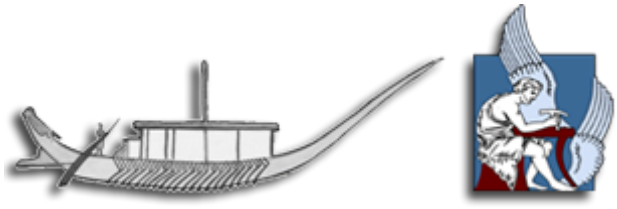
ELECTRONICS AND COMPUTER ENGINEERING DEPARTMENT

TECHNICAL UNIVERSITY OF CRETE

A SYSTEM TO MEASURE, CONTROL AND MINIMIZE END-TO-END HEAD TRACKING LATENCY IN IMMERSIVE SIMULATIONS, INVESTIGATING USER EXPERIENCE

Giorgos Papadakis

10/4/2013



Thesis submitted in fulfillment of the requirements for the degree of Master of Science in Electronics and Computer Engineering.

Department of Electronic and Computer Engineering. Laboratory of Distributed Multimedia Information Systems and Applications MUSIC

CHANIA 2013

Abstract

System latency (time delay) and its visible consequences are fundamental Virtual Environment (VE) deficiencies that can hamper user perception and performance. In order to realize this goal, we present an immersive simulation system which improves upon current latency measurement and minimization techniques. Hardware used for latency measurements and minimization is assembled by low-cost and portable equipment, most of them commonly found in an academic facility without reduction in accuracy of measurements. We present a custom-made mechanism of measuring and minimizing end-to-end head tracking latency in an immersive VE. The mechanism is based on an oscilloscope comparing two signals. One is generated by the head-tracker movement and reported by a shaft encoder attached on a servo motor moving the tracker. The other is generated by the visual consequences of this movement in the VE and reported by a photodiode attached to the computer monitor. Visualization and application-level control of latency in the VE was implemented using the XVR platform. To achieve interoperability between the head-tracker orientation data API and the VE application an intermediate API was developed. Minimization processes resulted in almost 50% reduction of initial measured latency. The description of the mechanism by which VE latency is measured and minimized will be essential to guide system countermeasures such as predictive compensation. The system presented in this thesis will be used to investigate the effect of latency on spatial awareness states.

This thesis also presents an experimental methodology exploring the effect of tracking latency on object recognition after exposure to an immersive VE, in terms of both scene context and associated awareness states. The immersive simulation consisted of a radiosity-rendered space divided in three zones including a kitchen/dining area, an office area and a lounge area. The space was populated by objects consistent as well as inconsistent with each zone's context. The simulation was displayed on a stereo head-tracked Head Mounted Display. Participants across three conditions of varying latency (system minimum latency vs. typical VE latency vs. extensive latency) were exposed to the VE. The same API developed for tracker

reading by the VE application was used from latency addition to the readings. Following the exposure participants were asked to complete an object-based memory recognition task. Participants also reported one of two states of awareness following each recognition response which reflected either the recollection of contextual detail, or the sense of familiarity.

Declaration

The work in this thesis is original and no portion of the work referred to here has been submitted in support of an application for another degree or qualification of this or any other university or institution of learning.

Signed: Date:

Giorgos Papadakis

Acknowledgments

Αρχικά, θα ήθελα να εκφράσω τις ιδιαίτερες ευχαριστίες μου στην κ. Κατερίνα Μανιά, την επιβλέπουσα καθηγήτρια μου, που με τις γνώσεις της αλλά πάνω από όλα με πάντα ευχάριστη διάθεση και με πολύ υπομονή μου προσέφερε ανεκτίμητη βοήθεια σε αυτή την εργασία.

Οφείλω επίσης ένα ιδιαίτερο ευχαριστώ στο καθηγητή κ. Ε. Κουτρούλη του οποίου η βοήθεια ήταν καθοριστική στο να μπορέσω να ξεπεράσω τις πολλές δυσκολίες που είχα με τον σχεδιασμό και την υλοποίηση του συστήματος και στον καθηγητή ψυχολογίας κ. Μ. Coxxon για τη βοήθειά που μου προσέφερε με τα ερωτηματολόγια των πειραμάτων.

Ένα μεγάλο ευχαριστώ στα μέλη του εργαστηρίου στην Βαγγελιώ την Κατερίνα και τον Γιώργο για την πολύτιμη βοήθεια που μου προσέφεραν στην διεξαγωγή των πειραμάτων, βοηθώντας πολλές ώρες στο γραφείο της υποδοχής αλλά και στα παλαιότερα μέλη τον Αλέξανδρο και τον Γιάννη με τους οποίους περάσαμε πολλές ευχάριστες ώρες στον εργαστήριο έχοντας άριστη συνεργασία και κατά τη διάρκεια των μεταπτυχιακών μας σπουδών.

Ευχαριστώ επίσης το Εργαστήριο Διανεμημένων Πληροφοριακών Συστημάτων και Εφαρμογών, το Τμήμα Ηλεκτρονικών Μηχανικών και Μηχανικών Η/Υ και το Πολυτεχνείο Κρήτης τα οποία μου παρείχαν πολύτιμη χρηματοδότηση και υλικοτεχνικό εξοπλισμό.

Πάνω από όλα όμως θα ήθελα να ευχαριστήσω τους γονείς μου και την αδερφή μου, για, την πίστη τους σε μένα, και την υποστήριξή τους όλο το διάστημα των σπουδών μου και τους αφιερώνω αυτή την εργασία. Τέλος, θα ήθελα να ευχαριστήσω τους φίλους μου, για την ηθική και ψυχολογική στήριξη από την αρχή μέχρι και το τέλος αυτής της εργασίας.

Στους γονείς και την αδερφή μου

It is never too late to be what you might have been.

George Eliot

Ποτέ δεν είναι αργά να γίνεις αυτό που θα μπορούσες να έχεις γίνει

George Eliot

Publications

2011 Papadakis, G., Mania, K., Coxon, M., Koutroulis E. "The effect of tracking delay on awareness states in immersive virtual environments: an initial exploration." *ACM SIGGRAPH VRCAI '11.* Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry Pages 475-482

2011 Papadakis, G., Mania, K., Koutroulis, E. A System to Measure Control and Minimize End-to-End Tracking Latency in Immersive Simulations. *ACM SIGGRAPH VRCAI '11.* Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry Pages 581-584

2009 Papadakis, G., Mania, K. "The Cognitive Impact of Head Tracking Latency in Immersive Simulations." *ACM SIGGRAPH Symposium on Applied Perception in Graphics and Visualization 2009*. Pages 136-136.

Table of Contents

List of Tables

List of Figures

CHAPTER 1. Introduction

The work presented in this thesis aims to implement an innovative latency measurement mechanism as well as an immersive simulation of minimum latency.

End-to-end latency in a Virtual Environment (VE) is defined as the time lag between a user's action in the VE and the system's response to this action. VE latency comprises of four different types; user-input device lag, application-dependent processing lag, rendering lag and synchronization lag. The user input device lag is the lag introduced from the communication between the tracking system and the VE application. The application- dependent processing lag is the time required for the computation of the 3D model and depends on the complexity of the model and the application itself. The rendering lag is the time that passes until data sent from the VE application to the rendering hardware appears on a monitor or immersive display. The rendering lag depends on the scene and the viewpoint rendered at each time, so it varies through the VE application run-time.

The synchronization lag is the total time that data is waiting during the necessary communication of involved input devices, in- between the parallel processing stages of the VE application. It is application-relevant and depends on rendering processing stages which may not be well-synchronized to avoid delays of transmission. These stages are independent and it is possible that the input device deposits new tracking data shortly after the application reads the previous data. Thus, the application is busy processing the previous input before it reads and starts to process the new input, so that input data is delayed. Moreover, there is a fifth kind of lag, i.e. the frame-rate-induced lag, resulting from the fact that data displayed progressively become out of date, while the display is not updated fast. This type of lag is distinguished from other lag sources and is not considered as end-to-end latency. It is, though, perceivable by the users exposed to a VE and is considered unacceptable because of the resulting slow frame rates (Wloka 1995). Excessive system latency is a well-known defect of VE and teleoperation systems (Ellis, Mania, et al. 2004). It is particularly troublesome for head-tracked systems since delays in head orientation

measurement give rise to errors in presented visual direction (Stanney, Mourant and Kennedy 1998), (Mania, Adelstein, et al. 2004). The work presented in this thesis aims to implement an immersive simulation of minimum latency as well as a latency measurement mechanism. We extend past latency measurement and minimization techniques to build a port- able, low-cost, custom-made but also accurate latency measurement system and, ultimately, create a VE with minimal head tracking latency.

The term Virtual Environment (VE) is generally understood to mean an environment that is described in three dimensions be presented on a computer display. They are perhaps most commonly encountered in computer games, but are also found in research, simulation, training and design — particularly architectural design. The term Immersive Virtual Environment (IVE) in this thesis is referred to VEs that are displayed using equipment that produces an ego-centric view, allowing the view position and direction to be changed by moving the head and body in a natural way (Sutherland 1965). Today, this may be achieved through the use of an at least three degrees-of-freedom spatial tracker and a stereoscopic head-mounted display (HMD). When using an IVE one can attain a sense that one is actually present within the virtual environment that is displayed, and 'presence' has in fact been identified as a key feature for their general use (Held and Durlach 1992), or even their defining factor in terms of the human experience (Steuer 1992).

1.1. Main Contributions

- The development of a novel low-cost, custom-made portable latency measuring mechanism extending past latency measurement techniques.
- The development of a simulation system with minimum end-to-end latency to be controlled for a variety of head and hand tracking devices relevant to navigation of 3D environments and teleportation.
- An experiment exploring whether the cognitive impact of latency is severe for spatial awareness by investigating the effect of latency on 3D spatial cognition, spatial awareness states and 3D mental models and imagery.

1.2. Thesis Outline

This thesis is divided into a number of chapters. A brief summary of each chapter is presented here.

CHAPTER 2: Technical Background

This chapter introduces a set of fundamental terms in computer graphics starting with defining light and its properties, light energy, photometry and radiometry and continuing with computer graphics illumination models analysis. Furthermore, this chapter illustrates the complex variety of tools and equipment required to create, view and interact with immersive VEs. It provides background information regarding the key technologies used in order to implement the experimental framework and experimental protocol put forward and an overview of the technologies necessary to display and interact with immersive VEs.

Subsequently, latency and its effects on users of Virtual Environments are analyzed and methods for measurement and minimization of latency in Virtual Environments are presented. The last section of this chapter is focused on memory awareness states and schemata.

CHAPTER 3: The Virtual Environment Application

In this chapter, the technical requirements of the head-tracked stereoscopic 3D interactive system are introduced. The architecture of the application developed for the experiments is presented, along with the inherent architecture of the XVR Environment used to develop it.

CHAPTER 4: Hardware and Software setup for Latency Measurements and Minimization

Chapter 4 describes the hardware and software setup used for measuring the end-to-end head tracking latency of our VE system. In this chapter we also present the data collected from end-to end latency measurements. We describe techniques

we used for controlling and minimizing latency and we compare data before and after the application of these techniques.

CHAPTER 5: Latency Experiment

This chapter is concerned with the experimental methods employed when the actual experiments were conducted measuring the effect of latency in immersive simulations. The experimental procedure is presented, as well as the results of the experiments.

CHAPTER 6: Conclusion

Finally, the results and contributions of this thesis are presented. Future work, unveiled by relevant conclusions is suggested.

CHAPTER 2. Technical Background

2.1. Computer Graphics Rendering

2.1.1. The physical Behavior of Light

Light is one form of electromagnetic radiation, a mode of propagation of energy through space that includes radio waves, radiant heat, gamma rays and X-rays. One way in which the nature of electromagnetic radiation can be pictured is as a pattern of waves propagated through an imaginary medium. Primary properties of light are intensity, propagation direction, frequency or wavelength spectrum, and polarization, while its speed in a vacuum, 299,792,458 meters per second, is one of the fundamental constants of nature. The term "visible light" is used to describe the subset of the spectrum of electromagnetic energy to which the human eye is sensitive. This subset, usually referred to as the visual range or the visual band, consists of electromagnetic energy with wavelengths in the range of 380 to 780 nanometers, although the human eye has very low sensitivity to a wider range of wavelengths, including the infrared and ultraviolet ranges. The range of visible light

is shown in Figure 1. As shown, the wavelength at which the human eye is most sensitive is 555 nm.

In the field of computer graphics three types of light interaction are primarily considered: *absorption*, *reflection* and *transmission*. In the case of absorption, an incident photon is removed from the simulation with no further contribution to the illumination within the environment. Reflection considers incident light that is propagated from a surface back into the scene and transmission describes light that travels through the material upon which it is incident and can then return to the environment, often from another surface of the same physical object. Both reflection and transmission can be subdivided into three main types:

- **Specular:** When the incident light is propagated without scattering as if reflected from a mirror or transmitted through glass.
- **Diffuse:** When incident light is scattered in all directions.
- ***Glossy:*** This is a weighted combination of diffuse and specular.

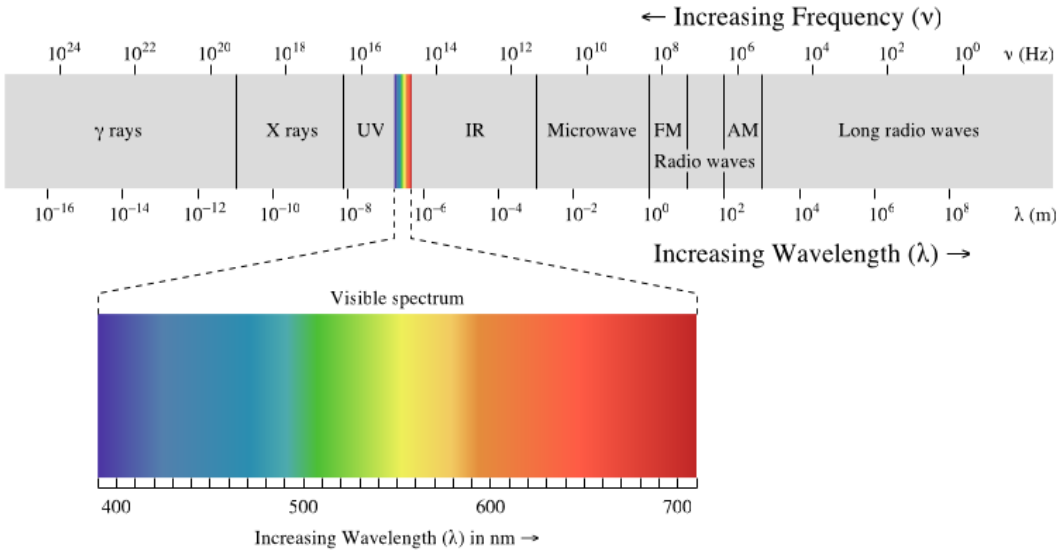


Figure 1: The visible portion of the electromagnetic spectrum.

Most materials do not fall exactly into one of the material categories described above but instead exhibit a combination of specular and diffuse characteristics.

In order to create shaded images of three dimensional objects, we should analyze in detail how the light energy interacts with a surface. Such processes may include *emission*, *transmission*, *absorption*, *refraction*, *interference* and *reflection* of light (Palmer 1999).

Emission

Emission is when light is emitted from an object or surface, for example the sun or man-made sources, such as candles or light bulbs. Emitted light is composed of photons generated by the matter emitting the light; it is therefore an intrinsic source of light.

Transmission

Transmission describes a particular frequency of light that travels through a material returning into the environment unchanged as shown in Figure 2. As a result, the material will be transparent to that frequency of light. Most materials are transparent to some frequencies, but not to others. For example, high frequency light rays, such as gamma rays and X-rays, will pass through ordinary glass, but the lower frequencies of ultraviolet and infrared light will not.

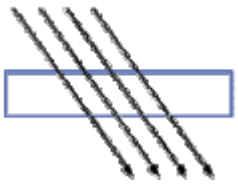


Figure 2: Light transmitted through a material

Absorption

Absorption describes light as it passes through matter resulting in a decrease in its intensity as shown in Figure 3, i.e. some of the light has been absorbed by the object. An incident photon can be completely removed from the simulation with no further contribution to the illumination within the environment if the absorption is great enough.

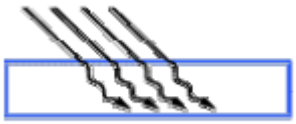


Figure 3: Light absorbed by a material

Refraction

Refraction describes the bending of a light ray when it crosses the boundary between two different materials as shown in Figure 4. This change in direction is due to a change in speed. Light travels fastest in empty space and slows down upon entering matter. The refractive index of a substance is the ratio of the speed of light in space (or in air) to its speed in the substance. This ratio is always greater than one.

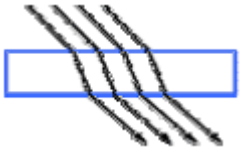


Figure 4: Light refracted through a material

Interference

Interference is an effect that occurs when two waves of equal frequency are superimposed. This often happens when light rays from a single source travel by different paths to the same point. If, at the point of meeting, the two waves are in phase (the crest of one coincides with the crest of the other), they will combine to form a new wave of the same frequency. However the amplitude of this new wave is the sum of the amplitudes of the original waves. The process of forming this new wave is called constructive interference (Flavios n.d.). If the two waves meet out of phase (a crest of one wave coincides with a trough of the other), the result is a wave whose amplitude is the difference of the original amplitudes. This process is called destructive interference (Flavios n.d.). If the original waves have equal amplitudes,

they may completely destroy each other, leaving no wave at all. Constructive interference results in a bright spot; destructive interference produces a dark spot.

- ***Reflection*** considers incident light that is propagated from a surface back into the scene. Reflection depends on the smoothness of the material's surface relative to the wavelength of the radiation (Physics of light And colour n.d.). A rough surface will affect both the relative direction and the phase coherency of the reflected wave. Thus, this characteristic determines both the amount of radiation that is reflected back to the first medium and the purity of the information that is preserved in the reflected wave. A reflected wave that maintains the geometrical organization of the incident radiation and produces a mirror image of the wave is called a specular reflection, as can be seen in Figure 5.

Figure 5: Light reflected off a material in different ways, from left to right, specular, diffuse, mixed, retro-reflection and finally gloss (Katedros n.d.)

