Virtual Environments

Introduction

What are virtual environments?

Virtual environments, sometimes also referred to as virtual reality, is essentially a medium composed of interactive computer simulations that sense the participant's position and actions and replace or augment the feedback to one or more senses, giving the feeling of being mentally immersed or present in the simulation (a virtual world) (Sherman & Craig, 2003)

Introduction

There are some key elements that are necessary to achieve an immersive experience:

- Virtual world
- Full immersion
- Interactivity

Introduction

From a technological point of view the following features are required:

- The visual (possibly also aural and haptic) displays that immerse the user in the virtual world and that block out contradictory sensory impressions from the real world
- The graphics rendering system that generates, at 20 - 30 frames/second, the ever-changing images
- The tracking system that continually reports the position and orientation of the user's head and limbs
- The database construction and maintenance system for building and maintaining detailed and realistic models of the virtual world

Display technology

There are several different display types that can be used for virtual environments.

- Head-mounted displays
- CAVE-type displays
- DOME-type displays
- Panoramic screens
- Workbenches
- 3D TVs
- Fishtank VR

Display Technology

Head mounted display: Sony HMZT1

Display Technology

More modern head-mounted devices (HMDs) provide better resolution and wider field-of-view (FoV).

Examples are HTC Vive, Occulus devices…

HTC Vive typically are programmed using SteamVR whereas Mixed Reality devices (Microsoft's built-in XR support) is now using OpenXR (a standard developed by the Kronos Group who is also doing OpenGL and other standards).

Display Technology

There are head-mounted displays that support augmented reality as well in addition to AR support provided by cell phones or tablets. Examples are:

- Microsoft's Hololens

- Magic Leap 1

Display Technology

Head-mounted displays

Advantages

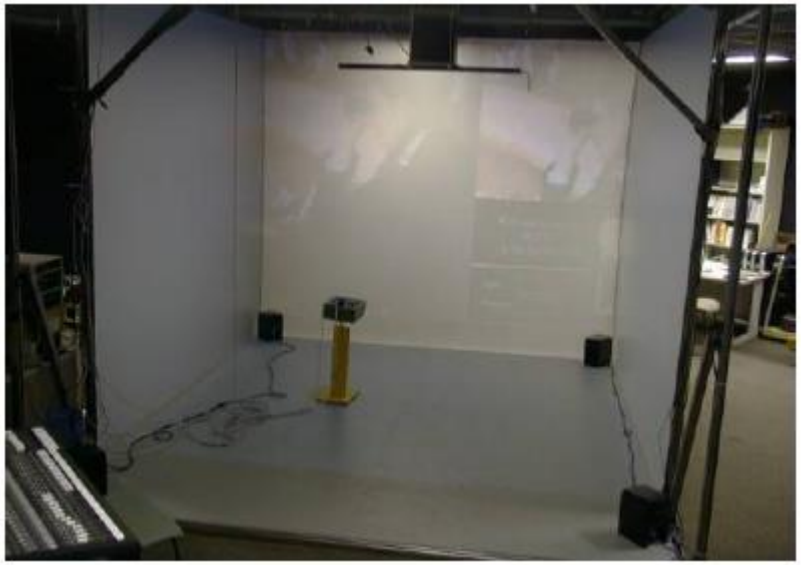
- Provide full-immersion and visible “screen space” in all directions.
- Less expensive

Disadvantages

- Limited screen resolution (often 800x600, 1280x720, 2160x1080)
- Need to be careful not to walk into real objects

Display Technology

CAVE-type displays: Mechdyne CAVE or Barco's I-Space



CAVE at NCSA (National Center for Supercomputing Applications) at UIUC (http://cave.ncsa.uiuc.edu/about.html)

Display Technology

CAVE™

- Provides the illusion of immersion by projecting stereo images on the walls and floor of a room-sized cube

