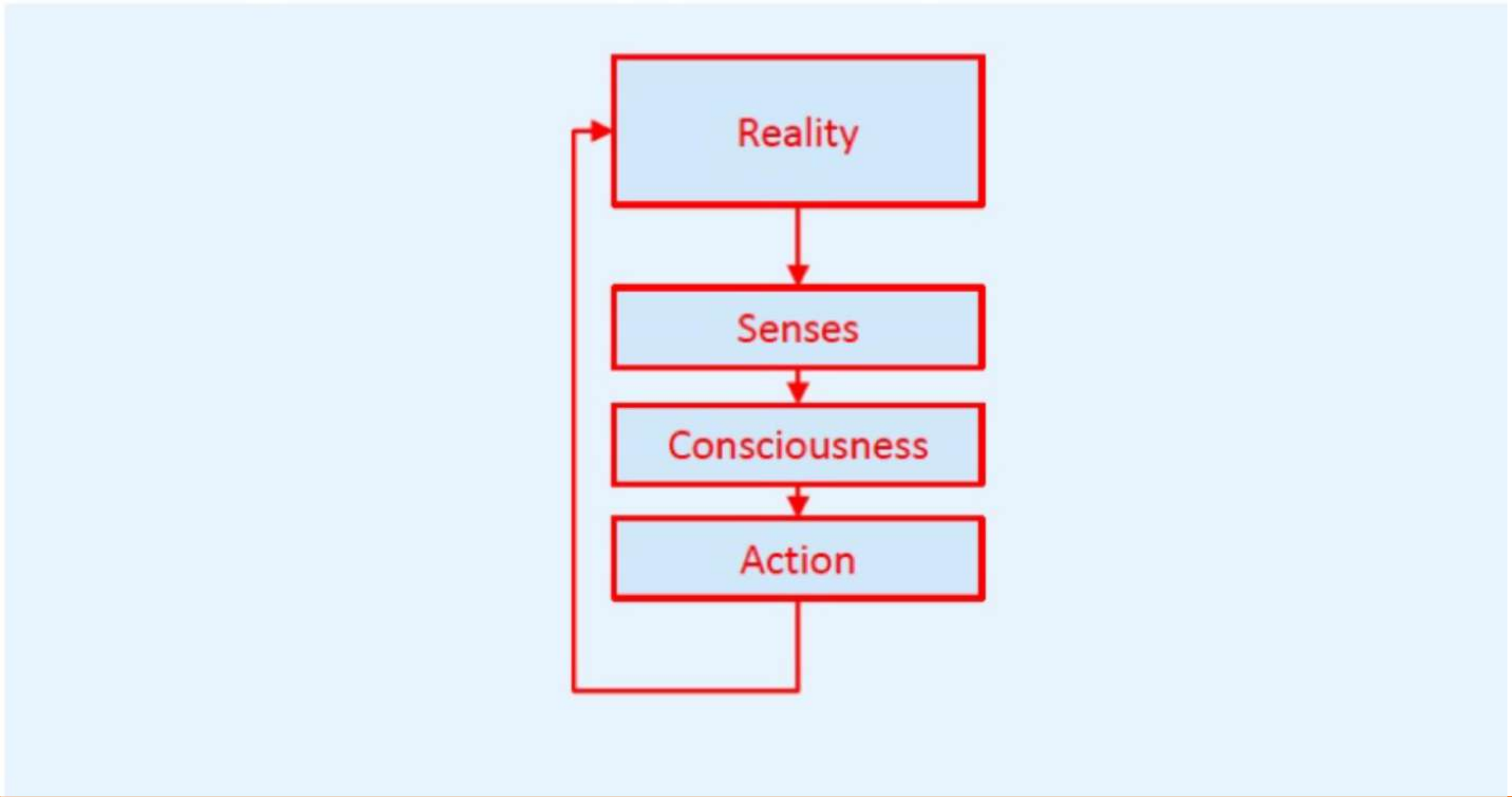
The eye is a sphere enclosing a lens that focuses light upon the light-sensitive retina fixed to the back of the eyeball. From the retina, signals are processed and transported via the optic nerve to the visual cortex where the action of seeing is organized. Light first enters the eye through the transparent *cornea*, and then through a hole called the *pupil* before being refracted by the lens to focus upon the retina. See Fig. But because light is coming from objects at incredible distances such as the moon (400,000 km) and from hand-held objects only a few centimeters away, the lens must be able to automatically adjust its refractive power to keep images in focus. This is called *accommodation* and is achieved by contracting and relaxing the ciliary muscles surrounding the lens.