2.1.2. Computer Graphics illumination

An illumination model computes the color at a point in terms of light directly emitted by the light source(s) (Foley, et al. 1990). A local illumination model calculates the distribution of light that comes directly from the light source(s). A global illumination model additionally calculates reflected light from all the surfaces in a scene which could receive light indirectly via intereflections from other surfaces. Global illumination models include, therefore, all the light interaction in a scene, allowing for soft shadows and color bleeding that contribute towards a more photorealistic image. The rendering equation expresses the light being transferred

from one point to another (Kajiya 1986). Most illumination computations are approximate solutions of the rendering equation:

$$I(x,y) = g(x,y)[\varepsilon(x,y) + \int_S p(x,y,z)I(y,z)\,dz]$$

Where;

- x, y, z are points in the environment,
- $I(x,y)$ is related to the intensity passing from y to x,
- $g(x,y)$ is a 'geometry' term that is 0 when x, y are occluded from each other and 1 otherwise,
- $p(x,y,z)$ is related to the intensity of light reflected from z to x from the surface at y, the integral is over all points on all surfaces S,
- $\varepsilon(x,y)$ is related to the intensity of light that is emitted from y to x.

Thus, the rendering equation states that the light from y that reaches x consists of light emitted by y itself and light scattered by y to x from all other surfaces which themselves emit light and recursively scatter light from other surfaces. The distinction between view-dependent rendering algorithms and view-independent algorithms is a significant one. View-dependent algorithms discretize the view plane to determine points at which to evaluate the illumination equation, given the viewer's direction, such as ray-tracing (Dingliana n.d.), (A. S. Glassner, Principles of Digital Image Synthesis 1995). View-independent algorithms discretize the environment and process it in order to provide enough information to evaluate the illumination equation at any point and from any viewing direction, such as Radiosity.

Bouknight (1970) introduced one of the first models for local illumination of a surface. This included two terms, a diffuse term and an ambient term. The diffuse term is based upon the Lambertian reflection model, which makes the value of the outgoing intensity equal in every direction and proportional to the cosine of the angle between the incoming light and the surface normal. The ambient term is constant and approximates diffuse inter-object reflection. Gouraud (1971) extended

this model to calculate the shading across a curved surface approximated by a polygonal mesh. His method calculated the outgoing intensities at the polygon vertices and then interpolated these values across the polygon as shown in Figure 6(middle).

Prong (1975) introduced a more sophisticated interpolation scheme where the surface normal is interpolated across a polygon and the shading calculation is performed at every visible point, as shown in Figure 6 (right). He also introduced a specular term. Specular reflection is when the reflection is stronger in one viewing direction, i.e. there is a bright spot called a specular highlight. This is readily apparent on shiny surfaces. For an ideal reflector, such as a mirror, the angle of incidence equals the angle of specular reflection. Although this model is not physically based, its simplicity and efficiency make it still the most commonly used local reflection model.

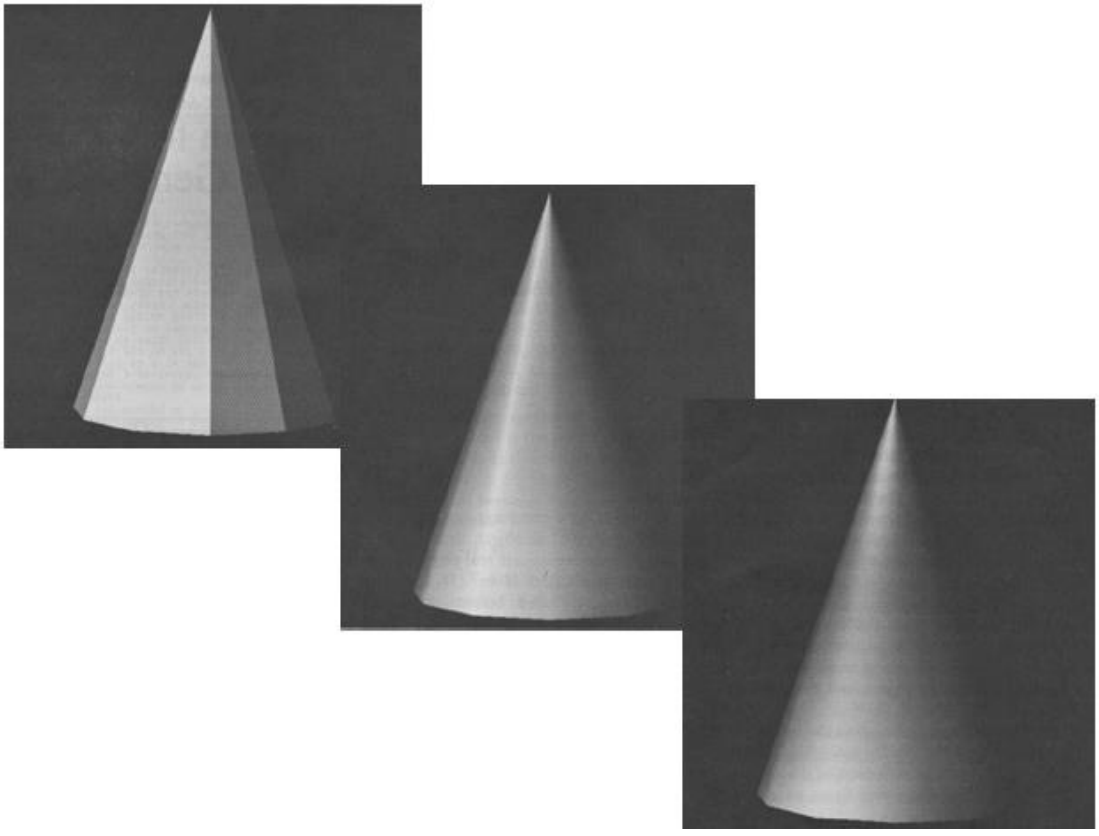


Figure 6: The differences between a simple computer-generated polyhedral cone (left), with linearly interpolated shading to give appearance of curvature (Gouraud Shading). Note Mach bands

at edges of faces (middle) and a more complex shading calculation, interpolating curved surface normals (Phong Shading). This is necessary to eliminate Mach Bands (right).

A global illumination model adds to the local illumination model, the light that is reflected from other non-light surfaces to the current surface. A global illumination model is physically correct and produces realistic images resulting in effects such as color bleeding and soft shadows. When measured data is used for the geometry and surface properties of objects in a scene, the image produced should then be theoretically indistinguishable from reality. However, global illumination algorithms are also more computationally expensive.

Global illumination algorithms produce solutions of the rendering equation proposed by (Kajiya 1986):

$$L_{out} = L_E + L_{in} \int r \cos\theta \, d\theta$$

where L_{out} is the radiance leaving a surface, L_E is the radiance emitted by the surface, L_{in} is the radiance of an incoming light ray arriving at the surface from light sources and other surfaces, r is the bi-directional reflection distribution function of the surface, θ is the angle between the surface normal and the incoming light ray and $d\theta$ is the differential solid angle around the incoming light ray.

The rendering equation is graphically depicted in Figure 7. In this figure L_{in} is an example of a direct light source, such as the sun or a light bulb, L_{in} is an example of an indirect light source i.e. light that is being reflected off another surface, R, to surface S. The light seen by the eye (L_{out}) is simply the integral of the indirect and direct light sources modulated by the reflectance function of the surface over the hemisphere Ω.

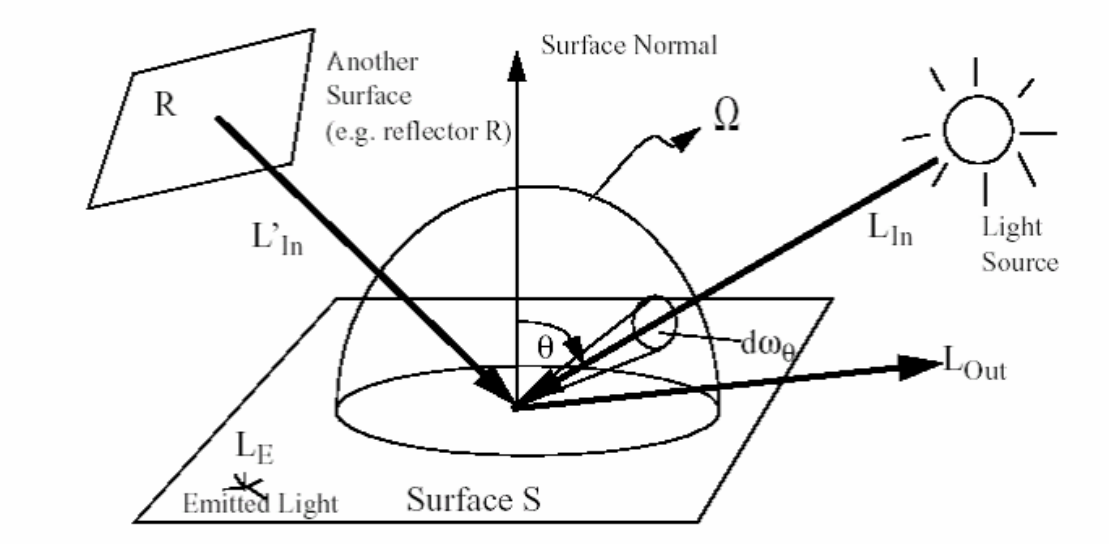


Figure 7: Graphical depiction of the rendering equation (Yee, Pattanaik and Greenberg 2001)

The problem of global illumination can be seen when you have to solve the rendering equation for each and every point in the environment. In all but the simplest case, there is no closed form solution for such an equation so it must be solved using numerical techniques and therefore this implies that there can only be an approximation of the solution (Lischinski n.d.). For this reason most global illumination computations are approximate solutions to the rendering equation.

The two major types of graphics systems that use the global illumination model are radiosity and ray tracing. In this work, we utilized radiosity rendering; however, we will refer to ray-tracing in order to emphasize algorithmic differences which guided us to our choice of radiosity.

2.1.3. Ray Tracing

Ray tracing is global illumination algorithm which calculates specular reflections (view dependent) and results in a rendered image. Rays of light are traced from the eye through the center of each pixel of the image plane into the scene, these are called primary rays. When each of these rays hits a surface it spawns two child rays, one for the reflected light and one for the refracted light. This process continues recursively for each child ray until no object is hit, or the recursion reaches some specified

maximum depth. Rays are also traced to each light source from the point of intersection. These are called shadow rays and they account for direct illumination of the surface, as shown in Figure 8. If a shadow ray hits an object before intersecting with the light source(s), then the point under consideration is in shadow. Otherwise, there must be clear path from the point of intersection of the primary ray to the light source and thus a local illumination model can be applied to calculate the contribution of the light source(s) to that surface point.

The simple ray tracing method outlined above has several problems. Due to the recursion involved and the possibly large number of rays that may be cast, the procedure is inherently expensive. Diffuse interaction is not modeled, nor is specular interaction, other than that by perfect mirrors and filters. Surfaces receiving no direct illumination appear black. In order to overcome this, an indirect illumination term, referred to as ambient light, is accounted for by a constant ambient term, which is usually assigned an arbitrary value (Glassner 1989). Shadows are hard-edged and the method is very prone to aliasing. The result of ray tracing is a single image rendered for a particular position of the viewing plane, resulting in a view –dependent technique.

In ray tracing each ray must be tested for intersection with every object in the scene. Thus, for a scene of significant complexity, the method rapidly becomes impracticable. Several acceleration techniques have been developed, which may be broadly categorized into two approaches: reducing the number of rays and reducing the number of intersection tests. Hall and Greenberg noted that the intensity of each ray is reduced by each surface it hits, thus the number of rays should be stopped before any unnecessary recursion to a great depth occurs (Hall and Greenberg 1983). Another approach, which attempts to minimize the number of ray object intersections, is spatial subdivision. This method encloses a scene in a cube that is then partitioned into discrete regions, each of which contains a subset of the objects in the scene. Each region may then be recursively subdivided until each sub-region (voxel or cell) contains no more than a preset maximum number of objects.

Several methods for subdividing space exist. (A. S. Glassner, Space Subdivision for Fast Ray Tracing, Vol.4, No. 10 1984) proposes the use of an octree, e.g. a structure where the space is bisected in each dimension, resulting in eight child regions. This subdivision is repeated for each child region until the maximum tree depth is reached, or a region contains less than a certain number of objects. Using such a framework allows for spatial coherence, i.e. the theory that similar objects in a scene affect neighboring pixels. Rays are traced through individual voxels, with intersection tests performed only for the objects contained within, rather than for all the objects in the scene. The ray is then processed through the voxels by determining the entry and exit points for each voxel traversed by the ray until an object is intersected or the scene boundary is reached.

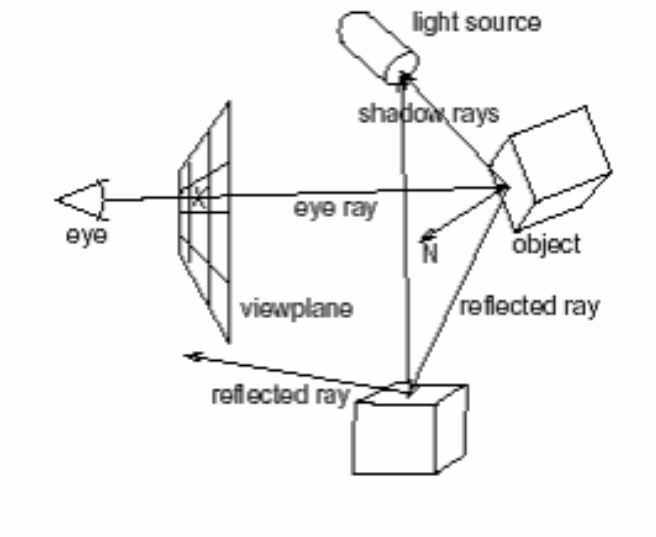


Figure 8: Ray Tracing

Ray tracing was not employed in this work because of the extreme computational demand of this method when real time scenes are produced. In order to achieve interactive frame rates, a view independent rendering framework is needed. Recent research explores methods for real time ray tracing (Mortensen, et al. 2008).

2.1.4. Radiosity

Radiosity is a global illumination algorithm. It is used in 3D computer graphics rendering. It is a solution of the rendering equation for scenes with purely diffuse surfaces using the finite element method. Contrasting Monte Carlo algorithms (i.e. path tracing) which handle all types of light paths, usual radiosity methods only take into account paths which leave a light source and are reflected diffusely a number of times (possibly zero) before hitting the eye. Such paths are represented as "LD*E". Calculations made with radiosity do not depend on the viewpoint. That makes radiosity calculations useful for all viewpoints but also increases computational complexity.

Initially radiosity methods were developed in about 1950 in the engineering field of heat transfer. Later they were refined for application to the problem of rendering computer graphics in 1984 by researchers at Cornell University (Goral, et al. 1984). The heat transfer theory describes radiation as the transfer of energy from a surface when that surface has been thermally excited. This encompasses both surfaces that are basic emitters of energy, as with light sources and surfaces that receive energy from other surfaces and thus have energy to transfer. The thermal radiation theory can be used to describe the transfer of many kinds of energy between surfaces, including light energy.

As in thermal heat transfer, the basic radiosity method for computer image generation makes the assumption that surfaces are diffuse emitters and reflectors of energy, emitting and reflecting energy uniformly over their entire area. Thus, the radiosity of a surface is the rate at which energy leaves that surface (energy per unit time per unit area). This includes the energy emitted by the surface as well as the energy reflected from other surfaces in the scene. Light sources in the scene are treated as objects that have self emittance.

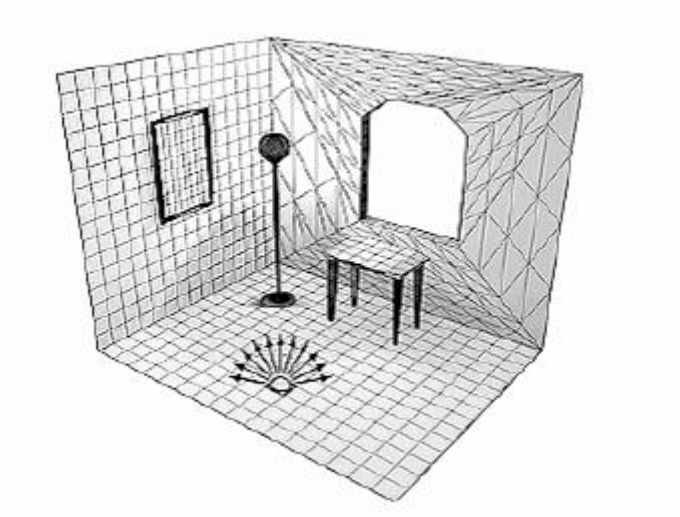


Figure 9: Radiosity patches (McNamara 2000)

The surfaces of the scene to be rendered are each divided up into one or more smaller surfaces (patches) (Figure 9). A form factor is computed for each pair of patches. Form factors are coefficients describing how well the patches can see each other. Each form factor represents the proportion of light leaving one patch (patch i) that will arrive at the other (patch j) (Siegel and Howell 1992). Patches that are far away from each other, or oriented at oblique angles relative to one another, will have smaller form factors. If other patches are in the way, the form factor will be reduced or zero, depending on whether the occlusion is partial or total. Thus the radiosity equation is:

$$B_i = E_i + \rho_i \sum_{j=1}^{n} F_{ij} B_j$$

Where:

B_x= Radiosity of patch x

E_x= Emissivity of patch x

ρ_x= Reflectivity of patch x

F_{xy}= Form factor of patch y relative to patch x

The reciprocity relationship of form factor F_{ij} (Siegel and Howell 1992) states:

$$A_i F_{ij} = A_j F_{ji}$$

Where A_i and A_j are the areas of patch j and i respectively, as shown in Figure 10.

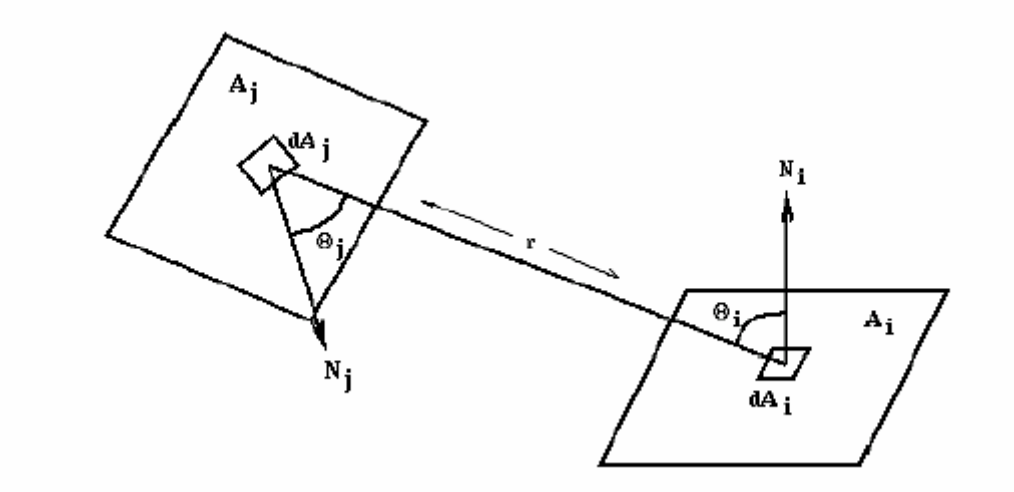


Figure 10: Relationship between two patches (Katedros n.d.)