Advantages

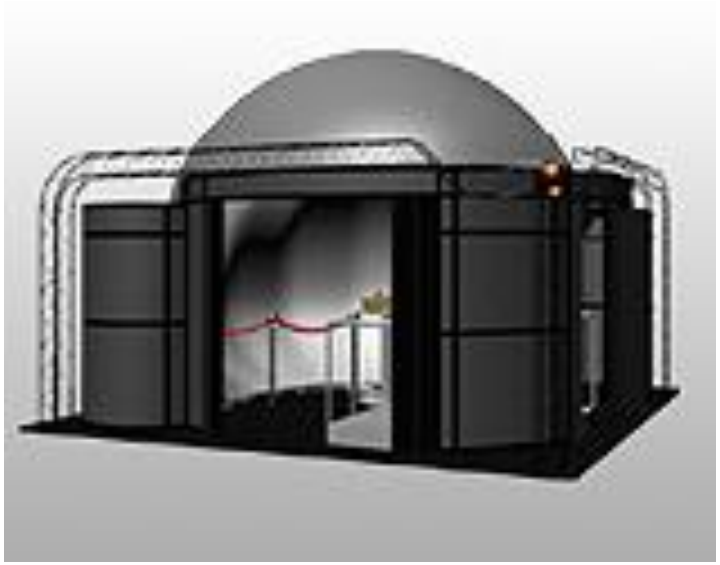
- A wide surrounding field of view
- The ability to provide a shared experience to a small group

Disadvantages

- The cost of multiple image generation systems (although not a serious limitation nowadays)
- Space requirement for rear projection
  - 4-8 feet or more, depending on the size of the screen
- Brightness limitation due to large screen size
  - Result in scenes of approximately full-moon brightness and hinder color perception
- Corner and edge effects that intrude on displayed scenes
  - An alternative is to use Dome systems in which imagery is projected onto a hemisphere surrounding

Display Technology

DOME-type displays



An illustration of DOME

Display Technology

DOME-type displays

Advantages

- Provide full-immersion and visible “screen space” in all directions.

Disadvantages

- Typically requires even more projectors than CAVE-type displays
- Edge blending is even more complex than calibration for CAVE-type displays

Display Technology

Panoramic Displays

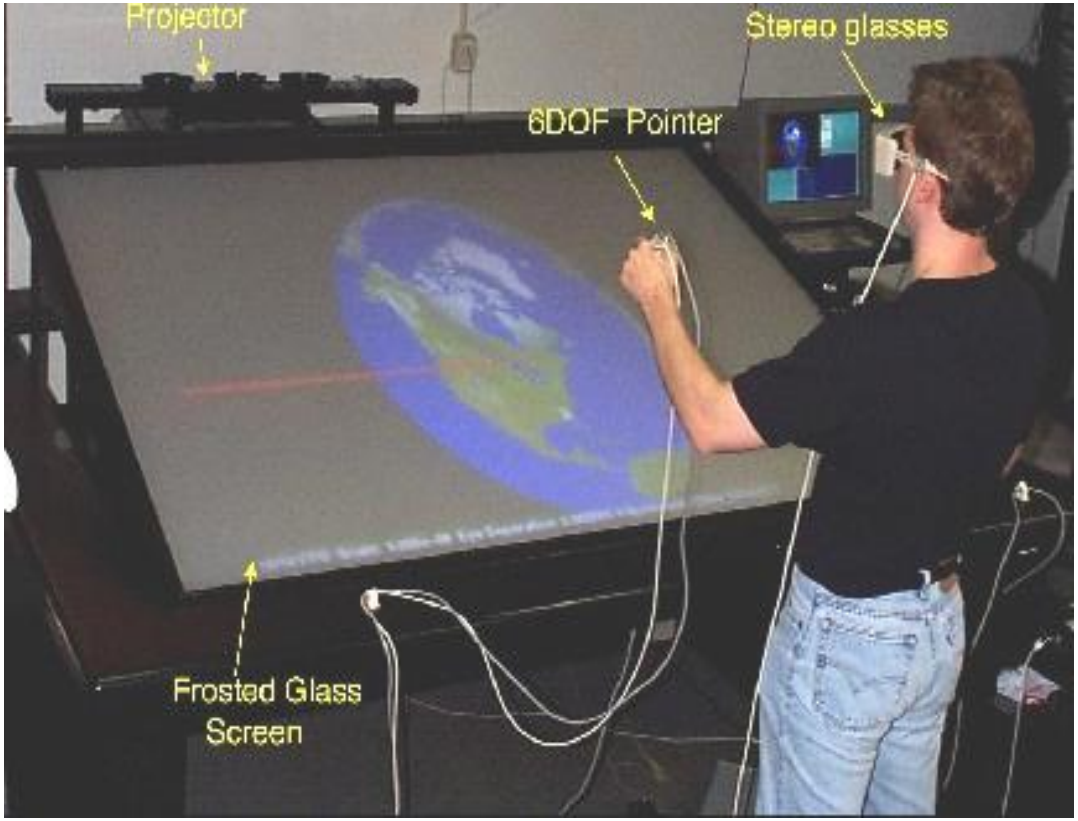
- One or more screens arranged in a panoramic configuration, or a single, curved screen on which images from multiple projectors are tiled together
- Especially suit groups; multidisciplinary design reviews commonly use this type of display
- One person drives the viewpoint