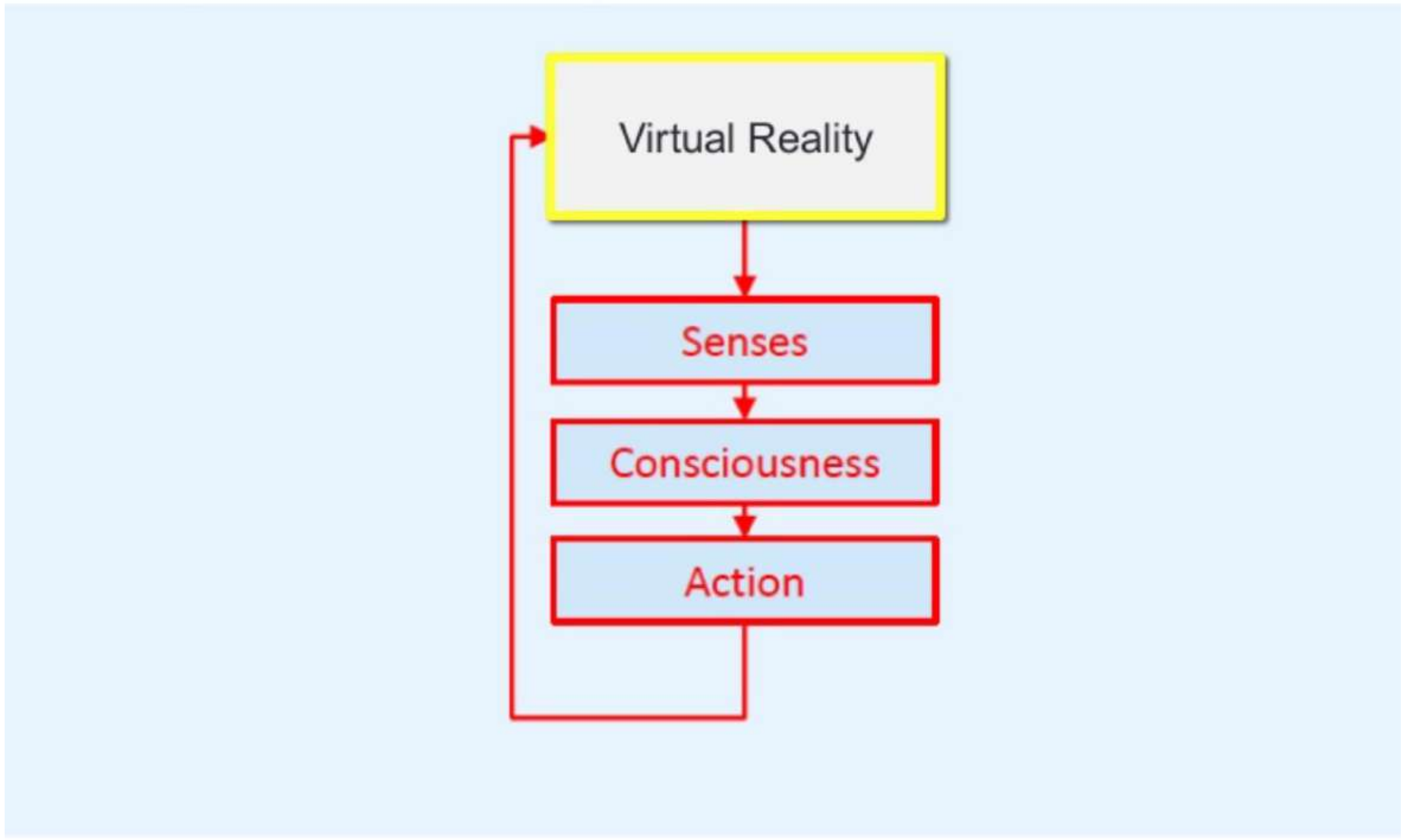
Color receptors

The retina contains two types of light-sensitive cells called *rods* and *cones*. The rods are sensitive to low levels of illumination and are active in night vision; however, they do not contribute towards our sense of color this is left to the cone cells that are sensitive to three overlapping regions of the visible spectrum: red, green and blue. Collectively, they sample the incoming light frequencies and intensities, and give rise to nerve impulses that eventually end up in the form of a colored image.