As the environment is closed, the emittance functions, reflectivity values and form factors form a system of simultaneous equations that can be solved to find the radiosity of each patch. The radiosity is then interpolated across each of the patches and finally the image can then be rendered. The equation is monochromatic, so color radiosity rendering requires calculation for each of the required colors.

The basic form factor equation is difficult even for simple surfaces. The number of calculations to compute the matrix solution scales according to n^3, where n is the number of patches. This becomes prohibitive for realistically large values of n. Nusselt (1928) developed a geometric analog that allows the simple and accurate calculation of the form factor between a surface and a point on a second surface. The Nusselt Analog involves placing a hemispherical projection body, with unit radius,

at a point on the A_i surface. The second surface - A_j - is spherically projected onto the projection body and then cylindrically projected onto the base of the hemisphere. The form factor then may be approximated by the area projected on the base of the hemisphere divided by the area of the base of the hemisphere, as shown in Figure 11.

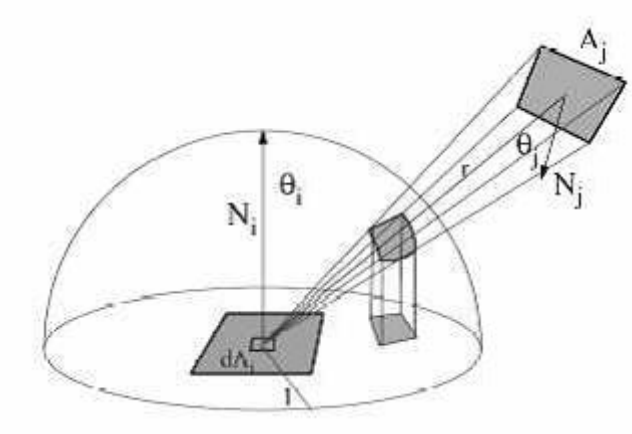


Figure 11: Nusselt's analog. The form factor from the differential area dA_i to element A_j is proportional to the area of the double projection onto the base of the hemisphere (Nusselt 1928).

Cohen and Greenberg (1985) proposed that the form factor between each pair of patches could also be calculated by placing a hemi-cube on each patch and projecting the environment on to it as defined by the Nusselt Analog. Each face of the hemicube is subdivided into a set of small, usually square ("discrete") areas, each of which has a precomputed delta form factor value, as shown in Figure 12. When a surface is projected onto the hemicube, the sum of the delta form factor values of the discrete areas of the hemicube faces which are covered by the projection of the surface is the form factor between the point on the first surface (about which the cube is placed) and the second surface (the one which was projected). The speed and accuracy of this method of form factor calculation can be affected by changing the size and number of discrete areas on the faces of the hemicube.

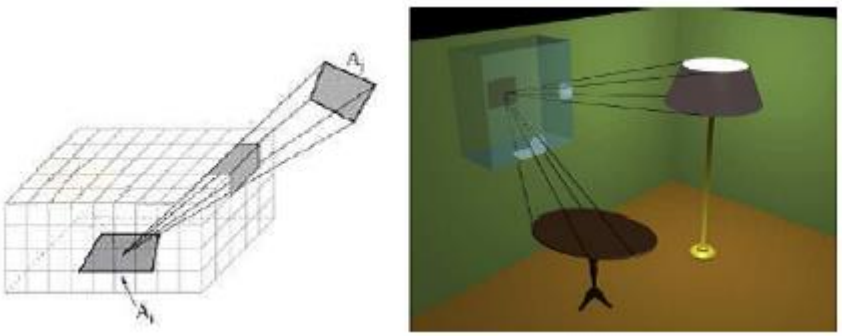


Figure 12: The hemicube (Langbein n.d.)

Radiosity assumes that an equilibrium solution can be reached; that all of the energy in an environment is accounted for, through absorption and reflection. It should be noted that because of the assumption of only perfectly diffuse surfaces, the basic radiosity method is viewpoint independent, i.e. the solution will be the same regardless of the viewpoint of the image. The diffuse transfer of light energy between surfaces is unaffected by the position of the camera. This means that as long as the relative position of all objects and light sources remains unchanged, the radiosity values need not be recomputed for each frame. This has made the radiosity method particularly popular in architectural simulation, targeting high-quality walkthroughs of static environments. Figure 13 demonstrates the difference in image quality that can be achieved with radiosity compared to ray tracing.

However, there are several problems with using the hemicube radiosity method. It can only model diffuse reflection in a closed environment; it is limited to polygonal environments; it is prone to aliasing and has excessive time and memory requirements. Also, only after all the radiosities have been computed in the scene is the resultant image displayed. There is a form factor between each pair of patches, so in an environment with N patches, N2 form factors must be stored. For a scene of moderate complexity this will require a vast amount of storage and as the form factor calculation is non-trivial the time taken to produce a solution can be extensive. This means that the user is unable to alter any of the parameters of the environment until

the entire computation is complete. Then once the alteration is made, the user must once again wait until the full solution is recomputed.

The visual quality of the rendered images in radiosity also strongly depends on the method employed for discretizing the scene into patches. A too fine discretization may give rise to artifacts, while with a coarse discretization, areas with high radiosity gradients may appear (Gibson and Hubbold 1997).To overcome these problems, the discretization should adapt to the scene. That is, the interaction between two patches should account for the distance between them as well as their surface area. In other words, surfaces that are far away are discretized less finely than surfaces that are nearby. These aspects are considered by the adaptive discretization method proposed by (Languénou , Bouatouch and Tellier 1992). It performs both discretization and system resolution at each iteration of the shooting process, which allows for interactivity. (Gibson and Hubbold 1997) demonstrated another solution for this problem by presenting an oracle that stops patch refinement once the difference between successive levels of elements becomes perceptually unnoticeable.

Progressive refinement radiosity (Cohen, Chen, et al. 2004) works by not attempting to solve the entire system simultaneously. Instead, the method proceeds in a number of passes and the result converges towards the correct solution. At each pass, the patch with the greatest unshot radiosity is selected and this energy is propagated to all other patches in the environment. This is repeated until the total unshot radiosity falls below some threshold. Progressive refinement radiosity generally yields a good approximation to the full solution in far less time and with lesser storage requirements, as the form factors do not all need to be stored throughout. Many other extensions to radiosity have been developed and a very comprehensive bibliography of these techniques can be found in (Ashdown 2004).

Figure 13: Differences between ray tracing (middle) and radiosity (right hand image).

Figure 14: Difference between standard direct illumination and radiosity

For the creation of the 3d scene for the VE application which was utilized for this thesis and is described in 0 we used the radiosity rendering.

2.1.5. Texture mapping

Texture mapping is a technique used in order to add detail, surface texture or to colorize a 3d model or graphic generated by a computer. Texture mapping is used for creating 3d objects for objects, avatars, rooms for virtual worlds. The size of texture map varies. It is recommended that pixel dimensions are a combination from powers of 2 (i.e. 32, 64, 128, 256 and 512 etc.)

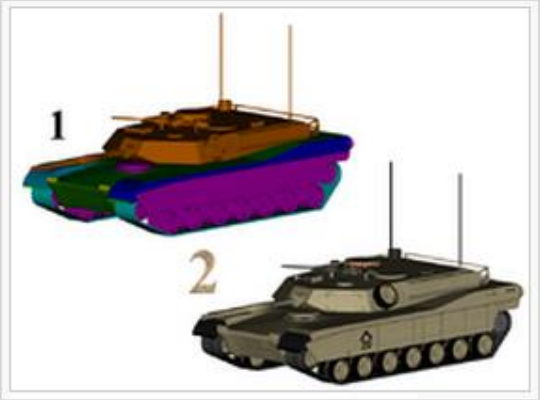


Figure 15: 1) 3D model without textures, 2) 3D model with textures.

A texture map is applied (mapped) to the surface of a shape or polygon (Radoff 2008). This process is analogous to applying patterned paper to a plain white box. Every vertex in a polygon is assigned a texture coordinate (which in the 2d case is also known as a UV coordinate) either via explicit assignment or by procedural definition. Image sampling locations are then interpolated across the face of a polygon to produce a visual result that seems to have more richness than could otherwise be achieved with a limited number of polygons. Multitexturing is the use of more than one texture at a time on a polygon. For instance, a light map texture may be used to light a surface as an alternative to recalculating that lighting every time the surface is rendered. Another multitexture technique is bump mapping, which allows a texture to directly control the facing direction of a surface for the purposes of its lighting calculations; it can give a very good appearance of a complex surface, such as tree bark or rough concrete that takes on lighting detail in addition to the usual detailed coloring. Bump mapping has become popular in recent video games as graphics hardware has become powerful enough to accommodate it in real-time.

Texture filtering directs the way the resulting pixels on the screen are calculated from the texels (texture pixels) . The nearest-neighbor interpolation is the fastest method is to use, but bilinear interpolation or trilinear interpolations between mipmaps are two commonly used alternatives which

reduce aliasing or jaggies. In the event of a texture coordinate being outside the texture, it is either clamped or wrapped.

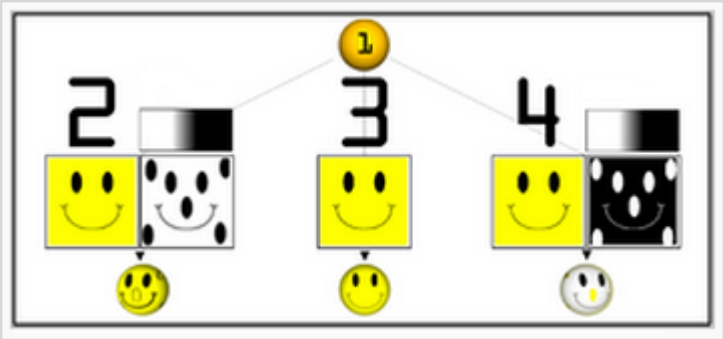


Figure 16: Examples of multitexturing.1) Untextured sphere, 2) Texture and bump maps, 3) Texture map only, 4) Opacity and texture maps.

The essential map types are described below:

Color (or Diffuse) Maps

As the name would imply, the first and most obvious use for a texture map is to add color or texture to the surface of a model. This could be as simple as applying a wood grain texture to a table surface, or as complex as a color map for an entire game character (including armor and accessories). However, the term texture map, as it's often used is a bit of a misnomer—surface maps play a huge role in computer graphics beyond just color and texture. In a production setting, a character or environment's color map is usually just one of three maps that will be used for almost every single 3D model.

Specular Map

Also known as; gloss map. A specular map tells the software which parts of a model should be shiny or glossy, and also the magnitude of the glossiness. Specular maps are named for the fact that shiny surfaces, like metals, ceramics, and some plastics show a strong specular highlight (a direct reflection from a strong light source). Specular highlights are the white reflection on the rim of a coffee mug. Another common example of specular reflection is the tiny white glimmer in

someone's eye, just above the pupil.

A specular map is typically a grayscale image, and is absolutely essential for surfaces that aren't uniformly glossy. An armored vehicle, for example, requires a specular map in order for scratches, dents, and imperfections in the armor to come across convincingly. Similarly, a game character made of multiple materials would need a specular map to convey the different levels of glossiness between the character's skin, metal belt buckle, and clothing material.

Bump, Displacement, or Normal Map

A bit more complex than either of the two previous examples, bump maps are a form of texture map that can help give a more realistic indication of bumps or depressions on the surface of a model.

To increase the impression of realism, a bump or normal map would be added to more accurately recreate the coarse, grainy surface of, for instance, a brick, and heighten the illusion that the cracks between bricks are actually receding in space. Of course, it would be possible to achieve the same effect by modeling each and every brick by hand, but a normal mapped plane is much more computationally efficient. Normal mapping is a significant process incorporated in the development of modern computer games.

Bump, displacement, and normal maps are a discussion in their own right, and are absolutely essential for achieving photo-realism in a render.

Figure 17: Taurus pt. 92 textured 3d model. Rendered in marmoset engine (real time game engine).

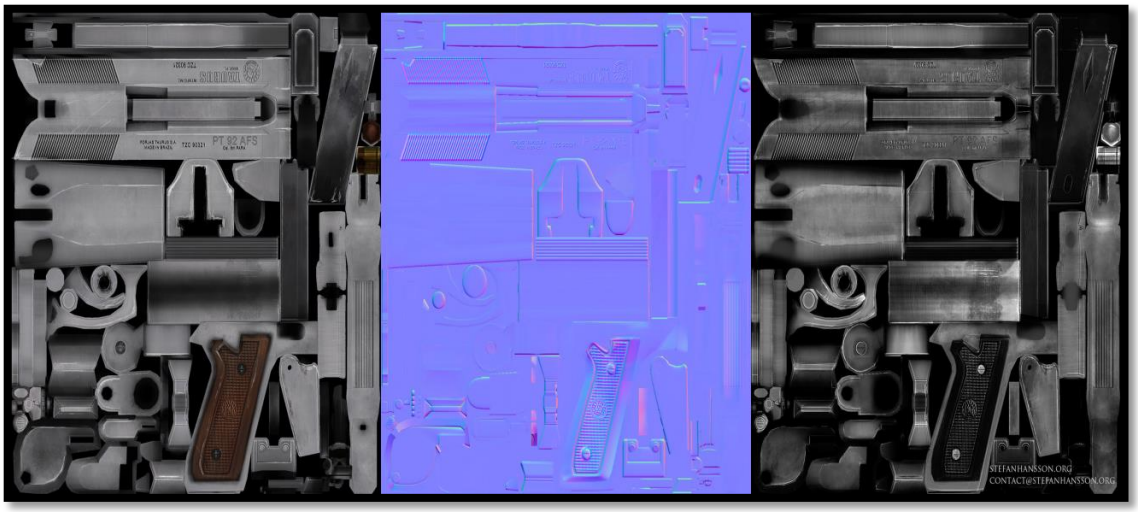


Figure 18: Diffuse, normal and specular map of the above 3d model.

Aside from these three map types, there are one or two others you'll see relatively often:

Reflection Map

It the software which portions of the 3D model should be reflective. If a model's entire surface is reflective or if the level of reflectivity is uniform a reflection map is usually omitted. Reflection maps are grayscale images, with black indicating 0% reflectivity and pure white indicating a 100% reflective surface.

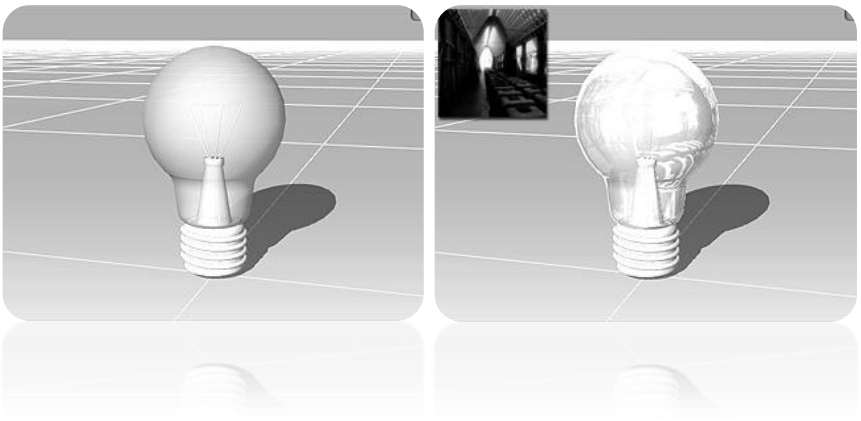


Figure 19: Mesh without any texture (left image). Reflection image projected onto the object (right image).

Transparency (or Opacity) Map

Exactly like a reflection map, except it tells the software which portions of the model should be transparent. A common use for a transparency map would be a surface that would otherwise be very difficult, or too computationally expensive to duplicate, like a chain link fence. Using a transparency, instead of modeling the links individually can be quite convincing as long as the model doesn't feature too close to the foreground, and uses far fewer polygons.

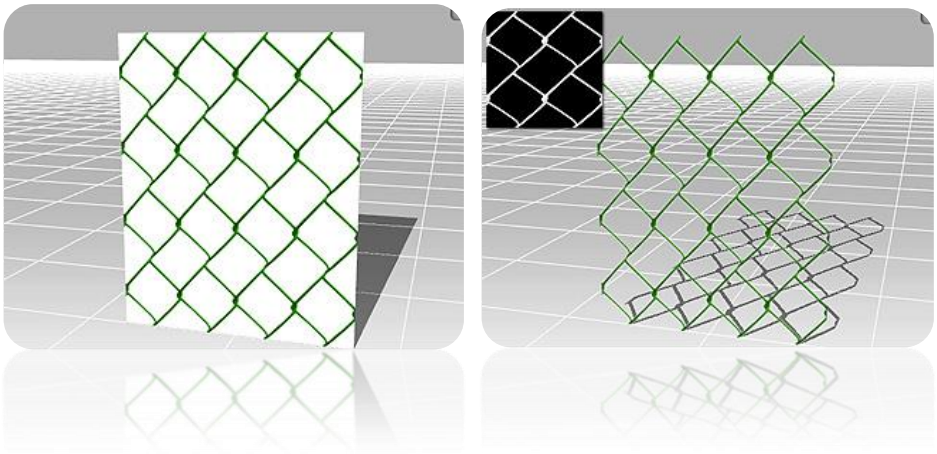


Figure 20: Mesh with diffuse map only (left image). Opacity texture applied on the mesh (right image).

Texture maps are crucial for the design of the virtual scene. They facilitate the reduction of the polygonal complexity of the 3d models used, which in any other way would hinder rendering performance. In particular, the normal maps of some low polygon count models used in our scene were acquired from their high polygon versions to keep all the fine details of the models, thus maintaining an acceptable lighting quality with low computational cost. An example of this method can be seen on the following picture. Another use of texture maps, are opacity maps which allow for the otherwise opaque window in our scene to become transparent.