Issues: Edge blending, viewpoint-dependent distortion correction, viewpoint-dependent gain correction

CURV™, by Mechdyne

Display Technology

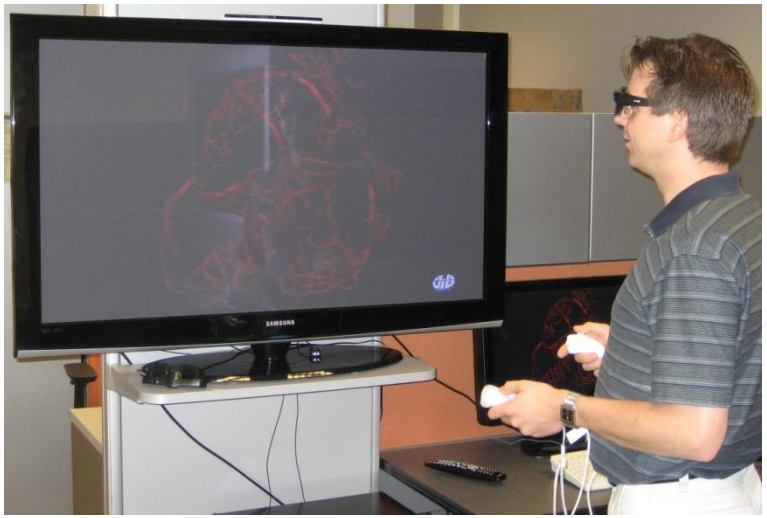
Workbenches: Flat, rear-projection screens that display images in stereo and can be set up in a horizontal or tilted position



Display Technology

3D TVs

Nowadays, 3D TVs got introduced into the consumer market, which significantly brought down the cost of 3D display technology. Based on the same technology as previous displays, they can also be driven by a computer to achieve an immersive effect.



Display Technology

Fishtank VR

- A desktop VR system in which images are displayed on a desktop monitor, usually in stereo, and coupled to the location of the head which is tracked, resulting in the illusion of looking into a “fishtank”
- Commonly applied in CAD and design areas where immersion is not of much significance

Fishtank VR
(http://www.faw.uni-linz.ac.at/save/)

Display Technology

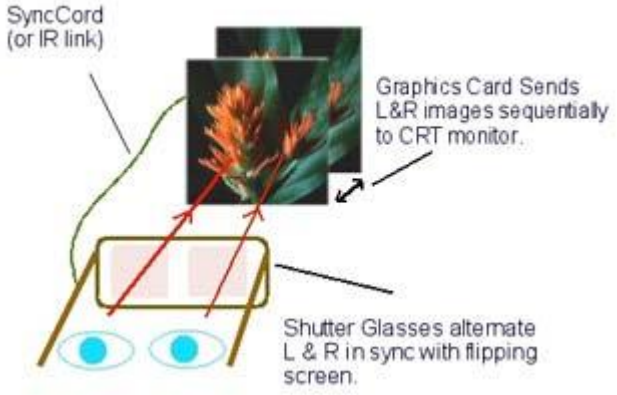
In order to achieve a 3D, different images need to be provided for the left and right eye. The computer can then generate different perspectives of the virtual world as they would be perceived from the left and right eye, respectively. All of the previously described systems, except head-mounted displays, use some form of glasses to achieve this effect. There are basically two different types of glasses:

- Active shutter glasses
- Passive polarized glasses

(There are also auto-stereoscopic displays that use some form of barrier screen to achieve the same effect without the need for glasses.)