Simple Sensing/Perception Model



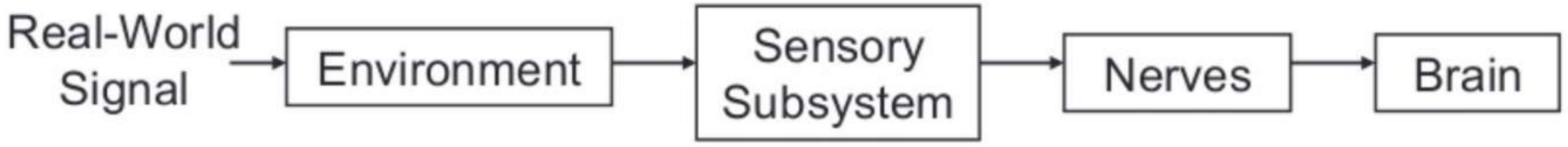
Simple Sensing/Perception Model



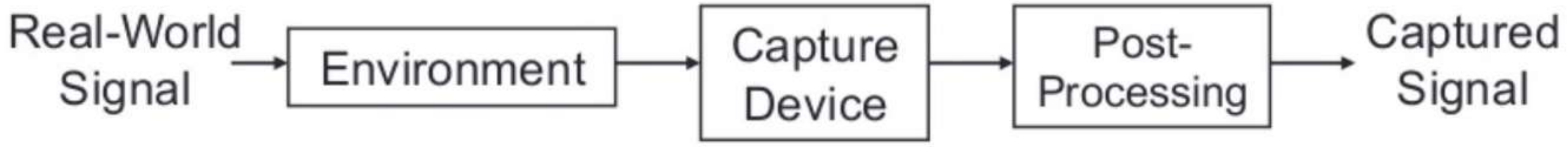
Using VR to stimulate the senses

Real-World Stimulus Paths

Direct Stimulation



Captured/Mediated Stimulation



Can use captured signal to stimulate human sensors

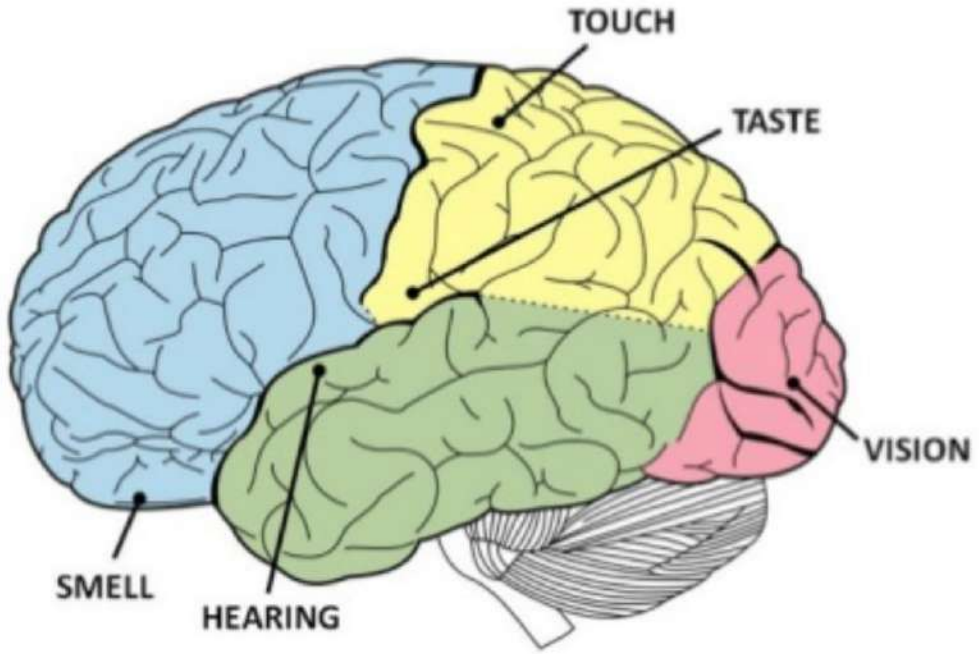
E.g. Listening to recorded music

Senses

- How an organism obtains information for perception:
  - Sensation part of **Somatic Division** of **Peripheral Nervous System**
  - Integration and perception requires the **Central Nervous System**
- Five major senses:
  - Sight (Opthalamoception)
  - Hearing (Audioception)
  - Taste (Gustaoception)
  - Smell (Olfacaoception)
  - Touch (Tactioception)

Relative Importance of Each Sense

- Percentage of neurons in brain devoted to each sense
  - Sight – 30%
  - Touch – 8%
  - Hearing – 2%
  - Smell - < 1%
- Over 60% of brain involved with vision in some way

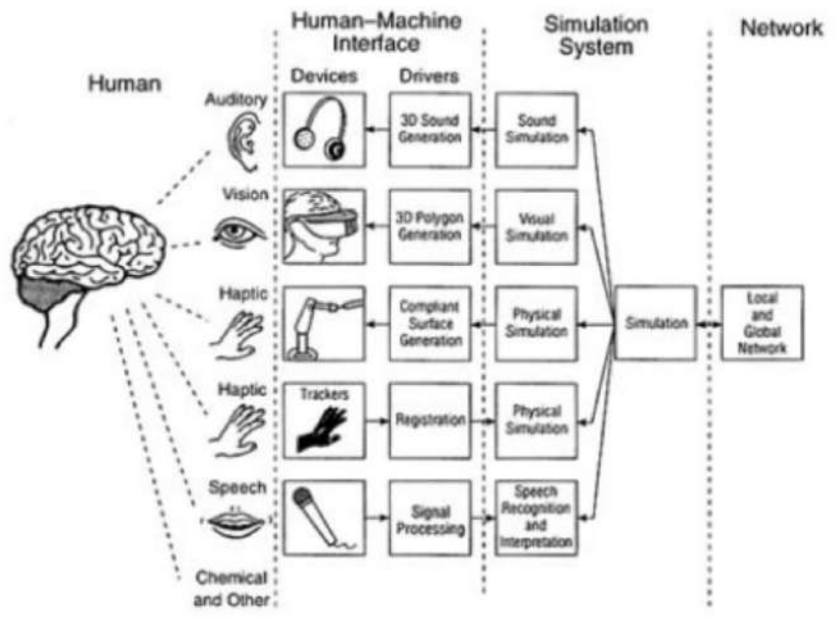


Other Lessor Known Senses..