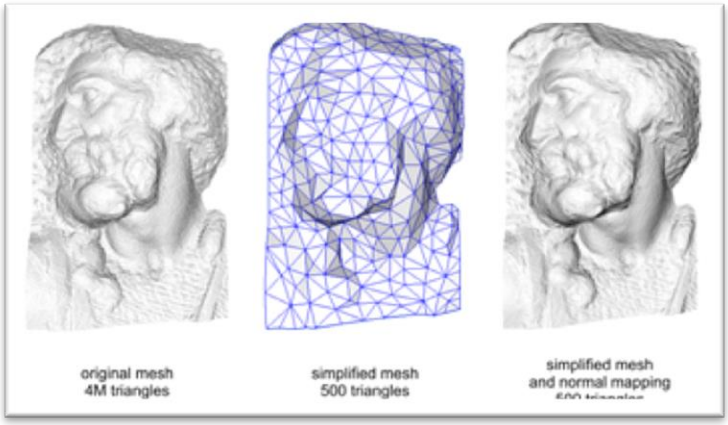


Figure 21: Normal mapping used to re-detail simplified meshes.

UVW mapping

UVW mapping is a mathematical technique for coordinate mapping. In computer graphics, it is most commonly a R^2 to R^3 map, suitable for converting a 2D image (a

texture) to a three dimensional object of a given topology. "UVW", like the standard Cartesian coordinate system, has three dimensions; the third dimension allows texture maps to wrap in complex ways onto irregular surfaces. Each point in a UVW map corresponds to a point on the surface of the object. The graphic designer or programmer generates the specific mathematical function to implement the map, so that points on the texture are assigned to (XYZ) points on the target surface. Generally speaking, the more orderly the unwrapped polygons are, the easier it is for the texture artist to paint features onto the texture. Once the texture is finished, all that has to be done is to wrap the UVW map back onto the object, projecting the texture in a way that is far more flexible and advanced, preventing graphic artifacts that accompany more simplistic texture mappings such as planar projection. For this reason, UVW mapping is commonly used to texture map non-platonic solids, non-geometric primitives, and other irregularly-shaped objects, such as characters and furniture.

UVW mapping was used for mapping the illuminated textures for the scene created for the VE application utilized in this thesis. Details about creation of the scene geometry, creation of the illuminated textures and the mapping of textures to the geometry are discussed in 0.

2.1.6. OpenGL

OpenGL (Open Graphics Library) is a standard specification developed by Silicon Graphics Inc. (SGI) in 1992 and managed by the non-profit technology consortium Khronos Group; defining a cross-language, cross-platform API used in authoring 2D and 3D computer graphics applications. It is widely used in CAD, virtual reality, scientific visualization, information visualization, flight simulation, and video games. The interface consists of over 250 different function calls which can be used to draw from simple primitives to complex three-dimensional scenes.

Two main purposes served by OpenGL are:

- Presenting a single, uniform interface to hide complexities of interfacing with different 3D accelerators
- Requiring support of the full OpenGL feature set for any implementation (with software emulation if needed) to hide differing capabilities of hardware platforms.

OpenGL's basic operation is converting primitives such as points, lines and polygons, into pixels using a graphics pipeline known as the OpenGL state machine. Most OpenGL commands either issue primitives to the graphics pipeline, or configure how these primitives are processed by the pipeline. Prior to OpenGL 2.0, each stage of the pipeline performed a static function and was configurable only within tight limits. OpenGL 2.0 offers several fully programmable stages using OpenGL Shading Language (GLSL).

OpenGL is a low-level, procedural API, requiring the dictation of each of the exact steps to render a scene, in contrast with descriptive (aka scene graph or retained mode) APIs, where a programmer only the description of is needed and the library manages the rendering details. OpenGL's low-level design requires good knowledge of the graphics pipeline by the programmer, but also gives a certain amount of freedom to novel rendering algorithms implementation.

OpenGL has historically influenced the development of 3D accelerators, promoting a base level of functionality common now in consumer-level hardware:

- Rasterized points, lines and polygons as basic primitives
- A transform and lighting pipeline
- Texture mapping
- Alpha blending
- Z-buffering

A brief description of the process in the graphics pipeline (Figure 22) could be:

1. Evaluation, of the polynomial functions which define certain inputs, like NURBS (Non-uniform rational basis spline) surfaces, approximating curves and the surface geometry.
2. Transforming and lighting of vertex operations, depending on their material. Also clipping of non-visible parts of the scene in order to reduce the viewing volume.
3. Rasterization or conversion of the components described above into pixels. The polygons are represented by the appropriate color by means of interpolation algorithms.
4. Per-fragment operations, i.e. updating values depending on incoming and previously stored depth values, or color combinations, etc.
5. Insertion of fragments into the frame buffer.

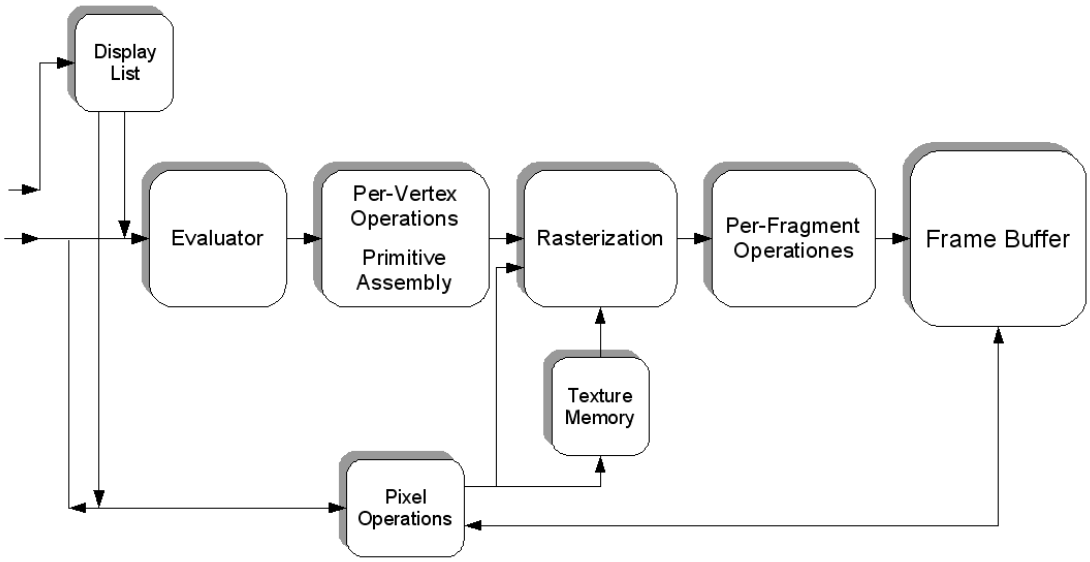


Figure 22: Simplified version of the OpenGL Graphics Pipeline Process

Functionality far above this baseline is provided by many modern 3D accelerators, but these new features are usually not radical revisions of this basic pipeline but enhancements of it.

OpenGL it provides only rendering functions (output-only). The core API has no concept of windowing systems, audio, printing to the screen or input devices, allowing the code that does the rendering to be completely independent of the

operating system it is running on making OpenGL capable for cross-platform development. However, some integration with the native windowing system is required to allow clean interaction with the host system and performed through add-on APIs.

GLUT – The OpenGL Utility Toolkit

The OpenGL Utility Toolkit (GLUT) is a utility library for OpenGL programs. It primarily performs system-level communication with the host operating system. Functions performed include window definition, window control, and monitoring of keyboard and mouse input. Routines for drawing a number of geometric primitives (both in solid and wireframe mode) are also provided, including cubes, spheres, and the Utah teapot. GLUT has some limited support for creating pop-up menus as well.

The two intentions of GLUT are to allow the creation of portable code between different operating systems (GLUT is cross-platform) and to assist learning of OpenGL. OpenGL programming while using GLUT usually takes only a few lines of code and does not require knowledge of operating system–specific windowing APIs.

2.1.7. Direct3D

Direct3D is part of Microsoft's DirectX application programming interface (API). Direct3D is available for Microsoft Windows operating systems (Windows 95 and above), and for other platforms through open source software (e.g. Wine). Direct3D is used to render three dimensional graphics in applications where performance is important, such as games. Direct3D also allows applications to run fullscreen rather than embedded in a window. Direct3D utilizes hardware acceleration if it is available on the graphics card, allowing for hardware acceleration of the entire 3D rendering pipeline or even only partial acceleration. Direct3D exposes the advanced graphics capabilities of 3D graphics hardware, including z-buffering, spatial anti-aliasing, alpha blending, mipmapping, atmospheric effects, and perspective-correct texture mapping. Integration with other DirectX technologies enables Direct3D to deliver

such features as video mapping, hardware 3D rendering in 2D overlay planes, and even sprites, providing the use of 2D and 3D graphics in interactive media titles.

Direct3D contains many commands for 3D rendering; however, since version 8, Direct3D has superseded the old DirectDraw framework and is also responsible for 2D graphics rendering (Microsoft 2013). Microsoft endeavors to constantly update Direct3D to support the latest technology available on 3D graphics cards. Direct3D offers full vertex software emulation but no pixel software emulation for features is available in hardware. For example, if software programmed using Direct3D requires pixel shaders and the video card on the user's computer does not support that feature, Direct3D will not emulate it, although it will compute and render the polygons and textures of the 3D models, although at a usually degraded quality and performance compared to the hardware equivalent. The API includes a Reference Rasterizer (or REF device), which emulates a generic graphics card in software, albeit it is too slow for most real-time 3D applications. It is typically only used for debugging.

2.1.8. Comparison of OpenGL and Dierct3D

Direct3D and OpenGL are two competing application programming interfaces (APIs). They can be used in applications in order to render 2D and 3D computer graphics, utilizing hardware acceleration when available. Modern graphics processing unit (GPUs) may implement a specific version of one or both of Direct3D and OpenGL.

Generally, Direct3D is designed for 3D hardware interfaces virtualization. Direct3D rives freedom to the game programmer from accommodation of the graphics hardware. OpenGL, on the other hand, is designed to be a 3D hardware-accelerated rendering system that can also be emulated in software. There are functional differences in how the two APIs operate. Direct3D expects hardware resources management from the application; OpenGL requires that the implementation does it. This tradeoff for OpenGL decreases difficulty in developing for the API, while at the same time increasing the complexity of creating a high-

performance implementation or driver. With Direct3D, hardware resources must be managed independently by the developer; however, the implementation is simpler, there is the flexibility that the developers allocate resources in the most efficient way possible for their application.

Another functional difference between the APIs was the way they handled rendering to textures until about 2005. The Direct3D method (SetRenderTarget()) is convenient, while prior versions of OpenGL required manipulating pixel buffers (P-buffers). This was cumbersome and risky: if the programmer's codepath was different from that anticipated by the driver maker, the code would have fallen back to software rendering, causing a substantial performance drop. However, widespread support for the "frame buffer objects" extension, which provided an OpenGL equivalent of the Direct3D method, successfully addressed this shortcoming, and the "render target" feature of OpenGL brought OpenGL up to par with Direct3D in this respect.

The two APIs provide nearly the same level of function outside of a few minor functional differences. Hardware and software vendors generally respond rapidly to changes in DirectX, while new features in OpenGL are mainly implemented first by vendors and afterward retroactively applied to the standard.

OpenGL was originally designed for SGI workstations. It includes a number of features, like stereo rendering and the "imaging subset", that were generally considered of limited utility for games - although stereoscopic gaming has drawn a lot more interest as of 2011. The API as a whole contains about 250 calls, but only a subset of perhaps 100 are useful for game development. However, no official gaming-specific subset was ever defined. MiniGL, released by 3Dfx as a stopgap measure to support glQuake, might have served as a starting point, but additional features like stencil were soon adopted by games, and support for the entire OpenGL standard continued. Today, workstations and consumer machines use the same architectures and operating systems, and so modern incarnations of the OpenGL

standard still include these features, although only special workstation-class video cards accelerate them.

One of the most heavily disputed differences between the two APIs is the OpenGL extension mechanism. This mechanism gives the ability to any driver to advertise its own extensions to the API and introducing new functions such as blend modes, new ways to transfer data to GPUs, or different texture wrapping parameters. This allows new functions to be exposed quickly, but can lead to confusion if different vendors implement similar extensions with different APIs. Many of these extensions are periodically standardized by the OpenGL Architecture Review Board (ARB), and some are made a core part of future OpenGL revisions.

OpenGL has always seen more use in the professional production and display of graphics, such as in computer animated films and scientific visualization graphics market than DirectX, while DirectX is used mostly for computer games. Currently both OpenGL and DirectX have a large enough overlap in functionality that either could be used for most common purposes, with the operating system itself often being the primary criterion dictating which is used, with DirectX the common choice on Windows, and OpenGL being used on nearly everything else. Some esoteric applications still divide the applicability of the two APIs: doing accelerated 3D across a network connection is only directly supported by OpenGL with GLX, for example.

Presiously many professional graphics cards only supported OpenGL, now virtually all professional cards which work on the Windows platform will also support Direct3D. This has changed in the professional graphics market from largely Unix-based hardware like SGIs and Suns to less expensive PC-based systems, leading to the growth of Windows in this market segment, while at the same time providing a new market for OpenGL software in Unix-based consumer systems running Linux or Apple OS X.

The main reason for OpenGL's dominance in the professional market was it’s performance. Many professional graphics applications were originally written in IRIS GL for high-end SGI workstations, which were far more capable, both graphically

and in raw CPU power, than the PCs of the time. Later, many of these were ported to OpenGL, even as the personal computer was evolving into a system powerful enough to run some professional graphics applications. Users were able to run Maya, for example, the successor to Alias on SGIs or Windows-based personal computers (and today on Linux, Mac OS X, and Windows). Price competition eventually broke SGI's dominance in the market, but the established base of OpenGL software engineers and the broadening user base for OpenGL in Apple, Linux, and other operating systems, have resulted in a market where both DirectX and OpenGL are viable, widespread APIs.

The other reason for OpenGL's early advantage was marketing and design. DirectX is a set of APIs that were not marketed towards professional graphics applications. Indeed, they were not even designed with those applications in mind. DirectX was an API designed for low-level, high-performance access to broadly available, lower-performance, consumer-priced graphics hardware for the purpose of game development. OpenGL is a much more general purpose 3D API, targeting a full range of graphics hardware from low-end commodity graphics cards up to professional and scientific graphics visualization well out of the range of the average consumer, and providing features that are not necessarily exclusive towards any particular kind of user.

Gaming developers typically haven't demanded as wide an API as professional graphics system developers. Many games don't need overlay planes, stencils, and so on, although this hasn't prevented some game developers from using them when available. In particular, game designers are rarely interested in the pixel invariance demanded in certain parts of the OpenGL standards, which are conversely highly useful to film and computer-aided modeling.

As described in 0 for the development of the VE application used for this thesis we used the XVR VE application framework. The he integrated 3D engine of the XVR framework built on top of OpenGL, allows to manage the visual output not only on a

standard graphical window (either on the web or hosted locally), but also on more advanced devices such as Stereo Projection Systems and Head Mounted Displays.

2.2. Virtual Reality Systems

2.2.1. Mainstream Virtual Reality

Virtual reality (VR) is a term that applies to computer-simulated environments that can simulate physical presence in places in the real world, as well as in imaginary worlds. Most current virtual reality environments are primarily visual experiences, displayed either on a computer screen or through special stereoscopic displays, but some simulations include additional sensory information, such as sound through speakers or headphones. Some advanced, haptic systems now include tactile information, generally known as force-feedback, in medical and gaming applications. Furthermore, virtual reality covers remote communication environments which provide virtual presence of users with the concepts of telepresence and telexistence or a virtual artifact either through the use of standard input devices such as a keyboard and mouse, or through multimodal devices such as a wired glove, head trackers, and omnidirectional treadmills. The simulated environment can be similar to the real world in order to create a lifelike experience, for example, in simulations for pilot or combat training, or it can differ significantly from reality, such as in VR games. In practice, it is currently very difficult create a high-fidelity virtual reality experience, due largely to technical limitations on processing power, image resolution, and communication bandwidth; however, the technology's proponents hope that such limitations will be overcome as processor, imaging, and data communication technologies become more powerful and cost-effective over time.

Virtual reality is often used to describe a wide variety of applications commonly associated with immersive, highly visual, 3D environments. The development of CAD software, graphics hardware acceleration, head mounted displays (HMDs), database gloves, and miniaturization have helped popularize the notion. In the book The Metaphysics of Virtual Reality (Heim 1993), seven different concepts of virtual

reality are identified: simulation, interaction, artificiality, immersion, telepresence, full-body immersion, and network communication. People often identify VR with head mounted displays and data suits.

2.2.2. Immersive VR Systems

Immersive systems are high tech, three dimensional display systems that allow users to be "immersed" into a displayed image. In an immersive environment, images are often displayed in stereoscopic 3D. Tracking systems can also be utilized, enabling a user to move all around these 3D images and even interact with them. The result is an experience that very much looks and feels like it is "real."

(Björk and Holopainen 2004), in Patterns in Game Design, divide immersion into four categories: sensory-motoric immersion, cognitive immersion, emotional immersion and spatial immersion. The last one tends to be the most suitable for the purposes of this thesis. Spatial immersion occurs when a user feels the simulated world is perceptually convincing. The user feels that he or she is really "there" and that a simulated world looks and feels "real".

Head-Mounted Displays (HMDs)

A head-mounted display (Figure 23) is a display device, attached on the head or as part of a helmet . A head-mounted-display can have either one (monocular HMD) or two (binocular HMD) small displays with lenses and semi-transparent mirrors embedded in a helmet, eye-glasses (also known as data glasses) or visor. The display units are miniaturized and may include CRT, LCDs, liquid crystal on silicon (LCos), or OLED. Sometimes multiple micro-displays are employed in order to increase total resolution and field of view.

Figure 23: A binocular HMD

Some of the main characteristics of Head-Mounted Displays are:

The main characteristics of HMDs are (Cakmakci and Rolland 2006):

Ability to show stereoscopic imagery

A binocular HMD has the ability to display separate images to each eye, so it can be used to display stereoscopic imagery. It should be borne in mind that so-called 'Optical Infinity' is generally taken by flight surgeons and display experts as about 9 meters. This is the distance at which, given the average human eye rangefinder "baseline" (distance between the eyes or Inter-Pupillary Distance (IPD)) of between 6 and 8 cm, the angle of an object at that distance becomes essentially the same from each eye. At smaller ranges, the perspective from each eye is significantly different and the expense of generating two different visual channels through the Computer-Generated Imagery (CGI) system becomes worthwhile.