Display Technology

To achieve a 3-D viewing effect, shutter-glasses can be used to show different images to the left and right eye. The shutter-glasses use filters that can block or let pass incoming light. This property can be changed electrically. By showing alternating images for the left and right eye and letting the light pass only for the corresponding eye, 3-dimensional viewing is possible.



Display Technology

Polarized glasses utilize polarized light, often horizontally and vertically polarized light, to achieve the same effect as active shutter glasses. For example, the left image uses horizontally polarized light and for the right image vertically polarized light is used. Then a simple polarization filter can be used inside the glasses to separate the left and right image. For projector-based displays this requires two projectors, each of them with a polarization filter mounted in front of the lens to polarize the outgoing light appropriately. The biggest advantage of these types is that the glasses can be very cheaply made.

Display Technology

Brightness

There are several factors that influence of the brightness when using display technology suitable for virtual environments:

- Screen size (when using projectors)
- Polarization filters
- Active shutter glasses
- Rear-projection vs. front projection

With all these brightness-reducing factors, very powerful high-end projectors are often used for these displays.

Display Technology

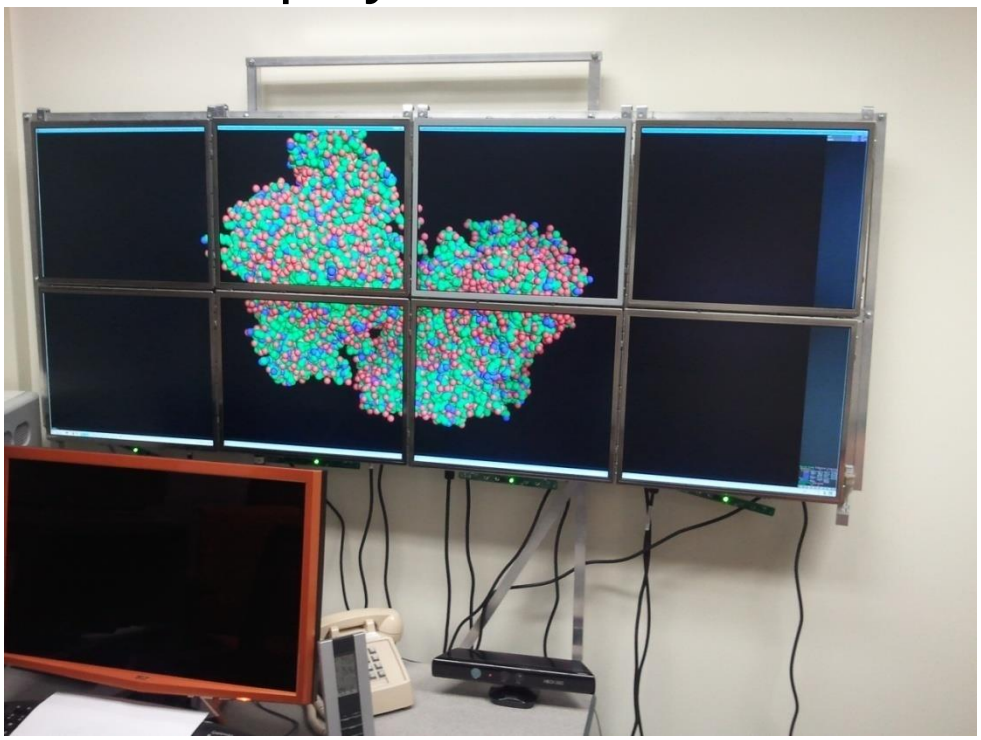
Screen resolution

Typically, the screen resolution of a specific display is fixed. However, there are ways of increasing the screen resolution if needed. For example, very large displays often require a higher resolution to avoid pixelation effects. For this, tiled displays can be used by either using more than one monitor or projector:

Display Technology

Screen resolution

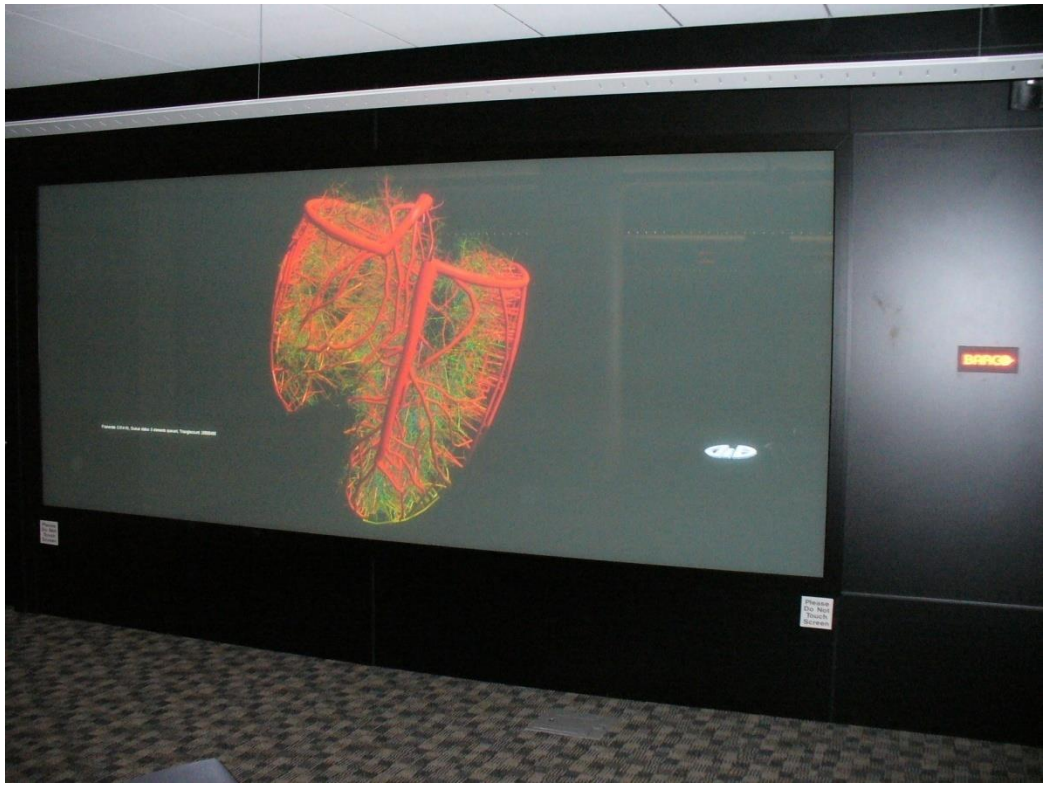
Tiled monitor-based display:



Display Technology

Screen resolution

Tiled projector-based display:



Motion Tracking

In order to achieve a fully immersive effect, the system needs to know where the user is located at. With that information, the system can adapt the view settings based on the users location. For example, an effect can be achieved such that the user gets the impression of actually being surrounded by the virtual environment and every move of the user translates into a move within the virtual environment.

Motion Tracking

Mechanical Tracker

- A simple mechanical tracker can take the form of mechanical arm attached to the tracked object
- Very useful when integrated with a hand-held device
- High accuracy and low latency due to its electromechanical nature
- Restricted active volume (movement)

Phantom Omni (Sensable)

Motion Tracking

Optical Tracker

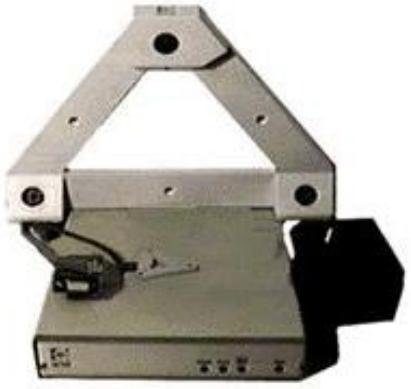
- Infrared video cameras that record the movement of a person
  - Attached to the person is a collection of markers in the form of small marker spheres fixed to a critical joints
  - When the moving person is illuminated with infrared light the marker balls are readily detected within the video images
- Fast and low latency
- The system depends on the line-of-sight, so the orientation of the cameras must ensure that the markers are always visible
- Often prone to interference caused by ambient lighting conditions

ARTTrack1 and ARTTrack2, by Advanced Realtime Tracking Inc. (http://www.ar-tracking.de)

Motion Tracking

Ultrasonic Tracker

- Ultrasonic sound waves are used to locate the user's position and orientation
- Usually used for fishtank VR in which the ultrasonic tracker is placed on the top of the monitor and records the user's head movements
- Simple and low cost
- Slow, restricted active volume, sensitive to temperature and depends on the line-of- sight