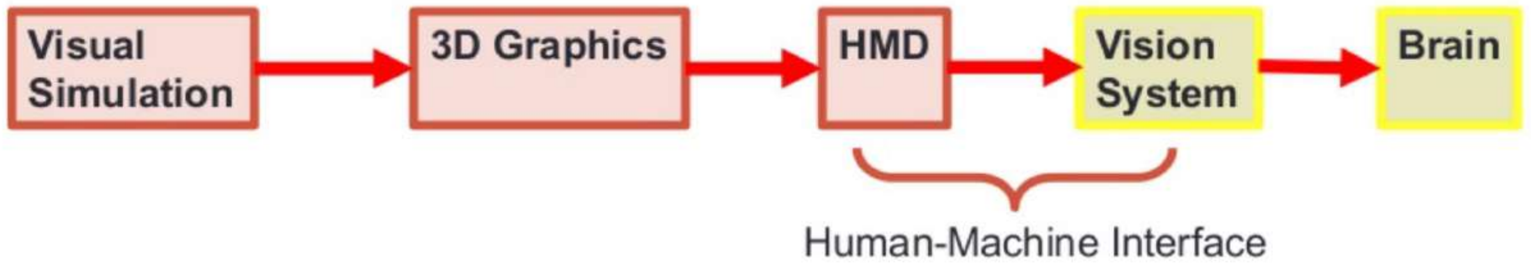
- Proprioception = sense of body position
  - what is your body doing right now
- Equilibrium = balance
- Acceleration
- Nociception = sense of pain
- Temperature
- Satiety
- Thirst
- Micturition
- Amount of CO_2 and Na in blood

VR System Overview

- Simulate output
  - E.g. simulate real scene
- Map output to devices
  - Graphics to HMD
- Use devices to stimulate the senses
  - HMD stimulates eyes



Example: Visual Simulation



Virtual Reality Hardware

VR hardware constitute of sensors which act as transducer to convert the energy it receives into a signal from an electrical circuit. This sensor has receptor to collect the energy for conversion and organism has sense organs such as eyes and ears for the same purpose.

The hardware produces stimuli that override the senses of the user ased on human motions. The VR hardware accomplishes this by using sensors for tracking motions of user such as button presses, controller movements, eye and other body part movements. It also considers the physical surrounding world because only engineered hardware and software does not constitute the complete VR system. The organism (users) and its interaction with the hardware is equally important.

VR hardware constitute of sensors which act as transducer to convert the energy it receives into a signal from an electrical circuit. This sensor has receptor to collect the energy for conversion and organism has sense organs such as eyes and ears for the same purpose. As the user moves through the physical world, it has its own configuration space which are transformed or configured correspondingly.

VR Devices

VR devices are the hardware products used for VR technology to happen. The different key components of VR system are discussed below. The figure (number) shows the high-level view of Virtual World Generator (VWG). The inputs are received from the user and his surroundings and appropriate view of the world are rendered to displays for VR experiences.

Personal Computer (PC)/Console/Smartphone

Computers are used to process inputs and outputs sequentially. To power the content creation and production significant computing power is required, thereby making PC/consoles/smartphones important part of VR systems. The VR content is what users view inside and perceive so it is equally important as other hardware's.

Input devices

Input devices provides users the sense of immersion and determines the way a user communicates with the computer. It helps users to navigate and interact within a VR environment to make it intuitive and natural as possible. Unfortunately, the current state of technology is not advanced enough to support this yet. Most commonly used input devices are joysticks, force Balls/Tracking balls, controller wands, data gloves, trackpads, On-device control buttons, motion trackers, bodysuits, treadmills and motion platforms (virtual omni).

Output Devices

Devices that each stimulate a sense organ. Output devices are used for presenting the VR content or environment to the users and it is utmost devices to generate an immersive feeling. These include visual, auditory or haptic displays. Like input devices, the output devices are also underdeveloped currently because the current state-of-art VR system does not allow to stimulate human senses perfect ideal manners. Most systems support visual feedback, and only some of them are enhanced it by audio or haptic information.

Virtual Reality - Sensors

Feeling the Need for Speed with Accelerometers

The accelerometer is the tiny technical marvel that lets your phone know which way is up and it's how it knows whether to switch to landscape or portrait mode. One accelerometer can measure force in one direction. For example, one accelerometer can tell you if it's pointing towards the ground or not. Gravity exerts force on it when it points down, but not when it points in another direction.

If you increase the number of accelerometers you can measure movement along all three axes. Not only that, but you can also measure how strong the acceleration is. Which means you can do things like translate the strength of a virtual golf swing or some other movement that needs variability in speed and strength.

You Spin me Right Round Like a Gyroscope

If it's the measurement of fine rotation you want to achieve, then a gyroscope is what you need. Just about all of us have seen a gyroscope before. They're sold as educational toys and usually consist of a set of concentric rings with a spinning disc in the middle. Once the disc is spinning you can turn the rings that house it any way you want; the wheel keeps spinning in the same direction. If you measure the direction of the wheel then you'll always know which way is up. This is why airplanes have gyroscopes – to help them achieve level flight even when they have little or no visibility.

Inside your phone or VR controller there's a MEMS gyroscope that helps to make smooth rotational measurement possible. It's not, however, a tiny replication of that mechanical spinning gyroscope. There are in fact quite a few different takes on making a gyro that small, but it gets pretty weird to us normal non-engineering types.

The Attraction of a Magnetometer

A magnetometer is, at the most basic level, a device that can detect a magnetic field. How exactly does that help you measure motion? Well, since the Earth is bathed in a magnetic field the magnetometer can act as an electronic compass. The magnetic north pole of the earth provides an absolute reference point which can be incredibly valuable when you need to do the math around positioning. It's not the most reliable absolute reference point, but it can help.