Inter-Pupillary Distance (IPD)

The distance between the two eyes. It is measured at the pupils. It is important in designing Head-Mounted Displays.

Field of view (FOV)

Most HMDs offer considerably less than humans FOV which is around 180°. Typically, a greater field of view results in a greater sense of immersion and better situational awareness. Consumer-level HMDs typically offer a FOV of about 30-40° while professional HMDs offer a field of view of 60° to 150°.

Resolution

HMDs usually mention either the total number of pixels or the number of pixels per degree. Listing the total number of pixels (e.g. 1600×1200 pixels per eye) is borrowed from how the specifications of computer monitors are presented. However, the pixel density, usually specified in pixels per degree or in arcminutes per pixel, is also used to determine visual acuity. 60 pixels/degree (1 arcmin/pixel) is usually referred to as eye limiting resolution, above which increased resolution is not noticed by people with normal vision. HMDs typically offer 10 to 20 pixels/degree, though advances in micro-displays help increase this number.

Binocular overlap

Binocular overlap is the visible area that is mutual to both eyes. Binocular overlap is the basis for the sense of depth and stereo, allowing humans to sense which objects are near and which objects are far. Humans have a binocular overlap of about 100° (50° to the left of the nose and 50° to the right). The larger the binocular overlap offered by an HMD, the greater the sense of stereo. Overlap is sometimes specified in degrees (e.g. 74°) or as a percentage indicating how much of the visual field of each eye is common to the other eye.

Distant focus ("Collimation")

Optical techniques may be used to present the images at a distant focus, which seems to improve the realism of images that in the real world would be at a distance.

A key application for HMDs is training and simulation, allowing to virtually placing a trainee in a situation that is either too expensive or too dangerous to replicate in a real-life. Training with HMDs covers a wide range of applications from

driving, welding and spray painting, flight and vehicle simulators, dismounted soldier training, medical procedure training and more.

Depth perception inside an HMD requires different images for the left and right eyes. There are multiple ways to provide these separate images:

- **Use dual video inputs**, thereby providing a completely separate video signal to each eye. The advantage of dual video inputs is that it provides the maximum resolution for each image and the maximum frame rate for each eye. The disadvantage of dual video inputs is that it requires separate video outputs and cables from the device generating the content.
- **Time-based multiplexing:** Techniques such as frame sequential combine two separate video signals into one signal by alternating the left and right images in successive frames. Time-based multiplexing preserves the full resolution per each image, but reduces the frame rate by half. For example, if the signal is presented at 60 Hz, each eye is receiving just 30 Hz updates. This may become an issue with accurately presenting fast-moving images.
- **Side by side or top/bottom multiplexing:** This method allocates half of the image to the left eye and the other half of the image to the right eye. Side-by-side and top/bottom multiplexing provide full-rate updates to each eye, but reduce the resolution presented to each eye. Many 3D television broadcasts chose to provide side-by-side 3D which saves the need to allocate extra transmission bandwidth and is more suitable to fast-paced sports action relative to time-based multiplexing techniques.

Not all HMDs provide depth perception. Some lower-end modules are essentially bi-ocular devices where both eyes are presented with the same image. 3D video players sometimes allow maximum compatibility with HMDs by providing the user with a choice of the 3D format to be used.

Head-mounted displays may also be used with tracking sensors that allow changes of angle and orientation to be recorded. When such data is available in the system computer, it can be used to generate the appropriate computer-generated

imagery (CGI) for the angle-of-look at the particular time. This allows the user to "look around" a virtual reality environment simply by moving the head without the need for a separate controller to change the angle of the imagery. In radio-based systems (compared to wires), the wearer may move about within the tracking limits of the system.

Our HMD used in the experiments for this thesis was Kaiser Electro-optics Pro-View 50 Head Mounted Display with a Field-of-View comprising 50 degrees diagonal (Figure 24). The HMD was capable of displaying two separate images with 100% overlapping at XGA resolution (1024H x 768V) with full color and 60Hz Vertical scan rate. The projection field of view was 50° diagonal; 30° (V) x 40° (H) and the Inter-Pupillary distance was adjustable from 55 to75 cm.

Figure 24: An HMD of the type used in this research, with no periphery shielding

Stereoscopy

Stereoscopy (also called stereoscopic or 3-D imaging) is a technique for creating or enhancing the illusion of depth in an image by presenting two offset images

separately to the left and right eye of the viewer. Both of these 2-D offset images are then combined in the brain to give the perception of 3d depth. Three strategies have been used to accomplish this: have the viewer wear eyeglasses to combine separate images from two offset sources, have the viewer wear eyeglasses to filter offset images from a single source separated to each eye, or have the light source split the images directionally into the viewer's eyes (no glasses required; known as autostereoscopy).

Stereoscopy creates the illusion of three-dimensional depth from images on a two-dimensional plane. Human vision uses several cues to determine relative depths in a perceived scene. Some of these cues are:

- Stereopsis (horizontal disparity of human eyes)

  This is the difference in the images projected onto the back the eye (and then onto the visual cortex) because the eyes are separated horizontally by the interocular distance.

- Accommodation of the eyeball (eyeball focus)

  This is the muscle tension needed to change the focal length of the eye lens in order to focus at a particular depth.

- Convergence

  This is the muscle tension required to rotate each eye so that it is facing the focal point.

- Occlusion of one object by another

  An object that blocks another is assumed to be in the foreground.

- Subtended visual angle of an object of known size

  The closer a viewed object is, the more visual angle subtends at the eye.

- Linear perspective (convergence of parallel edges)

Objects get smaller the further away they are and parallel lines converge in distance.

- Vertical position

  Objects higher in the scene generally tend to be perceived as further away.

- Lighting shadows:

  Closer objects are brighter, distant ones dimmer. There a number of other more subtle cues implied by lighting, the way a curved surface reflects light suggests the rate of curvature, shadows are a form of occlusion.

- Relative motion:

  Objects further away seem to move more slowly than objects in the foreground.

- Change in size of textured pattern detail:

  Close objects appear in more detail, distant objects less.

- Atmospheric haze:

  Distant objects get blurred by the atmosphere.

- Change in size of textured pattern detail:

  Close objects appear in more detail, distant objects less.

All the above cues, with the exception of the first three, are present in traditional two-dimensional images such as paintings, photographs, and television. Stereoscopy is the enhancement of the illusion of depth in a photograph, movie, or other two-dimensional image by presenting a slightly different image to each eye, and thereby adding the first of these cues (stereopsis) as well. It is important to note that the second cue is still not satisfied and therefore the illusion of depth is incomplete.

Stereopsis appears to be processed in the visual cortex in binocular cells having receptive fields in different horizontal positions in the two eyes. Such a cell is active

only when its preferred stimulus is in the correct position in the left eye and in the correct position in the right eye, making it a disparity detector.

When a person stares at an object, the two eyes converge so that the object appears at the center of the retina in both eyes. Other objects around the main object appear shifted in relation to the main object. In the following example (Figure 25), whereas the main object (dolphin) remains in the center of the two images in the two eyes, the cube is shifted to the right in the left eye's image and is shifted to the left when in the right eye's image (Figure 26).

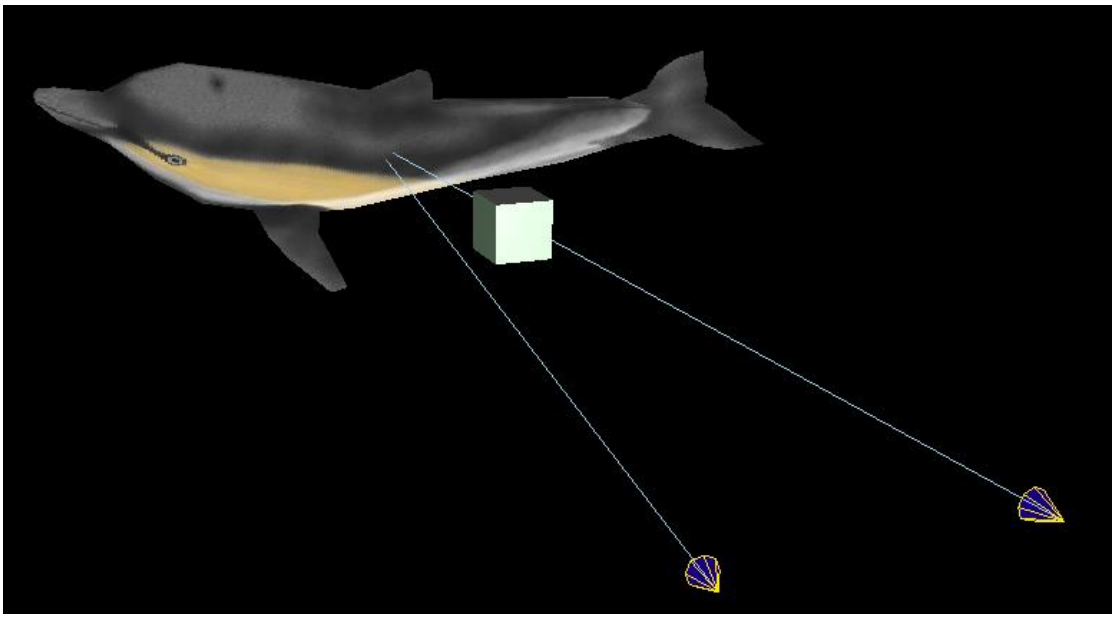


Figure 25: The two eyes converge on the object of attention

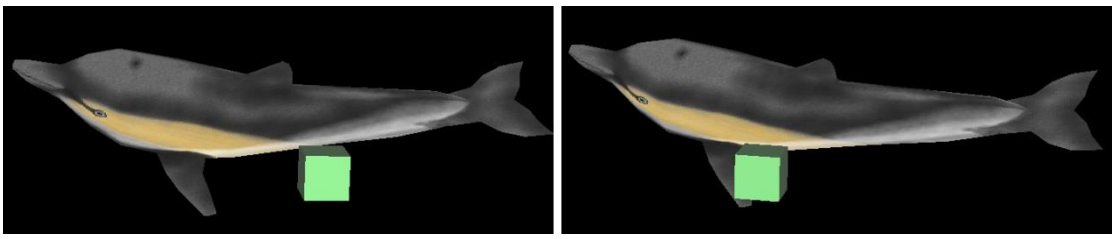


Figure 26: The cube is shifted to the right in left eye's image (left) and to the left on right eye's image (right)

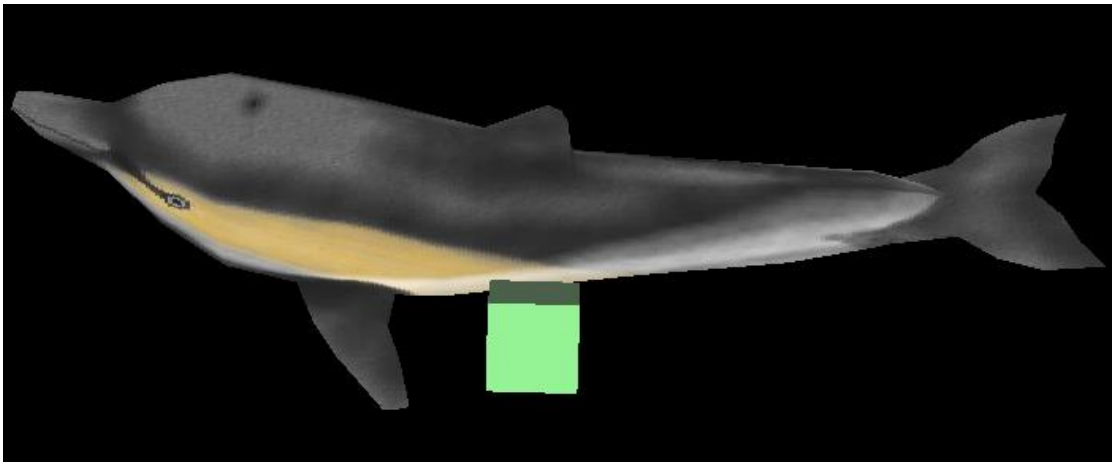


Figure 27: We see a single, Cyclopean, image from the two eyes' images

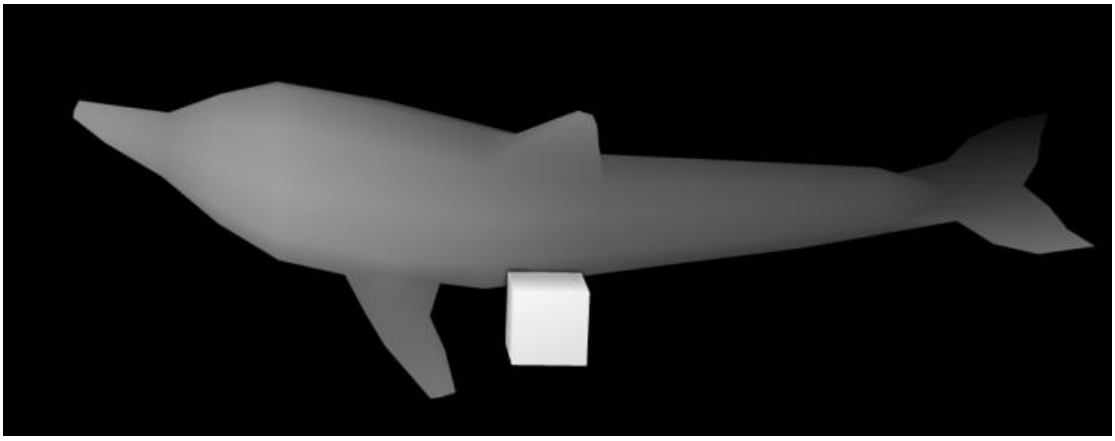


Figure 28: The brain gives each point in the Cyclopean image a depth value, represented here by a grayscale depth map

Because each eye is in a different horizontal position, each has a slightly different perspective on a scene yielding different retinal images. Normally two images are not observed, but rather a single view of the scene, a phenomenon known as singleness of vision. Nevertheless, stereopsis is possible with double vision. This form of stereopsis was called qualitative stereopsis (Ogle 1950). If the images are very different (such as by going cross-eyed, or by presenting different images in a stereoscope) then one image at a time may be seen, a phenomenon known as binocular rivalry.

While stereopsis is considered the dominant depth cue in most people, if the other cues are presented incorrectly they can have a strong detrimental effect. In order to render a stereo pair one needs to create two images, one for each eye in such

a way that when independently viewed they will present an acceptable image to the visual cortex and it will fuse the images and extract the depth information as it does in normal viewing. If stereo pairs are created with a conflict of depth cues then one of a number of things may occur: one cue may become dominant and it may not be the correct/intended one, the depth perception will be exaggerated or reduced, the image will be uncomfortable to watch, the stereo pairs may not fuse at all and the viewer will see two separate images (Figure 29).

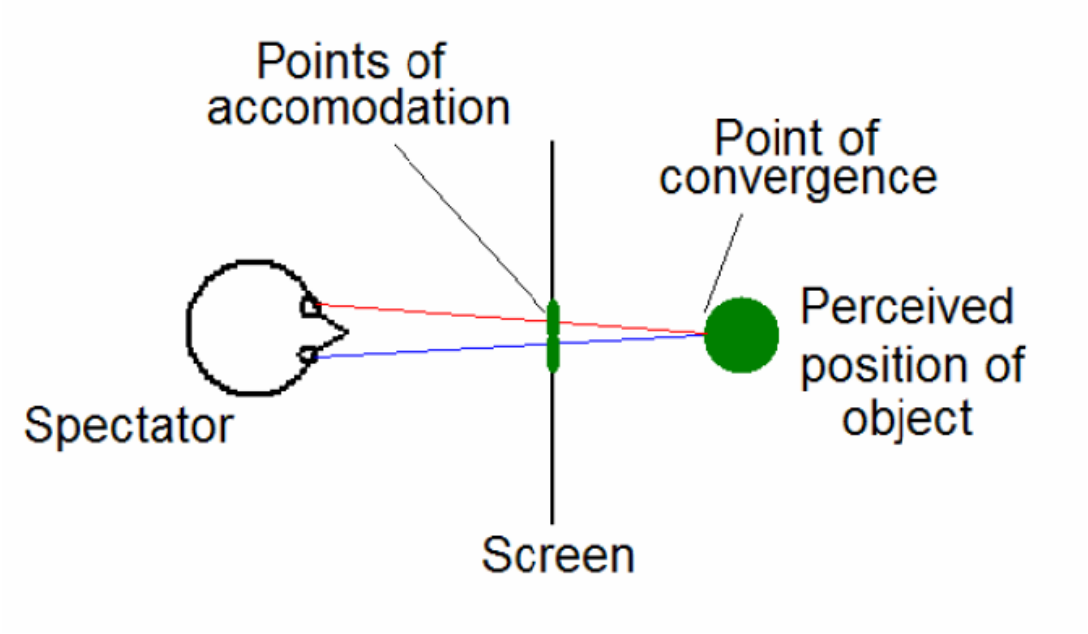


Figure 29: Different points of convergence and accommodation in stereopsis.

Stereoscopic Rendering

There are a couple of methods of setting up a virtual camera and rendering two stereo pairs, many methods are strictly incorrect since they introduce vertical parallax. An example of this is called the "Toe-in" method, while incorrect it is still often used because the correct "off axis" method requires features not always supported by rendering packages. Toe-in is usually identical to methods that involve a rotation of the scene. The toe-in method is still popular for the lower cost filming because offset cameras are uncommon and it is easier than using parallel cameras which requires a subsequent trimming of the stereo pairs.

On-axis (Incorrect)

In this projection the camera has a fixed and symmetric aperture; each camera is pointed at a single focal point. Images created using the "on-axis" method will still appear stereoscopic but the vertical parallax it introduces will cause increased discomfort levels. The introduced vertical parallax increases out from the center of the projection plane and is more important as the camera aperture increases (Figure 30).