Logitech Ultrasonic Head Tracker
(http://www.i-glassesstore.com/logtractracs.html)

Motion Tracking

Electromagnetic Tracker

- Employ a device called a source that emits an electromagnetic field, and a sensor that detects the radiated field
  - The source, which can be no bigger than a 2-inch cube, can be placed on a table or fixed to a ceiling
  - The sensor is even smaller and is readily attached to an HMD or fitted within a 3D mouse
- Fast and very low latency; no light-of-sight restriction
- Restricted active volume and are prone to interference of metallic objects

miniBIRD, by Ascension Technology Corp. (http://www.ascension-tech.com/products/minibird.php)

Motion Tracking

Other game-type devices can also be used for tracking purposes, such as:

Input Devices

For virtual devices, traditional input devices, such as keyboard and mouse, typically do not work well. For once, most virtual environment do not have a table to put them on. Also, their 3D capabilities are limited at best.

Hence, other more specialized input devices are needed. Often times, these devices are tracked as well to provide the capability of defining points in 3D.

Input Devices

SpaceMouse/SpaceBall

- Hand-held device containing a tracker sensor and some buttons, used for navigating or picking objects within a VE
- 6 DOF operations
  - Transitions in X, Y, Z axes and rotations around X (pitch), Y (yaw), and Z (roll) axes
  - Some allow zooming in/out objects

SpaceMouse™, SpaceBall™ 5000, by 3DConnexion Corp. (http://www.vrlogic.com/html/3dconnexion/3d_connexion.html)

Input Devices

Gloves

- Gloves equipped with sensors that track the user's hand movement
- Enable natural interaction with objects
- Modern VR gloves are used to communicate hand gestures (such as pointing and grasping) and in some cases return tactile signals to the user's hand

Pinch gloves

Accele gloves

Haptic Devices

Haptic devices provides a sense of touch with computer generated environments, so that when virtual objects are touched, they seem real and tangible.For example, a medical training simulator in which a doctor can feel a scalpel cut through virtual skin, feel a needle push through virtual tissue, or feel a drill drilling through virtual bone

Current Technologies

- Force feedback joystick
- Virtual styluses
  - Sensable phantom series, by Sensable Technologies
- Virtual gloves
  - Immersion cyber series, by Immersion Corp.

Force Feedback Joystick

Force Feedback Joystick

A device allowing the users to feel force of magnitude and orientation, aside from measurement of depression and twist of its stick

Sensable Phantom Series

By Sensable Technologies (http://www.sensable.com)

- Positional sensing: X, Y, Z, pitch, roll, yaw
- Force feedback: X, Y, Z
- Range of motion: hand movement pivoting at wrist
- Maximum force: 1.8 lbs
- Intended for use in haptic research and free-form modeling

- Positional sensing: X, Y, Z, (pitch, roll, yaw with an additional separate encoder stylus gimbal)
- Force feedback: X, Y, Z
- Range of motion: hand movement pivoting at wrist
- Maximum force: 1.9 lbs

Virtual Styluses

Advantages

- Inexpensive
- Easy to set up and operate
- Works on a desktop
- Well suited for remote manipulation

Disadvantages

- Not immersive
- Haptic response at a single point only

Virtual Gloves

Immersion cyber series, by Immersion Corp.
(http://www.immersion.com/)

- Senses position of finger; no force feedback

- Adds tactile feedback to CyberGlove using vibrations on fingertips or palm
- Limited to simple pulses or sustained vibration

Force Feedback

- Force feedback for fingers and hand

- Force feedback for hand and arm
- Can be used together with CyberGrasp

Virtual Gloves

Advantages

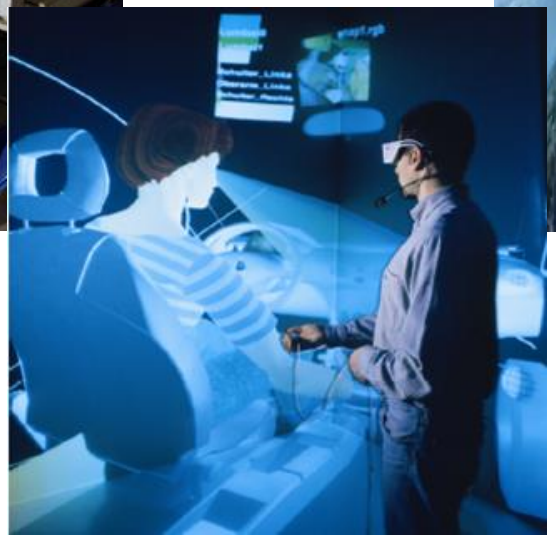
- Multiple points of haptic and tactic responses
- Allows for full immersion with HMDs