The Lidless Eye of Optical Tracking

If you've ever checked out the special features of a CG-heavy DVD or Blu Ray you've probably seen actors covered in little white balls and skin tight suits. Think of Andy Serkis as Gollum in the Lord of the Rings or Bill Nighy as Davey Jones in the Pirates of the Caribbean.

What you're seeing is optical motion tracking. Usually infrared light, which is invisible to the naked eye, is beamed out onto the scene and then captured by a special IR-sensitive camera. The dots are places on specific parts of the actor's body and face, so that an accurate wireframe of their motion can be built and then mapped onto a CG character, which is what you see in the film itself.

A Sixth Sense

As you can tell, sensors such as these are critical in making VR a practical technology. The more accurate the sensor or combination of different sensors, the more realistic the VR experience. The future will probably hold even more advanced sensors and we’re already seeing devices that measure muscle movement directly, such as the Myo armband that allows for finger and hand tracking. Some HMD makers are also building eye tracking technology into their products, with a company called Fove leading the market in this regard.

Head-coupled System

The idea for a head-coupled system is 30 (!) years old, having originally been proposed by Ivan Sutherland in been carried out by the military, in the 1970's [Furn-91], and by NASA Ames in the 1980's [FMHR-86]. The first commercial system(VPL Eye Phone[Teit-90]) became available in 1989.

A head-coupled display system is the classical 3D presentation device for virtual reality. It enables a user to feel immersed in a computer-generated environment. Usually, a head-coupled system presents a binocular image(a stereoscopic image with different images for the left and right eyes) to the observer: two monitors display the image pair to the eyes. Some mono scopic display systems also exist. In order to provide a wide field of view, a special optical system is applied. Most systems are directly mounted on the head (head- mounted display, HMD) in the form of a helmet or goggles [CHBF-89]; others need a special mounting(e.g.,BOOM [MBPF-90]).

There are two main categories of head-coupled display systems: opaque systems, and see- through systems:

- **Opaque systems** block the user’s view of the outside world, allowing them to concentrate on the virtual world.
- **See-through** systems superimpose computer-generated environments on to the user’s real surroundings. Figure 1 shows one opaque system(left side: "Datavisor' from n-Vision) and one see-through system (right side: "i-glasses!' from Virtual I/O). Surveys of currently available systems can be found from time to time in various VR newsletters(e.g., [RTG-94]).

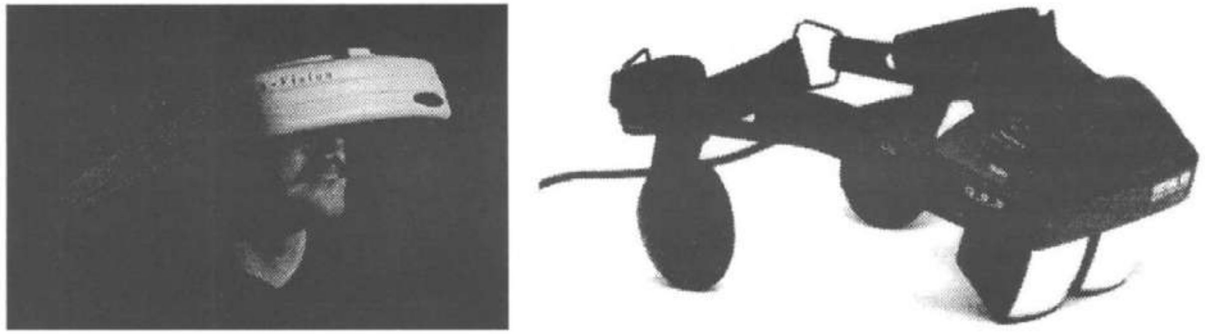


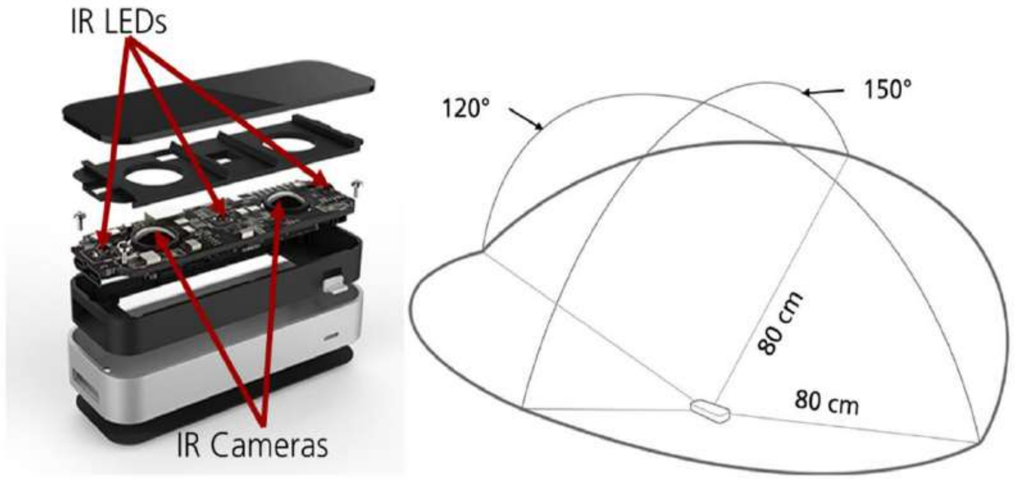
Fig. 1: Head-coupled systems

Integrated VR systems

The **head-mounted display** (**HMD**) is a display device, worn on the head or as part of a helmet (See Helmet-mounted display for aviation applications), that has a small display optic in front of one (monocular HMD) or each eye (binocular HMD). An HMD has many uses including gaming, aviation, engineering, and medicine . Virtual reality headsets are HMDs combined with IMUs. There is also an optical head-mounted display (OHMD), which is a wearable display that can reflect projected images and allows a user to see through it.