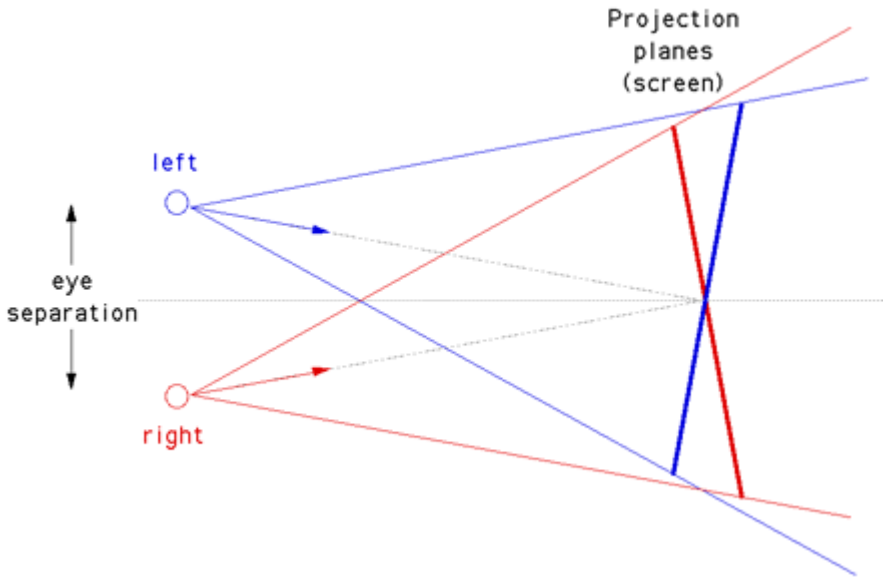


Figure 30: On-axis stereo rendering (incorrect).

Off-axis (Correct)

This is the correct way to create stereo pairs. It introduces no vertical parallax and is therefore creates the less stressful stereo pairs. Note that it requires a non-symmetric camera frustum (Figure 31); this is supported by some rendering packages, in particular, OpenGL.

Objects that lie in front of the projection plane will appear to be in front of the computer screen, objects that are behind the projection plane will appear to be "into" the screen. It is generally easier to view stereo pairs of objects that recede into the screen; to achieve this one would place the focal point closer to the camera than the objects of interest. Note this doesn't lead to as dramatic an effect as objects that pop out of the screen.

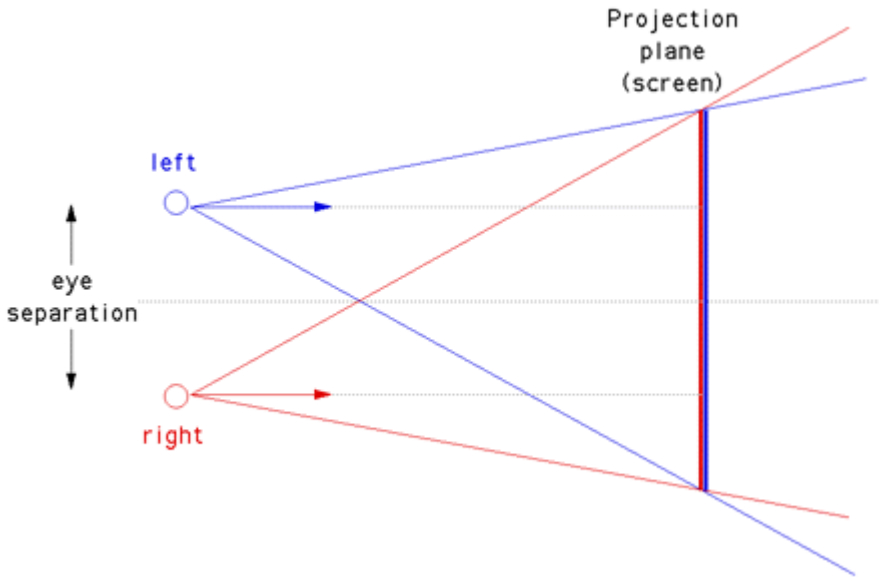


Figure 31: Off-axis stereo rendering (correct).

The degree of the stereo effect depends on both the distance of the camera to the projection plane and the separation of the left and right camera. Too large a separation can be hard to resolve and is known as hyperstereo. A good ballpark separation of the cameras is 1/20 of the distance to the projection plane; this is generally the maximum separation for comfortable viewing. Another constraint in general practice is to ensure the negative parallax (projection plane behind the object) does not exceed the eye separation.

A common measure is the parallax angle defined as $theta = 2\tan^{-1}(Dx/2d)$ where Dx is the horizontal separation of a projected point between the two eyes and d is the distance of the eye from the projection plane (Figure 32). For easy fusing by the majority of people, the absolute value of theta should not exceed 1.5 degrees for all points in the scene. Note $theta$ is positive for points behind the scene and negative for points in front of the screen. It is not uncommon to restrict the negative value of theta to some value closer to zero since negative parallax is more difficult to fuse especially when objects cut the boundary of the projection plane.

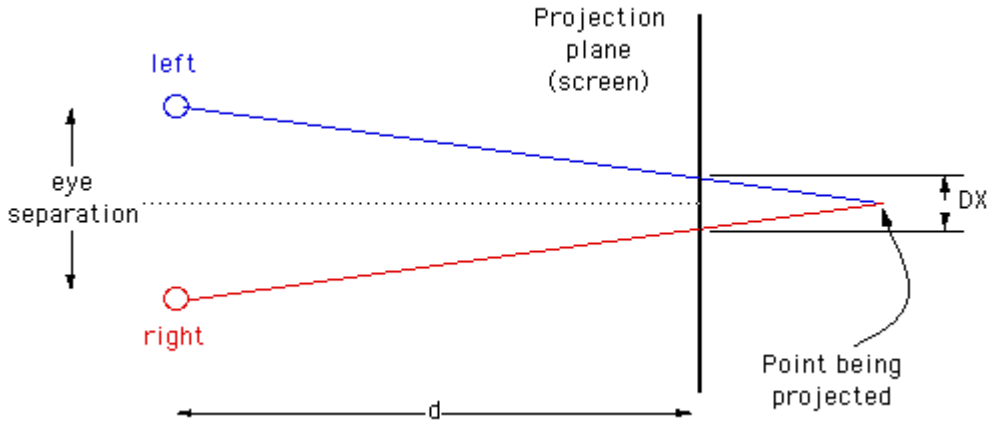


Figure 32: Off-axis stereo rendering equation.

Head Tracking

Head-mounted displays may also be used with tracking sensors that allow changes of angle and orientation to be recorded. When such data is available in the system computer, it can be used to generate the appropriate computer-generated imagery (CGI) for the angle-of-look at the particular time. This allows the user to "look around" a virtual reality environment simply by moving the head without the need for a separate controller to change the angle of the imagery. In radio-based systems (compared to wires), the wearer may move about within the tracking limits of the system.

There are currently two types of available tracking device that may be employed according to their capability of tracking rotational movement (3-degrees-of-freedom) and rotational as well as translational movement (6-degrees-of-freedom).

The 3-degrees-of-freedom (3-dof) tracker allows the viewpoint to be rotated around the x, y and z axis, which provides the ability to orient the viewing direction left or right, up or down, or twist it laterally. When using a 3-dof tracker only rotation is accounted for with no provision for translating the viewpoint through the environment except on pre-defined paths, so the application of such devices will typically require the subject to remain seated in a static location.

By comparison, the 6-degrees-of-freedom (6-dof) devices track not only the rotational position of the head, but also the physical location in space. 6-dof devices therefore allow the environment to not only be scrutinized by rotating the head, but also navigated by physically walking or moving around. It should be noted however that since the HMD will obscure the users normal view of the physical environment and because of the attached cabling, this type of interaction with the environment must be carefully thought out and does not lend itself to every application.

A common set of metrics is to evaluate the performance of the of head trackers are

- Accuracy

  This is a measure of the error in the position and orientation reported by the tracker.

- Resolution

  This is the smallest change in position and orientation that can be detected by the tracker.

- Update rate

  This is the rate at which position and orientation measurements are reported by the tracker to the host computer.

- Latency (also known as Lag)

  This is the delay between a change in position and orientation and the report of the change to the host computer.

- Working volume

  This is the volume within which the tracker can measure position and orientation with its specified accuracy and resolution.

A good position tracker should have high accuracy, ne resolution and high update rate. Its latency should be low and its working volume should be large. Ideally, it should not need any specialized environment for operation.

In addition, the parts of the tracker that need to be worn should be small and light in weight, to ensure user comfort.

2.3. Effects of Latency in Virtual Environment Users

End-to-end latency in a Virtual Environment (VE) is defined as the time lag between a user's action in the VE and the system's response to this action. VE lag comprises of four different types

- ***User-input-device lag***

The input device reports 3d position and/or orientation data to the VR application. The user input device lag is the lag introduced from the communication between the tracking system and the VE application.

- ***Application-dependent processing lag***

The application-dependent processing lag is the time required for the computation of the 3D model. Once the user input device data arrives to the host workstation, the application processes it. Processing lag is highly application-dependent and thus highly variable.

- ***Rendering lag***

The rendering lag is the time that passes until data sent from the VE application to the rendering hardware appears on a monitor or immersive display. The rendering lag depends on the scene and the viewpoint rendered at each time, so it varies through the VE application run-time. The scan-out of the display causes additional lag.

- ***Synchronization lag***

The synchronization lag is the total time that data is waiting during the necessary communication of involved input devices, in-between the parallel processing stages of the VE application. It is application-relevant and depends on rendering processing stages which are not well-synchronized to avoid delays of transmission. These stages are independent and it is possible that the input device deposits new tracking data

shortly after the application reads the previous data. Thus, the application is busy processing the previous input before it reads and starts to process the new input, so that input data is delayed.

- *Frame-rate-induced lag*

Moreover, there is a fifth kind of lag, i.e. the frame-rate-induced lag, resulting from the fact that data displayed progressively become out of date, while the display is not updated fast. This type of lag is distinguished from other lag sources and is not considered as end-to-end latency. It is, though, perceivable by the users and results in slow frame rates while exposed to a VE to be considered unacceptable (Wloka 1995).

Excessive system latency is a well-known defect of VE and teleoperation systems (Ellis, Mania, et al. 2004). It is particularly troublesome for head-tracked systems since delays in head orientation measurement give rise to errors in presented visual direction.

Perceptible latency that is experienced by its visual consequences on a display is one of the most notable problems facing current VE applications. (Ellis, Adelstein, et al. 1999). Perceptible latency has been shown to have undesirable effects on users of virtual environments, including a lack of accuracy during tracking tasks, loss of immersion (Garrett, Aguilar and Barniv 2002) and cybersickness, a form of motion sickness that occurs as a result of exposure to VEs, poses a serious threat to the usability of VR systems and is one of the most important health and safety issues that may influence the advancement of VE technology (Stanney, Mourant and Kennedy 1998), as well as disorientation, discomfort and even nausea (Kennedy, et al. 1992).

Although manufacturers of high performance computer graphics systems often sacrifice latency for frame rate, findings of (Ellis, Adelstein, et al. 1999) suggest they could improve their systems' interactivity by altering their existing trade-off.

High end-to-end latency can severely degrade users' performance in a VE (Ellis, Bréant, et al. 1997), (Ellis, Wolfram and Adelstein 2002). The RMS (Root Mean Square) tracking errors, which are an objective measure of user's performance, are

caused mostly by visual latency, rather than spatial sensor distortion or low update rates (Ellis, Adelstein, et al. 1999). Latency also affects users' performance on 3D object placement tasks (Watson, et al. 2003), in terms of completion time and accuracy (Liu, et al. 1993). While users can exhibit sensorimotor adaptation that might improve manual performance when time delays exist in situations where task preview is available (Cunningham, Billock and Tsou 2001), (Cunningham, Chatziastros, et al. 2001), the presence of delay has been shown to hinder operator adaptation to other display distortions such as static displacement offset (Held, Efstathiou and Greene 1966).

More recently interest has been directed towards the subjective impact of latency on the users' reported sense of presence. Latency, as well as update rate, is considered as a factor affecting the operator's sense of presence in the environment (Welch, et al. 1996), (Uno and Slater 1997). Lower latencies were associated with a higher self-reported sense of presence and a statistically higher change in heart rate for users, while exposed to a stress-inducing (fear of heights), photorealistically rendered VE, involving walking around a narrow pit (Meehan, et al. 2003) (Figure 33). The role of VE scene content and resultant relative object motion on latency detection has been examined by presenting observers in a head-tracked, stereoscopic head mounted display with environments having differing levels of complexity ranging from simple geometrical objects to a radiosity-rendered scene representing a hypothetical real-world setting (Mania, Adelstein, et al. 2004). Such knowledge will help understand latency perception mechanisms and, in turn, guide VE designers in the development of latency countermeasures. In this study, a radiosity-rendered scene of two interconnected rooms was employed. Latency discrimination observed was compared with a previous study in which only simple geometrical objects, without radiosity rendering or a 'real-world' setting, were used employing formal psychophysical techniques which are far-removed from simulated tasks. The user is instructed to report the consequences of latency focused on differences between paired stimuli of varied tracking latency. They reveal that the Just Noticeable Difference (JND) for latency discrimination by trained observers, averages ~15ms or

less, independent of scene complexity and real-world meaning. Such studies were, though, far-removed from real application scenarios of interaction with synthetic scenes or remote telemanipulation applications because the user is required to solely focus on identifying the visual or other consequences of latency while no other task is performed. Moreover, there is always an issue that due to the intense nature of psychophysical experimentation, a small amount of users are normally tested, resulting in doubts concerning the generality of such results.

Figure 33: Scene for testing presence in a VR (Meehan, et al. 2003)

Although previous research identifies the detrimental effect of tracking latency on mainly motor task performance, it could be true that for generic spatial awareness tasks such as navigation or visual search, the effect of latency is less severe and humans adapt to it while forming a detailed mental map of the space. In this thesis, we explore whether the effect of latency on spatial cognition, memory performance and spatial awareness severely hampers spatial awareness in an Immersive Virtual Environment (IVE) or whether humans are good at perceiving 3D space while using the memory awareness methodology detailed below, irrespectively of relatively high latency.

2.4. Measuring Latency

In order to examine the effects of latency in a VE it is necessary that end-to-end latency is effectively measured minimized and controlled. Several measurement techniques have been introduced through years.

In an early measurement method (Liang, Shaw and Green 1991), an electromagnetic tracker was attached to a moving pendulum (Figure 34). Tracker readings were time-stamped and stored in a host computer. The computer monitor was displaying the current time of the clock that was generating the time stamps. A video camera was simultaneously recording the monitor display and the swing of the pendulum. The video was later analyzed frame-by-frame. End-to-end latency was determined by comparing the display time when the pendulum was passing by the zero-crossing point and the time stamp stored on the host computer.

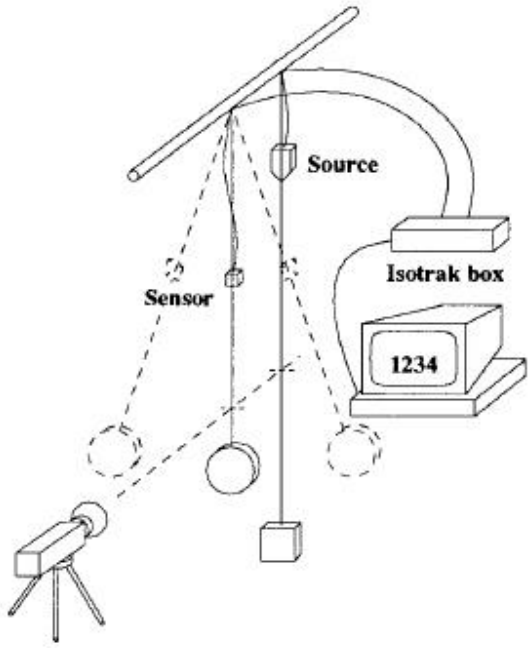


Figure 34: Early latency measurement technique using a camera (Liang, Shaw and Green 1991)

Later, the use of an oscilloscope instead of a video camera was introduced (Miné 1993), (Jacoby, Adelstein and Ellis 1996). The oscilloscope was used to compare three inputs, estimating the end-to-end latency. The first input was deducted from a LED-photodiode pair that was marking the zero-crossing point of the pendulum. The second input was deducted from a Digital-to-Analog (D/A) converter attached to the host computer, reporting tracker position readings. Comparing these two inputs

determined the input device lag. Additionally, a third photodiode was monitoring brightness changes on the system's display, while a specific polygon displayed was changing color from white to black and vice-versa at the time that zero crossings were reported to the system. Comparison between the first and the third input was used to measure the overall end-to end latency.

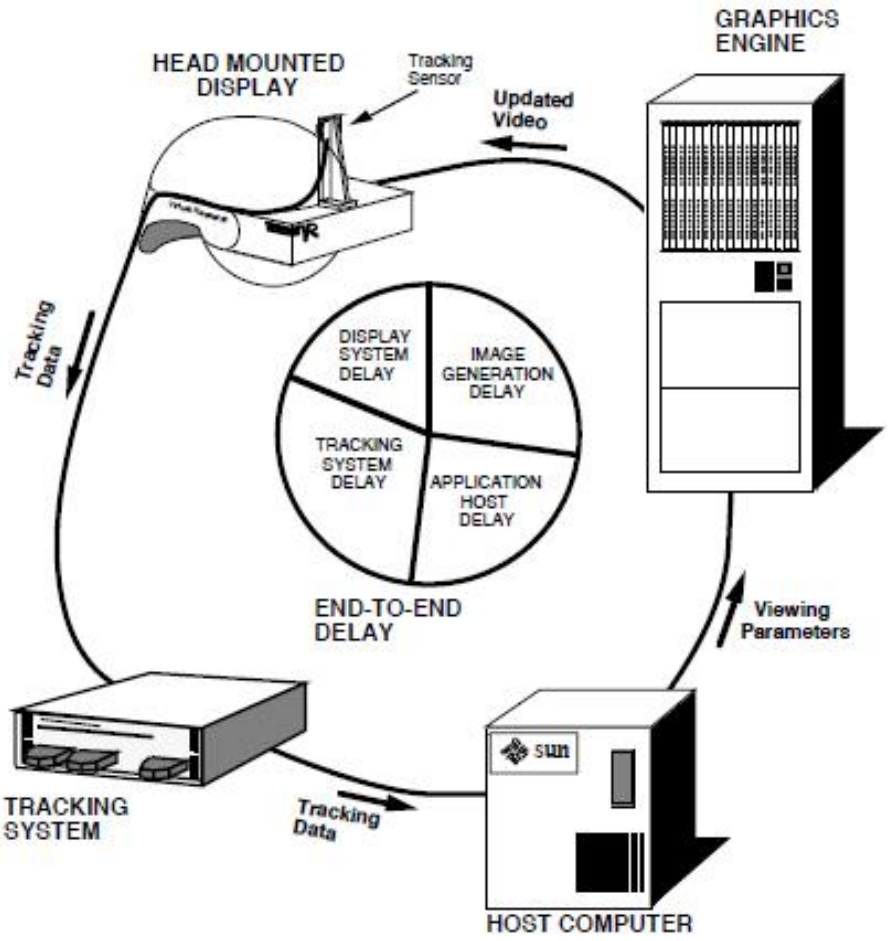


Figure 35: Miné's latency measuring technique using an oscilloscope (Miné 1993)

In a more recent study, a slightly modified technique was used (Jacoby, Adelstein and Ellis 1996). Instead of the first LED-photodiode pair responsible for monitoring and reporting the motion of the pendulum, a swing arm motor equipped with a shaft encoder was used. The arm repeatedly moved the tracker back-and-forth through a pre-set threshold point and the encoder reported crosses of the threshold to an oscilloscope. This input was, at first, compared with the input deducted from a

photodiode monitoring the VE system's screen, which was displaying the same rectangular color transition as in (Miné 1993).