Disadvantages

- Expensive
- Difficult to set up and operate

Applications for Virtual Environments

There are numerous application examples for virtual environments ranging from engineering, medical, or psychology:



Setup

First, we need to pick the technological components for our virtual environments system:

- Display system
- Tracking system
- Input devices

As often the case, budgetary concerns play an important role in picking those components since hardware for virtual environments tend to be pricy (even though prices are coming down now that this technology entered the consumer market).

Software

There are software packages that already are aware of virtual environment hardware, in which virtual environments can be created and then readily be displayed. For example,

- VegaPrime by Presagis
- Vizard by WorldViz

However, these are more on the expensive side (often with an annual subscription) and they may limit the capabilities, particularly on the research egde.

Software

Other software packages let you interface directly with graphics libraries, such as OpenGL or OpenSceneGraph. Examples are:

- CAVElib
- VRJuggler
- FreeVR
- Vrui

CAVElib is a commercial product with considerable annual fees. VRJuggler is free software based on Java. Similarly FreeVR and Vrui are open-source C/C++ libraries that currently only work in Linux/Unix.

Software

FreeVR

FreeVR develop by Bill Sherman provides a C interface in which callback functions are called for initialization, drawing, and input events. It readily supports OpenGL and OpenSceneGraph code where the drawing parts of existing software can simply be added to the drawing callback function. The handling of rendering to different screens and input devices is then automatically handled.

Software

Vrui

Similar to FreeVR, Vrui utilizes callback functions to provide an interface. Since it is C++-based, these callback functions are encapsulated in a class derived from Vrui::Application and GLObject. Vrui is one of the few libraries that supports multi-pass rendering, such that the rendering can be split amongst a set of computers. This is particularly important for more complex display setups, such as CAVE or panoramic displays, with several display screens.

Software

Vrui

To support multi-pass rendering, Vrui uses the ssh mechanism to log into remote computers and start the rendering program. As a result, it requires an additional object derived from GLObject::DataItem that contains all necessary information for the rendering step. As such, this object needs all information to recreated any graphics content if necessary.

Software

Vrui

The actual rendering is done in the method `display`, which can be initialized in `initContext`. Both functions take a parameter, which is the object described on the previous slide. The `display` method is going to be called whenever any of the screens need updated. This means that if the display consists of several screens the `display` method may be called several times.

There is also a method `frame` that is only called exactly once per frame that can be used for updated internal states.

Calibration

For almost any virtual environment, calibration is a crucial step. There are several calibration steps that have to be performed.

First, the tracking system itself requires calibration in which a coordinate system is defined for the tracking system.

Then, this coordinate system needs to be aligned with the display coordinate system. Basically, the system needs to know what the coordinates of the display(s) is/are.

Calibration

We can then scale the coordinates such that everything is measured in standard units, for example inch.

This allows us to measure the entire space for the virtual environment using those units. This allows us to measure the distance between the user and display as well as the corners of the display itself.

This is important as the software needs to know the physical location of the display within the tracking coordinate system in order to achieve an immersive effect.

Calibration

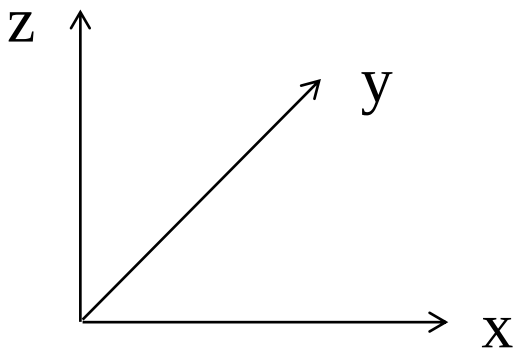
Vrui

In Vrui, the tracking coordinate system can be manipulated before the actual virtual environment program sees it. This is done via the VRDeviceDemon which encapsulates any type of input device in a generic interface. This way, it does not matter to the virtual environment software what type of input device is used. Hence, very heterogenic hardware environments can be created which allows the exact same software to run on various different display systems.