A typical HMD has one or two small displays, with lenses and semi-transparent mirrors embedded in eyeglasses (also termed data glasses), a visor, or a helmet. The display units are miniaturized and may include cathode ray tubes (CRT), liquid-crystal displays (LCDs), liquid crystal on silicon (LCos), or organic light-emitting diodes (OLED). Some vendors employ multiple micro-displays to increase total resolution and field of view.

HMDs differ in whether they can display only computer-generated imagery (CGI), or only live imagery from the physical world, or combination. Most HMDs can display only a computer-generated image, sometimes referred to as virtual image. Some HMDs can allow a CGI to be superimposed on real-world view. This is sometimes referred to as augmented reality (AR) or mixed reality (MR). Combining real-world view with CGI can be done by projecting the CGI through a partially reflective mirror and viewing the real world directly. This method is often called optical see-through. Combining real-world view with CGI can also be done electronically by accepting video from a camera and mixing it electronically with CGI.



VR Software

VR Software exits in many forms and for many VR applications. These include:

Three Dimensional Modeling Software-- Since virtual reality is a three-dimensional medium, all the objects in a virtual world must be specified such that they can be viewed from any angle. A simple picture of an object is not sufficient. The actual geometry of the objects must be specified using 3D modeling software, and then imported into the virtual environment.

Two-Dimensional Graphics Software--Since there is a limit to the geometric complexity of the objects in a virtual world, it is often useful to be able to "paint" the surfaces of the objects with additional detail. This process is called texture mapping and requires two-dimensional graphics software often called "paint programs". Images created with this software (or photographed, digitized, and then edited with this software) can then be stretched over the geometric framework produced in the 3D modeling software to create detailed, interesting objects.

A few examples of 2D graphics software: Adobe Photoshop, Adobe Illustrator, Fractal Design Painter

Digital Sound Editing Software--Sound is a very important, yet often neglected aspect of virtual reality. Although the state-of-the-art in 3D graphics is very impressive to look at, it is not very realistic, and does not approach the potential of the human perceptual system: a virtual environment is low resolution, looks like computer graphics, and is not likely to be mistaken for the real thing. Sound reproduction technology is much further along (is it live or is it Memorex?). Digital sound is nearly as high resolution as human hearing. Digital editing software allows you to cut, splice, mix, and loop the sounds of your virtual environment.

Simulation Software--Virtual Reality requires sophisticated software in order to provide a compelling experience. The software must be able to process the inputs coming from trackers and input devices, and then update the displays at least 20 times each second. This is further complicated by the fact that there may be more than one person in the virtual world, more than one set of inputs and displays, and more than one networked computer running the simulation. The software creates and maintains an internal database of all the objects in the virtual world, continually monitors changes to the database, and distributes this information to all the computers participating in the virtual world. **Eg:** of VR Simulation Software: , Sense-8, Division, Superscape, Cosmo, VRML

PHYSICAL SIMULATION

Physical Simulation of materials processing involves the exact reproduction of the thermal and mechanical processes in the laboratory that the material is subjected to in the actual fabrication or end use. A small sample of the actual material is used in the simulation. The material follows the same thermal and mechanical profile that it would in the full scale fabrication process or end use of the material. Depending on the capability of the machine performing the simulation, the results can be extremely useful. When the simulation is accurate, the results can be readily transferred from the laboratory to the full size production process.

What Is the Virtual Reality Toolbox?

The Virtual Reality Toolbox is a solution for interacting with virtual reality models of dynamic systems over time. It extends the capabilities of MATLAB and Simulink into the world of virtual reality graphics.

- **Virtual worlds** — Create virtual worlds or three-dimensional scenes using standard Virtual Reality Modeling Language (VRML) technology.
- **Dynamic systems** — Create and define dynamic systems with MATLAB and Simulink.
- **Animation** — View moving three-dimensional scenes driven by signals from the Simulink environment.

Manipulation — Change the positions and properties of objects in a virtual world while running a simulation.

To provide a complete working environment, the Virtual Reality Toolbox includes additional components:

• **VRML viewer** — Use either the Virtual Reality Toolbox viewer or, for PC platforms, the blaxxun Contact plug-in for Web browsers to display your virtual worlds.

• **VRML editor** — For PC platforms, use V-Realm Builder to create and edit VRML code. For UNIX or Linux platforms, use the MATLAB text editor to write VRML code to create virtual worlds

Introduction to VRML.

VRML Overview

The Virtual Reality Modeling Language (VRML) is the language you use to display three-dimensional objects with a VRML viewer.

This section includes the following topics:

•"VRML History" on page 1-10 — Events leading up to the creation of the VRML97 standard.

•"VRML Coordinate System" on page 1-11 — The VRML coordinate system is different from the MATLAB coordinate system.