In another study, the previous techniques are modified by directly monitoring the RGB analog output signals of the VGA, instead of using a photodiode in order to monitor the display (Hill, Adelstein and Ellis 2004). Critical portions of the VE application code are also "trapped", thus, producing timestamps and signals to the oscilloscope. These signals were used to bridge internal and external measurements and provide information about timing at different stages of the VE execution. However, taking into account the RGB signal instead of the photodiode readings of the monitor does not fairly offer an accurate measurement of the end-to-end latency, as it does not correspond to what the user actually sees. Relevant research also attempts to minimize latency by assessing its level relevant to the internal system components, and reorganizing the communication between them more efficiently.

Recent estimating methods make use of video analysis comparing movement of the head tracker and the resulting movement of a simulated image on a screen that are captured simultaneously using a video camera (Steed 2008). However, such estimation methods do not provide more accurate measurements nor do they provide information concerning latency increases throughout the processing of the interactive VE itself. This information is essential in order to be able to understand how these different stages contribute to the overall latency and thus, be able to reorganize these components in order to achieve minimal latency.

Recent estimating methods make use of video analysis comparing movement of the head tracker and the resulting movement of a simulated image on a screen. The simultaneous movements are captured using a video camera (Steed 2008) or encoded by photodiode readings of luminance gradients (one gradient that the tracked object is moved across and another gradient that is produced by the VE) (Di Luca 2010). However, such estimation methods result in potentially less accurate measurements than using oscilloscope readings of electronic signals from the VE host computer and, therefore, cannot be further expanded in order to provide information

concerning latency increases throughout the processing of the interactive VE itself. This information is essential in order to be able to understand how these different stages contribute to the overall latency and thus, be able to reorganize these components in order to achieve minimal latency.

In 3.8 of this thesis we presented a custom-made mechanism of measuring and minimizing end-to-end head tracking latency in an immersive VE. Our mechanism builds on previous mechanisms by using an oscilloscope to compare two signals, assembled by low-cost, custom-made and portable equipment. One signal is generated by the head-tracker movement and reported by a shaft encoder attached on a servo motor moving the tracker. The other signal is generated by the visual consequences of this movement in the VE and reported by a photodiode attached to the computer monitor. The end-to-end head tracking latency of the VE is the measured time-shift between these two signals. The presented system calculates this time-shift by off-line processing the tracker position and display brightness measurements stored in a computer derived from the oscilloscope using a USB connection. Thus, an accurate measuring mechanism is provided, utilizing equipment commonly found in an academic facility.

2.4.1. Virtual Environment Pipeline

The VE visual pipeline (Figure 38) covers all steps up to the display of a VE scene on the output device of interest starting from the sensor inputs that contribute to the rendering of that scene. The graphics pipeline typically accepts some representation of three-dimensional primitives as an input and results in a 2D raster image as output. OpenGL and Direct3D are two notable 3d graphic standards, both describing very similar graphic pipeline. The latency for a VE system is the sum of the completion times required for each of the consecutive processes in the pipeline illustrated by Figure 38.

The tracker acquires the current position and orientation information of the head. Although various motion trackers may use different transduction methods to acquire position and orientation information, we are only concerned with latency. Therefore,

the relevant metric for a given tracker is simply the length of time between data acquisition and when that data is deposited into shared memory and made available to the next component in the pipeline, the simulation/graphics subsystem. This data is deposited, through the tracker driver into shared memory, making data available to the next component in the pipeline, e.g. the simulation application.

The simulation/graphics subsystem carries out a number of actions on the tracker data retrieved from shared memory. First, application-related calculations that are part of the physical simulation or graphical user interface are performed on the CPU. These are calculations that may be viewpoint dependent and may impact on the viewable image. Any simulation or application-related calculations that do not rely on updates of the user's viewing position or of other sensed input action can be pre-computed outside of the direct path of the VE pipeline, and therefore do not need to contribute to overall latency.

After the completion of these calculations, the geometry of the scene is transferred to the Graphics Processor Unit (GPU) of the graphics card. In the case of stereoscopic visualization, viewpoint transformations specific to each channel of the stereo viewpoint are performed. The resulting image (or images in case of stereo viewpoints) is subsequently rasterized into pixels and drawn into a temporary buffer of the graphics hardware video ram (VRAM), e.g. the back video buffer. The rendering time of an image may depend on the number of polygons, vertices, textures, pixels, or a combination of all of these. The final stage of the pipeline is the buffer swap from the back to front buffer (Figure 36Figure 37, Figure 37). The use of the back buffer at the VRAM serves as a staging area where images are assembled before scanned onto the display in order to avoid visual discontinuities and artifacts that would otherwise occur if parts of the front buffer were changed during the scan-out process. To avoid visual discontinuities that would arise from modifying the front video buffer while scanning to the screen, the buffer swap is generally performed during the vertical blank interval of the monitor, when the screen is dark.

After the image is swapped to the front buffer, pixels will be scanned out onto the display in rows from the top of the screen down, and from left to right within each row. Because of the scan-out, the latency of a scene will vary, being lower in the upper left portion of the display than in the lower right. To simplify the discussion of latency in this thesis, we refer to the latency in the system only up to the uppermost left pixel of the output device. At each refresh cycle of the screen a pulse called V-sync is generated. Synchronization between buffer swapping and the v-sync signal prevents "image tearing" by timing video time swaps to match the vertical blank interval of the monitor. The v-sync pulse indicates when the front buffer is ready to accept the next video frame, and only then the back buffer is being swapped. When synchronization between the back/front buffer swap and the v-sync pulse is used the maximum frame rate of the graphics redraw is limited by the interval between successive v-syncs, no matter how quickly video buffers can be filled with newer image information.

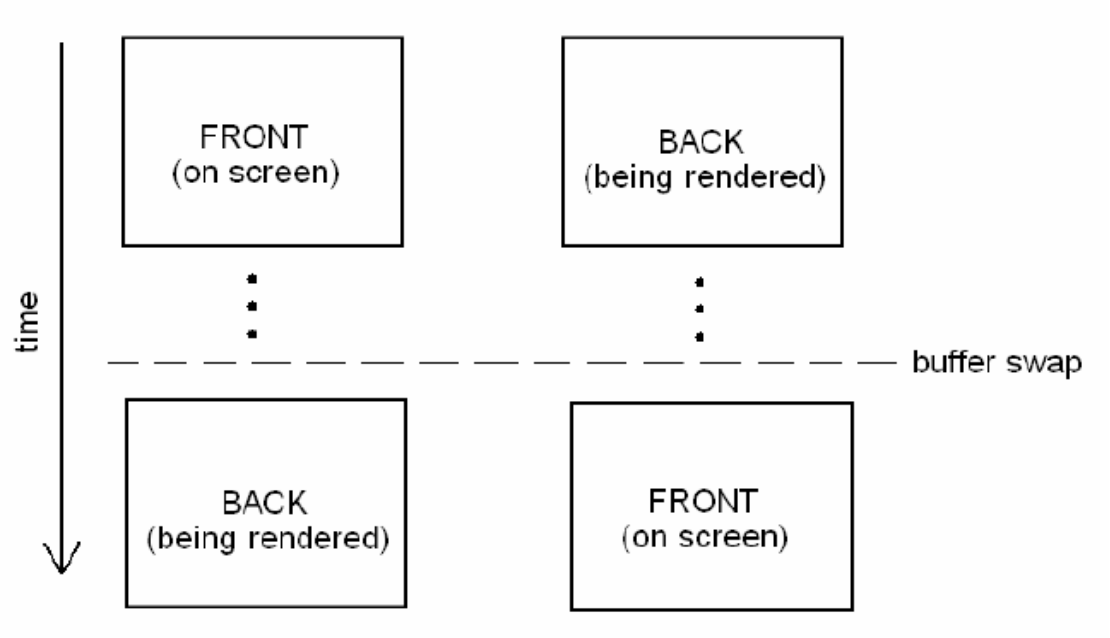


Figure 36: Double buffering in monoscopic rendering

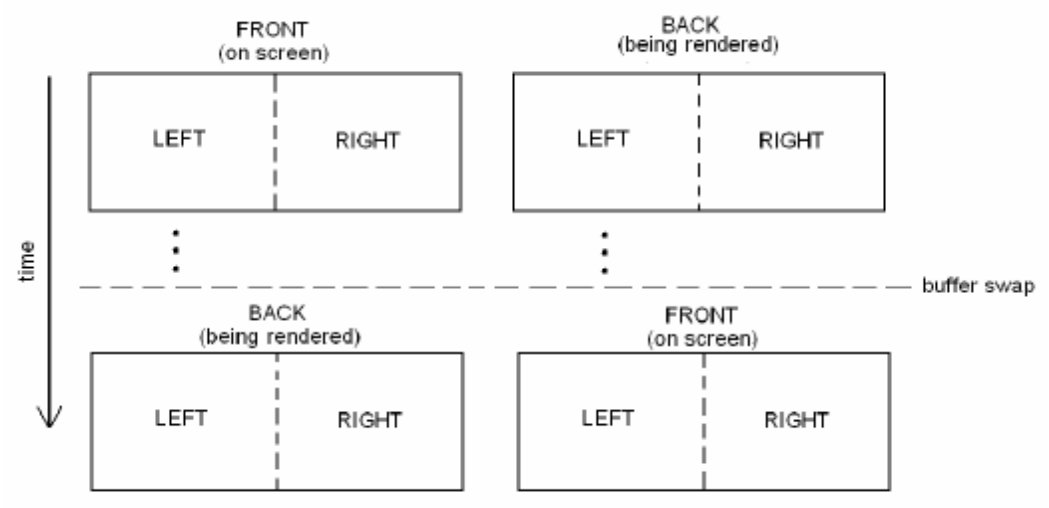


Figure 37: Double buffering in passive stereo rendering.

The latency of the VE system is the sum of the completion times required for each of the consecutive processes in the pipeline to be processed. In order to measure end-to-end latency, we have to also take into account the time that pixels take to be scanned onto the display; this is dependent on the hardware of the display. Thus, the term 'internal latency' indicates the latency contributed from all the processes of the VE visual pipeline except from the scan-out process.

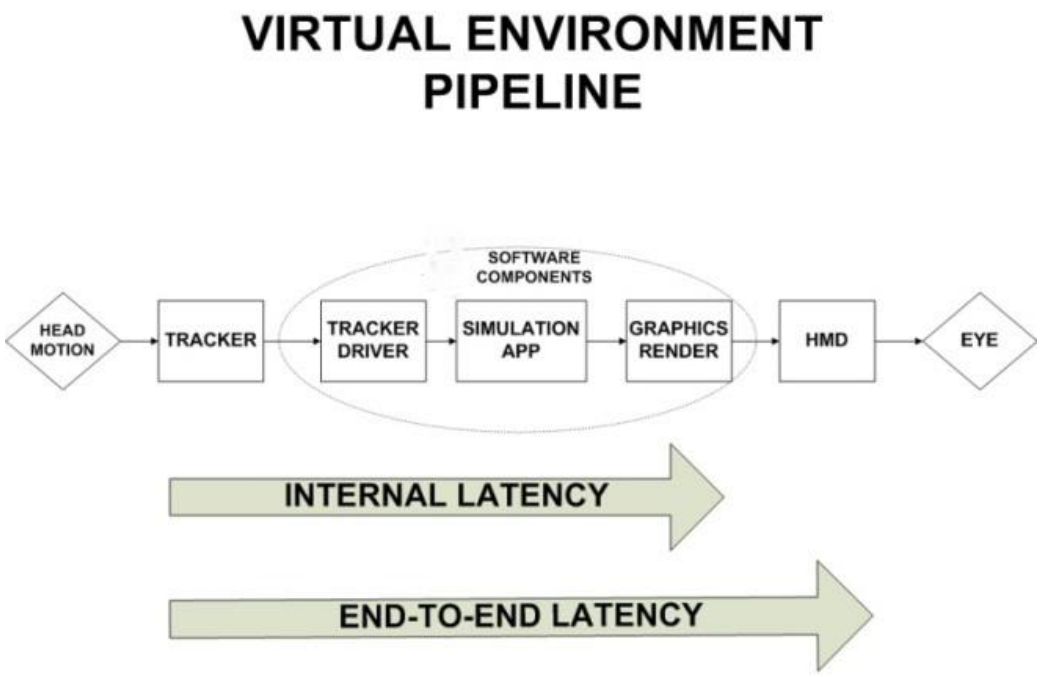


Figure 38: The VE pipeline

2.4.2. Added Latency sources

Triple Buffering

Multiple buffering is the use of more than one buffer to hold a block of data, so that a "reader" will see a complete (though perhaps aged) version of the data, rather than a partially-updated version of the data being created by a "writer".

In computer graphics systems double buffering of video frames to avoid visual discontinuities is commonly used. While the front buffer in video memory (VRAM) is directly scanned onto a monitor as an analog video signal, the back buffer in VRAM serves as a staging area where new images can be assembled. With double buffering, an image is swapped to the front buffer only when the image has been completely assembled in the back buffer. This prevents visual artifacts that would otherwise occur if parts of the front buffer were changed during the scan-out process. Synchronization between buffer swapping and the v-sync signal prevents image "tearing" (Meehan, et al. 2003) by timing video buffer swaps to match the vertical

blank interval of the monitor. The back buffer is swapped to the front to begin scanning only when the display hardware is ready to accept the next video frame, as indicated by the VGA video v-sync pulse. Double buffering necessarily requires more video memory and CPU time than single buffering because of the video memory allocated for the back buffer, the time for the copy operation, and the time waiting for synchronization. Moreover, when v-sync/buffer swap synchronization is used, the interval between successive v-syncs limits the maximum frame rate of the graphics redraw, regardless how quickly video buffers can be filled with new image information.

When v-sync is enabled, the graphics card can often fill both buffers and then have to stop working on any new frames until the monitor indicates it is ready for a new frame for its next refresh. Only then can the graphics card clear the primary buffer, switch buffers and begin rendering the next frame in the secondary buffer. This waiting is what causes a drop in FPS when v-sync is enabled on many systems. In order to avoid this drop, modern graphic cards actually triple-buffer the video memory. In triple buffering the VE pipeline o back buffers and can immediately start drawing in the one that is not involved in such copying. The third buffer, the front buffer, is read by the graphics card to display the image on the monitor. Once the monitor has been drawn, the front buffer is flipped with (or copied from) the back buffer holding the last complete screen. Since one of the back buffers is always complete, the graphics card never has to wait for the software to complete. Consequently, the software and the graphics card are completely independent, and can run at their own pace. Finally, the displayed image was started without waiting for synchronization and thus with minimum lag.

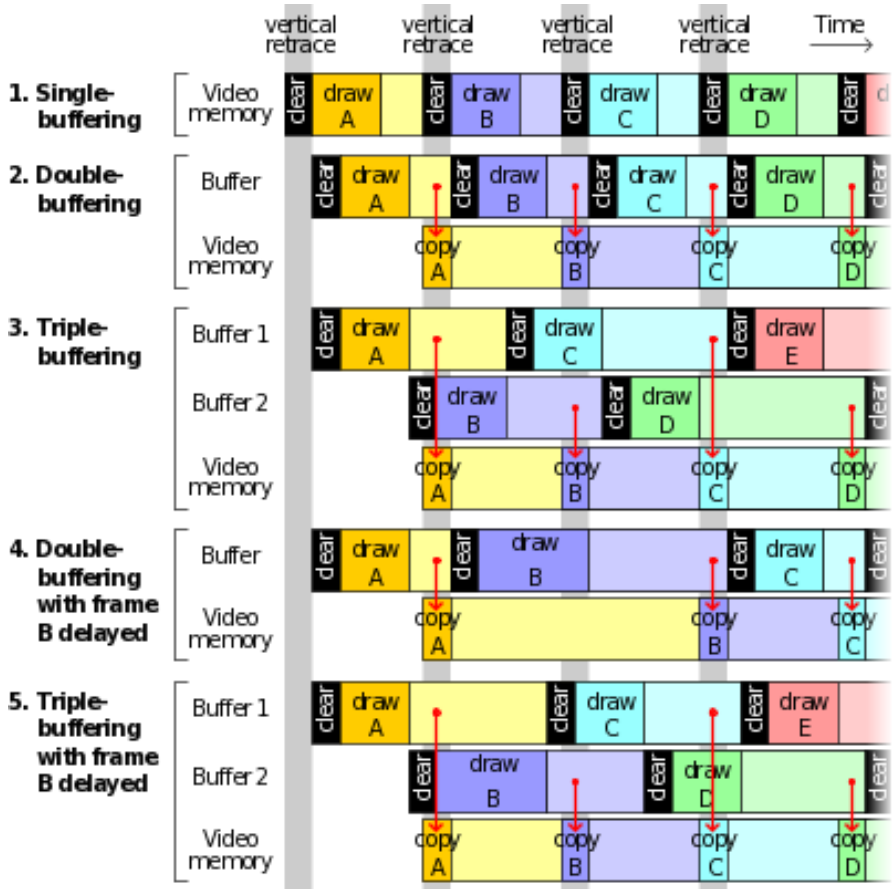


Figure 39: Single, double and triple buffering operation (v-sync enabled).