Calibration

Vrui

Since Vrui allows you to change the tracking coordinate system, it is often easier to transform it into a standardized coordinate system using standard measurements, such as inch:



Then, the lower left corner and axes of each display can be actually measured in that coordinate system.

Virtual Environments

There are several software solution for creating virtual environments. Some are commercial, such as VegaPrime, but some are open-source. Examples for open-source projects that are useful for creating virtual environments are:

- Blender: create 3D models
- Cal3D: skeletal-based 3D character animation library
- Delta3D: 3D game engine based on open scene graph
- Panda3D: Another 3D game engine
- Google's Sketchup

Vru integration

Since Vrui is OpenGL-based, most OpenGL-based software can be rewritten to work with Vrui.

The rendering part of the existing software needs to be called within the `display` method of the application class derived from `Vrui::Application`. Optionally, you can use the `initContext` method to create a display list of the rendering and then call that display list to increase rendering performance.

Vru integration

Even software based on Open Scene Graph (OSG) can be modified to effectively just create OpenGL renderings even though Open Scene Graph applications themselves typically handle the window management.

For this, we can use the OSG viewer class and set the view settings (modelview, projection, viewport) to what OpenGL is using and then tell it to render. This way it will render exactly the way we want it to, i.e. by assuming the exact same view settings, once we call its `renderingTraversal` method.

Vru integration

By default Open Scene Graph, creates a head light which can ruin the immersive effect in multi-display settings. Essentially, this head light causes the lighting to be different on every display which becomes very obvious at the seems.

This can be resolved by simple turning off by calling:

```
viewer->getCamera()->getView()->
setLightingMode(osg::View::NO_LIGHT);
```

Since Vrui sets up a light in OpenGL itself, that lighting will then still be used for the Open Scene Graph rendering.

Vru integration

With Delta3D being based on Open Scene Graph, it can be integrated in Vrui just as easily as Open Scene Graph itself. In fact, the rendering part is exactly identical.

We simply need to add the Delta3D objects that are part of the scene so that those are rendered via Open Scene Graph.

Vrui integration

From there on, we can create a 3D scene, which will serve as the virtual world. This can include buildings, furniture, cars, or whatever you need.

We can also add people (for example using Cal3D). Of course, we can make these people walk around if need be as well.

Similarly, other objects can move around also, such as cars. We can also add a physics engine to make objects behave more realistically by following the laws of physics.

Open Scene Graph

Open Scene Graph is based on OpenGL. It utilizes a tree structure to represent the geometry that is to be rendered. This tree structure follows the principle of hierarchical rendering in that a node can influence the rendering of its sub-tree below it. For example, a node can be a transformation which would then apply to the entire sub-tree.

Each node of this tree is of type osg::Node (or a class derived from that). The provided Vrui sample code already contains a root element for your tree.

Open Scene Graph

Adding geometry

You can then simply add child nodes via the `addChild` method of a node. It takes a new node as its only parameter. So in order to add geometry, you can create a new node and load the geometry using the `readNodeFile` method with the filename as its parameter.

Open Scene Graph supports various file formats, including OpenCollada. New formats can also be added via the plug-in mechanism of Open Scene Graph.

Open Scene Graph

Adding transformations

Similar to adding geometry, transformations can be added to manipulate the position of subsequent geometry. For this you can create a new transformation class `osg:Transform`. Since this is derived from `osg::Node` you can add it to the tree as previously described. If you add more children nodes to this transformation, the transformation will be applied to all children (and the further).

Open Scene Graph

Adding transformations

There are more specific transformations that you can use, such as `the osg::MatrixTransform` and the `osg::PositionAttitudeTransform`. These provide methods to set the transformation accordingly:

```
osg::MatrixTransform::setMatrix (Matrix &mat)
```

```
osg::PositionAttitudeTransform::setPosition (Vec3D &pos);
```

```
osg::PositionAttitudeTransform::setScale (Vec3D &scale);
```

Gaming Environments

Unity

- Programmed in C#
- Support for multitude of devices, mostly head mounted, but also handheld, such as Android and IOS

Unreal

- Programmed in C++