•"VRML File Format" on page 1-13 — VRML files use a hierarchical structure to describe three-dimensional objects and their movements.

VRML Support

The Virtual Reality Modeling Language (VRML) is an ISO standard that is open, text-based, and uses a WWW-oriented format. You use VRML to define a virtual world that you can display with a VRML viewer and connect to a Simulink model.

The Virtual Reality Toolbox uses many of the advanced features defined in the current VRML97 specification. The term VRML, in this guide, always refers to VRML as defined in the VRML97 standard ISO/IEC 14772-1:1997, available from http://www.web3d.org. This format includes a description of 3-D scenes, sounds, internal actions, and WWW anchors.

MATLAB Interface

The Virtual Reality Toolbox provides a flexible MATLAB interface to virtual reality worlds. After creating MATLAB objects and associating them with a virtual world, you can control the virtual world by using functions and methods.

Simulink Interface

With a Simulink model, you can observe a simulation of your dynamic system over time in a visually realistic 3-D model.

The Virtual Reality Toolbox provides blocks to directly connect Simulink signals with virtual worlds. This connection lets you visualize your model as a three-dimensional animation.

You can implement most of the Virtual Reality Toolbox features with Simulink blocks. Once you include these blocks in a Simulink diagram, you can select a virtual world and connect Simulink signals to the virtual world. The Virtual Reality Toolbox automatically scans a virtual world for available VRML nodes that Simulink can drive.

VRML Viewers

The Virtual Reality Toolbox contains a viewer that is the default viewing method for virtual worlds. This Virtual Reality Toolbox viewer is supported on PC, UNIX, Mac OS X, and Linux platforms.

If you are on a PC platform, you can install a VRML plug-in and view a virtual world in your preferred Web browser. For PC platforms, the Virtual Reality Toolbox includes the VRML plug-in blaxxun Contact. This is the only supported VRML plug-in.

VRML Editor

For PC platforms, the Virtual Reality Toolbox includes one of the classic VRML authoring tools, V-Realm Builder by Ligos Corp. With the addition of this VRML authoring tool, the Virtual Reality Toolbox provides a complete authoring, development, and working environment for carrying out 3-D visual simulations.

You use a VRML editor to create the virtual worlds you connect to Simulink block diagrams:

• **PC platforms** — V-Realm Builder Version 2.0 is included with the Virtual Reality Toolbox. If you do not want to use V-Realm Builder, you can use your favorite VRML editor.

Use the command vrinstall to install the editor before editing a virtual world. See “Installing the VRML Editor (Windows)” on page 2-29.

For information on using V-Realm Builder with the Virtual Reality Toolbox, see Chapter 5, “Virtual Worlds.”

• **UNIX/Linux platforms** — The default VRML editor for UNIX/Linux platforms is the MATLAB editor. If you do not want to use the MATLAB editor, you can set the Editor preference to your favorite text editor.

Real-Time Workshop Support

The Virtual Reality Toolbox seamlessly integrates with Real-Time Workshop targets. It supports simulations that use code generated by Real-Time Workshop and a third-party compiler on your desktop computer. The Virtual Reality Toolbox also supports code executed in real time on external target computers. It enables interaction with real-time code generated by Real-Time Workshop and compiled with a third-party C/C++ compiler.

Real-Time Windows Target

The Simulink interface in the Virtual Reality Toolbox supports the Real-Time Windows Target. Using the Simulink external mode, you can interact with real-time code generated by Real-Time Workshop and compiled with a third-party C/C++ compiler in the Real-Time Windows Target environment. See the Real-Time Windows Target User's Guide documentation for further details.

SimMechanics Support

You can use the Virtual Reality Toolbox to view the behavior of a model created with SimMechanics. First, you build a model of a machine in Simulink using SimMechanics blocks. Then, create a detailed picture of your machine in a virtual world, connect this world to the SimMechanics body sensor outputs, and view the behavior of the bodies in a VRML viewer.

Hardware Support

The Virtual Reality Toolbox contains functions for using special hardware devices, including Joystick and Space Mouse. It can also connect to common hardware devices, including joysticks and Magellan Space Mouse, using Simulink blocks.

Client-Server Architecture

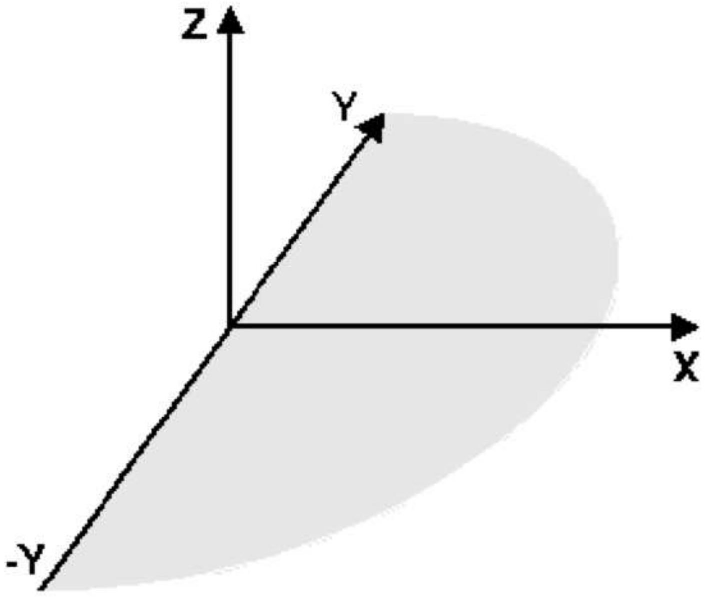
The Virtual Reality Toolbox connects MATLAB and Simulink to a VRML-enabled Web browser using the TCP/IP protocol. The toolbox can be used in two configurations:

Single computer — MATLAB, Simulink, and the virtual reality representations run on the same host computer.

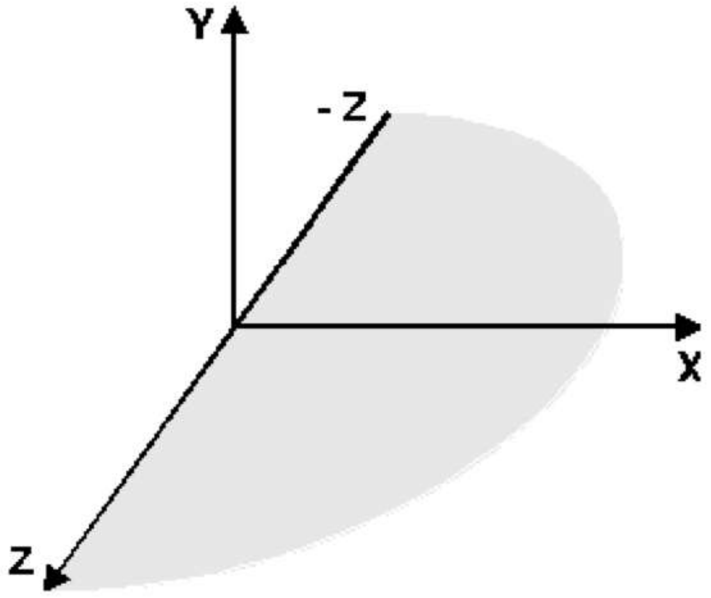
Network computer — You can view an animated virtual world on a computer separate from the computer with the Virtual Reality Toolbox server. Multiple clients can be connected to one server. You can adjust parameters to tune network performance.

VRML Coordinate System

VRML uses the right-handed *Cartesian coordinate system*. If your thumb, index finger, and middle finger of the right hand are held so that they form three right angles, then your thumb symbolizes the *x*-axis, your index finger the *y*-axis (pointing up), and your middle finger the *z*-axis.

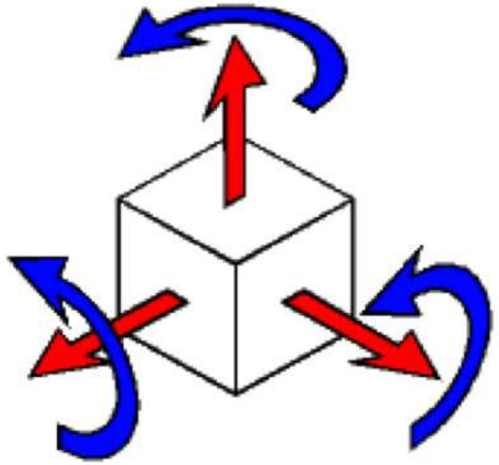


MATLAB graphics coordinate system



VRML coordinate system

Rotation angles — In VRML, rotation angles are defined using the *right-hand rule*. Imagine your right hand holding an axis while your thumb points in the direction of the axis toward its positive end. Your four remaining fingers point in a counter clockwise direction. This counter clockwise direction is the positive rotation angle of an object moving around that axis.



Child objects — In the hierarchical structure of a VRML file, the position and orientation of child objects are specified relative to the parent object. The parent object has its local coordinate space defined by its own position and orientation. Moving the parent object also moves the child objects relative to the parent object.

Measurement units — All lengths and distances are measured in *meters*, and all angles are measured in *radians*.

VRML File Format

You need not have any substantial knowledge of the VRML format to use the VRML authoring tools to create virtual worlds. However, it is useful to have a basic knowledge of VRML scene description. This helps you to create virtual worlds more effectively, and gives you a good understanding of how the virtual world elements can be controlled using the Virtual Reality Toolbox.

This section introduces VRML. For more information, see the VRML97 Reference. This reference is available online at http://www.web3d.org. Many specialized VRML books can help you understand VRML concepts and create your own virtual worlds. For more information about the VRML, refer to an appropriate third-party VRML book.

In VRML, a 3-D scene is described by a hierarchical tree structure of objects (nodes). Every node in the tree represents some functionality of the scene. There are 54 different types of nodes. Some of them are *shape nodes* (representing real 3-D objects), and some of them are *grouping nodes* used for holding child nodes. Here are some examples:

- **Box node** — Represents a box in a scene.
- **Transform node** — Defines position, scale, scale orientation, rotation, translation, and children of its subtree (grouping node).
- **Material node —** Corresponds to material in a scene.
- **Directional Light node —** Represents lighting in a scene.
- **Fog node —** Allows you to modify the environment optical properties.
- **Proximity Sensor node —** Brings interactivity to VRML97. This node generates events when the user enters, exits, and moves within the defined region in space.

Thank You