Dynamic and Static Asynchrony

While the triple-buffer's frame of additional delay involves extra time between the simulation/graphics application software and the output display hardware, the remaining of unnecessary latency, stems from a lack of synchronization between the software and the input tracking device. This asynchrony between the tracker and simulation/graphics subsystems has two components: one that dynamically varies from update-to-update and the other that remains constant.

The dynamic component results from the absence of synchronization between the tracker readings and graphics application updates. Without synchronization, the tracker sampling frequency is not identical to (nor an exact integer multiple of) the graphics VSync rate. For example, as (Hill, Adelstein and Ellis 2004) describe, for a tracker that updates by ~120Hz rate and a VE application that generates graphics at an ~60hz rate, the time of data retrieval from shared memory by the ~60 Hz

simulation/graphics application keeps slipping farther behind the instant data is deposited by an ~120 Hz tracker driver. This slip represents a linear growth in age from one cycle to the next of data in shared memory. This growth continues until the shared memory contents are old enough (1/120 sec or 8.3ms) to be supplanted by a fresher sample from the tracker. The slip and resetting is illustrated in Figure 40 by the repeating (cyan) wedge-shaped segments, representing the duration data remains in shared memory from arrival until retrieval by the simulation/graphics application. Based on the ideal of the simple ramp pattern and the measured minimum and maximum values reported in Figure 40, shared memory time accounts for a theoretical average (mean ± variance) of 4.2 ± 1.2 ms in additional latency.

The constant component is the interval that follows completion of all simulation graphics computations for the cycle until that cycle's image is swapped into the front video buffer. While this latency component follows completion of the cycle's computations, it is still due to static asynchrony with the input device. Because tracker data is read from shared memory at the top of the 60 Hz simulation/graphics cycle, and because all computation occupies only the first 1.3ms in Hill's study (2004), (summing from the application, left eye, and right eye simulation application components in Figure 40), the remainder of the cycle until the video buffer swap is spent idle. Essentially, the results of the completed simulation and graphics calculations age by the duration of this idle time. As measured from the system once the extra 16.7ms from triple buffer has been removed, Figure 40 indicates that the idle time following all computation can add up to 14.5ms of pre-swap delay to the latency path between tracker readings and visual display. Removal of the triple buffer was necessary to make the plot in Figure 40 because this was the only way to associate particular OpenGL buffer swap function callbacks with their specific simulation frame. A portion of the pre-swap delay is unavoidable because it is still needed to perform queued GPU calculations. The triple buffer's extra latency was removed in the manner described in a subsequent section.

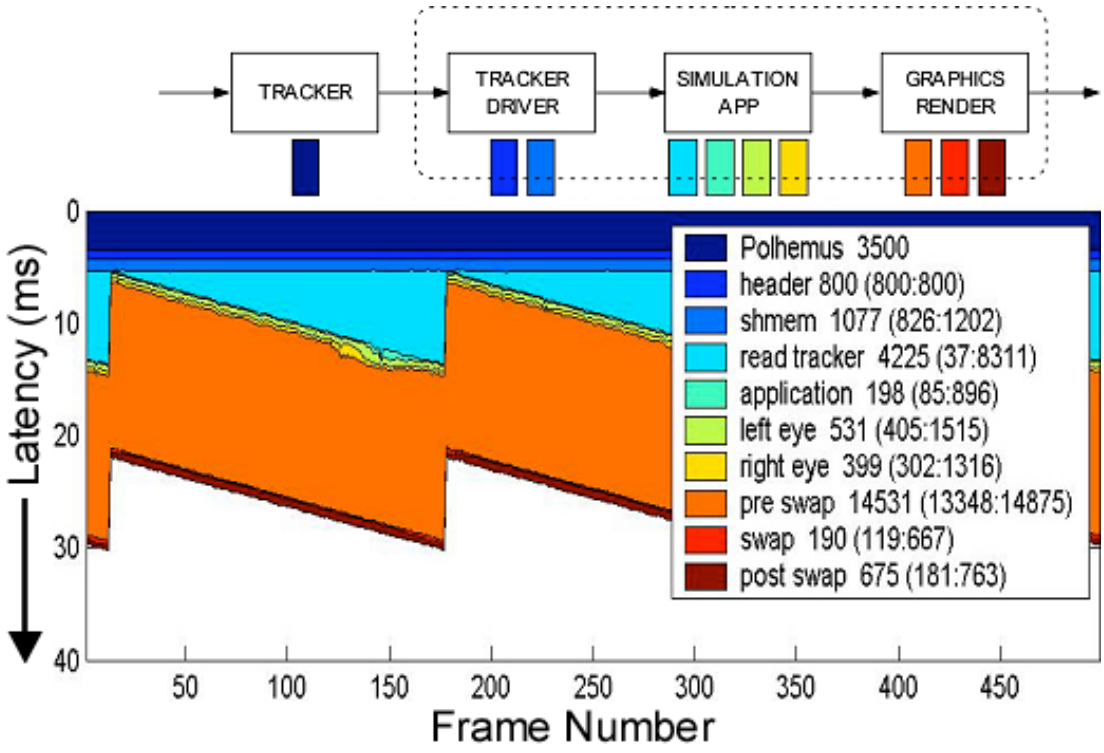


Figure 40: VE system timing diagram (Hill, Adelstein and Ellis 2004)

2.4.3. Hardware

2.5. Latency Minimization

In a VE, the expected minimum achievable latency is the sum of lag contributed from each of the three pipeline components, i.e. the tracking system component, the application-dependent processing component and the rendering component (Wloka 1995) (Figure 38). According to previous research on latency minimization, the initial latency measured (45 ± 1.8ms) was by far higher than the minimum predicted by adding known latencies of the system components (Hill, Adelstein and Ellis 2004). The additional lag was found to be caused by the synchronization lag and lies between the three other subsystems. Part of this lag lies between the graphics application and the display rendering subsystem and is mostly caused by the synchronization that occurs between the buffer swapping and the v-sync signal (Hill, Adelstein and Ellis 2004). The rendering subsystem video memory (VRAM) consists of two buffers, the back buffer where changes are made to the image and the front buffer from which the image is displayed at the screen. Moreover, most of the

modern graphic cards, in fact, triple-buffered the image, adding an extra frame of delay to the VE system. Though this triple buffering may have been useful for high performance 180 Hz CRT displays, this was unnecessary for the 60Hz Head Mounted Displays (HMDs) used in previous work (Hill, Adelstein and Ellis 2004), as well as, for the Rockwell-Collins Proview XL50 to be used in future work. Triple buffering and vertical-sync can be disabled though, by turning off the proper settings in the graphics card's control panel. To prevent image tearing, the v-sync signal may be reused, without restricting the buffers swap rate, ensuring that swapping is regulated in order that only one swapping occurs at each v-sync.

Lack of synchronization also occurs between the application software and the tracking device. This asynchrony consists of two components, one that varies at each update and one that remains constant. The varying component, called dynamic asynchrony, results from the absence of synchronization between the tracker device readings and the updates of the graphics application. The tracker sampling sequence is usually not identical to the graphics v-sync rate. This asynchrony has been eliminated (Hill, Adelstein and Ellis 2004) by synchronizing the tracker readings with the v-sync signal, after doubling the signal frequency (60 Hz) in order to match the one of the tracker (Polhemus Fastrack, 120Hz). The constant component, called static asynchrony, is the time that passes when the data processing is completed, however, at the same time the data remain idle waiting for the next monitor update cycle. Avoiding this "ageing" of processed data, implies receiving data from the tracker the last possible instant, necessary for completing all the computation needed to display the next frame in time. Since the update rate of the screen in (Hill, Adelstein and Ellis 2004) was 60Hz and the tracker update rate was 120 Hz, at each update cycle of the graphics subsystem, the tracker reports data two times. Software and hardware modifications can be used in order that the graphics subsystem skips the first reading of the tracker and uses the "fresher" second one, such as internally clocked "sleep" functions of the graphics application. This modification resulted in the elimination of up to 8.3ms of added latency (half of the screen update cycle) (Hill, Adelstein and Ellis 2004). The final modified VE resulted in a constant latency of 8.5ms for a simple

~100 polygon test environment and 13ms for a more realistic ~35k test environment, without taking into account the refresh (frame) rate latency. An important fact is that both of these measurements were in the 8-20ms range of perceptual tolerance for latency in a head tracked HMD-based VE system, however,

2.6. Memory awareness states and schemata

While previous background knowledge in this chapter introduces the basic principles of computer graphics and provides technical information relating to the complexities of IVE generation, this section provides background information regarding the memory schema and awareness states theory employed in this thesis. The effect of latency in spatial cognition in immersive simulations has inspired numerous studies carried out in real and Virtual Environments (VE). VEs are now becoming an increasingly popular alternative approach for research exploration spatial cognition. Moreover, simulation fidelity is based on simulation of spatial awareness as in the real world. This study includes an experiment exploring whether the effect of latency on spatial cognition, memory performance and spatial awareness severely hampers spatial awareness in an Immersive Virtual Environment (IVE) or whether humans are good at perceiving 3D space while using the memory awareness methodology detailed below, irrespectively of relatively high latency.

2.6.1. Memory and Perception

Human Memory is a system for storing and retrieving information acquired through our senses (Baddeley 1977), (Riesberg 1997). The briefest memory store lasts for only a fraction of a second. Such sensory memories are perhaps best considered as an integral part of the process of perceiving. Both vision and hearing, for instance, appear to have a temporary storage stage, which could be termed short-term auditory or visual memory and that could last for a few seconds. In addition to these, though, humans clearly retain long-term memory for sights and sounds. Similar systems exist in the case of other senses such as smell, taste and touch. In this section, the memory awareness methodology and memory schema theories employed in this thesis are analyzed.

2.6.2. The remember/know paradigm

In this section, the main methodology employed in this thesis is going to be analyzed. This methodology forms the core of the experimental design presented in 0.

In the process of acquiring a new knowledge domain, visual or non-visual, information retained is open to a number of different states. Accurate recognition memory can be supported by: a specific recollection of a mental image or prior experience (remembering); reliance on a general sense of knowing with little or no recollection of the source of this sense (knowing); strong familiarity rather than an un-informed guess (familiar); and guesses. 'Remembering' has been further defined as 'personal experiences of the past' that are recreated mentally (Gardiner and Richardson-Klavenhn 1992). Meanwhile 'knowing' refers to 'other experiences of the past but without the sense of reliving it mentally'. (Tulving 1992) provided the first demonstration that these responses can be made in a memory test, item by item out of a set of memory recall questions, to report awareness states as well. He reported illustrative experiments in which participants were instructed to report their states of awareness at the time they recalled or recognized words they had previously encountered in a study list. If they remembered what they experienced at the time they encountered the word, they made a 'remember' response. If they were aware they had encountered the word in the study list but did not remember anything they experienced at that time, they expressed a 'know' response. The results indicated that participants could quite easily distinguish between experiences of remembering and knowing. These distinctions provide researchers a unique window into the different subjective experiences an individual has of their memories.

Measures of the accuracy of memory can therefore be enhanced by self-report of states of awareness such as 'remember', 'know', 'familiar' and 'guess' during recognition (Conway, et al. 1997), (Brandt, Gardiner and MacRae 2006). Object recognition studies in VE simulations have demonstrated that low interaction fidelity interfaces, such as the use of a mouse compared to head tracking, as well as low

visual fidelity, such as flat-shaded rendering compared to radiosity rendering, resulted in a higher proportion of correct memories that are associated with those vivid visual experiences of a 'remember' awareness state (Mania, Troscianko, et al. 2003), (Mania, Wooldridge, et al. 2006), (Mania, Badariah and Coxon 2010). As a result of these studies, a tentative claim was made that those immersive environments that are distinctive because of their variation from 'real' representing low interaction or visual fidelity recruit more attentional resources. This additional attentional processing may bring about a change in participants' subjective experiences of 'remembering' when they later recall the environment, leading to more vivid mental experiences. The present research builds upon this pattern of results and its possible explanations.

Whilst researchers may be interested in measuring differences between the memorial experiences of remembering and knowing, there is recent evidence to suggest that how this is implemented in a practical sense can influence the accuracy of our measures of these. Specifically, the instructions and terminology influence the accuracy of participants' remember-know judgments (McCabe and Geraci 2009). In the past, there have been concerns raised about the use of the terms 'remember' and 'know' because the meaning that participants attach to these terms may be slightly different to those intended by the researchers. In clinical populations this has been a particular concern, and several researchers have replaced the terms 'remember' and 'know' with those of 'type a' and 'type b' (e.g. (Levine, et al. 1998); (Wheeler and Stuss 2003)). Recent evidence has suggested that these changes are also beneficial when measuring 'remember' and 'know' judgments in non-clinical populations (McCabe and Geraci 2009). Participants are generally more accurate, in that there are less false-alarms, when 'remember' and 'know' are replaced with the terms 'type a' and 'type b' in any instructions given. This procedure was therefore followed here.

Moreover, it has been shown that memory performance is frequently influenced by context-based expectations (or 'schemas') which aid retrieval of information in a memory task (Minsky 1975). A schema can be defined as a model of the world based on past experience which can be used as a basis of remembering events and provides

a framework for retrieving specific facts. In terms of real world scenes, schemas represent the general context of a scene such as 'office', 'theatre' etc. and facilitates memory for the objects in a given context according to their general association with that schema in place. Previously formed schemas may determine in a new, but similar environment, which objects are looked at and encoded into memory (e.g., fixation time). They also guide the retrieval process and determine what information is to be communicated at output (Brewer and Treyens 1981).

(Pichert and Anderson 1966) schema model predicts better memory performance for schema consistent items, e.g. items that are likely to be found in a given environment, claiming that in-consistent items are mostly ignored. Contrarily, the dynamic memory model (Holingworth and Henderson 1998) suggests that schema-inconsistent information for a recently-encountered episodic event will be easily accessible and, therefore, leads to better memory performance. Previous VE experiments revealed that schema consistent elements of VE scenes were more likely to be recognized than inconsistent information (Mourkoussis, et al. 2010), (Mania, Robinson and Brandt 2005), supporting the broad theoretical position of (Pichert and Anderson 1966). Such information has led to the development of a selective rendering framework. In this experimental framework, scene elements which are expected to be found in a VE scene may be rendered in lower quality, in terms of polygon count thereby reducing computational complexity without affecting object memory (Zotos, Mania and Mourkoussis 2009).

The experimental framework presented here and tested through limited pilot studies aims to investigate the specific effects of tracking delay on both the accuracy and the phenomenological aspects of object memories acquired in a VE. When adopted for full-scale experimentation, it is of interest to identify whether the presence of added tracking latency applied to a system of minimum tracking latency is associated with the stronger vivid visually induced recollections that have previously been demonstrated with lower interaction or visual fidelity [Mania et al. 2010]. A secondary goal is to investigate the potentially positive effect of schemas on object recognition tasks post-VE exposure.

2.5.4 Effect of Latency on Spatial Awareness

The utility of certain VEs for training such as flight simulators is predicated upon the accuracy of the spatial representation formed in the VE. Spatial memory tasks, therefore, are often incorporated in benchmarking processes when assessing the fidelity of a VE simulation. Spatial awareness is significant for human performance efficiency of such tasks as they require spatial knowledge of an environment. A central research issue therefore for real-time VE applications for training is how participants mentally represent an interactive computer graphics world and how their recognition and memory of such worlds correspond to real world conditions.

The experimental methodology presented focuses upon exploring the effect of head-tracking latency on object-location recognition memory and its associated awareness states while immersed in a radiosity-rendered synthetic simulation of a complex scene. The space was populated by objects consistent as well as inconsistent with each zone's context, displayed on a head-tracked, stereo-capable HMD. The main premise of the spatial awareness methodologies that memory performance is an imperfect reflection of the cognitive activity that underlies performance on memory tasks.

Although previous research identifies the detrimental effect of tracking latency on mainly motor task performance, it could be true that for generic spatial awareness tasks such as navigation or visual search, the effect of latency is less severe and humans adapt to it while forming a detailed mental map of the space. In this thesis, we also explore whether the effect of latency on spatial cognition, memory performance and spatial awareness severely hampers spatial awareness in an Immersive Virtual Environment (IVE) or whether humans are good at perceiving 3D space while using the memory awareness methodology detailed below, irrespectively of relatively high latency.

The technical implementation of this thesis includes:

- The development of a mechanism to measure end-to-end head tracking latency.
- The software rearrangements for controlling and minimizing latency as well adding constant amounts of latency.
- The development of an experimental 3D scene to be displayed on a stereo-capable high-end HMD in order to conduct a formal experiment investigating the effect of latency on spatial awareness states

Technical details as well as experiment data are presented in the following chapters.

2.7. Chapter Summary

This chapter introduced a set of fundamental terms of computer graphics starting with defining light energy and its properties and light propagation. Subsequently, computer graphics illumination models were analyzed. The following sections were focused on visual perception and its application to computer graphics rendering.

Furthermore, this chapter illustrated the complex variety of tools and equipment required to create, view and interact with immersive virtual environments. It provided background information regarding the key technologies used in this study and an overview of the technologies necessary to display and interact with immersive virtual environments.

Most importantly, this chapter presented the shortcoming of previous latency measurement approaches. In this thesis we present a low cost portable latency measurement approach which extends previous techniques and is also used to evaluate the cognitive impact of head tracking latency in immersive simulations.

We will proceed with analyzing the VE application framework used in this thesis.

CHAPTER 3. The Virtual Environment Application

In this section we will analyze the technical framework of the XVR Development Environment utilized for the implementation of the VE application used to measure and minimize tracking latency as well as assess the cognitive impact of end-to-end head tracking latency in immersive simulations.

3.1. XVR Overview

XVR is technology for the rapid development of Virtual Reality applications. Using a modular architecture and a VR-oriented scripting language, XVR content can be embedded on a variety of container application and making it suitable to write content ranging from web-oriented presentation to more complex VR installations. Originally created for the development of web-enabled virtual reality applications, XVR has evolved in the recent years to an all-around technology for interactive applications. Beside web3d content management, XVR supports a wide range of VR devices (such as trackers, 3d mice, motion capture devices, stereo projection systems and HMDs) and uses a state-of-the-art graphics engine for the real-time visualization of complex three-dimensional models that is perfectly adequate even for advanced off-line VR installations. XVR applications are developed using a dedicated scripting language whose constructs and commands are targeted to VR, and give the possibility to developers to deal with 3D animation, positional sounds effect, audio and video streaming and user interaction. In its current form XVR is an ActiveX component running on the various Windows platforms, and can be embedded in several container applications including the web browser Internet Explorer.

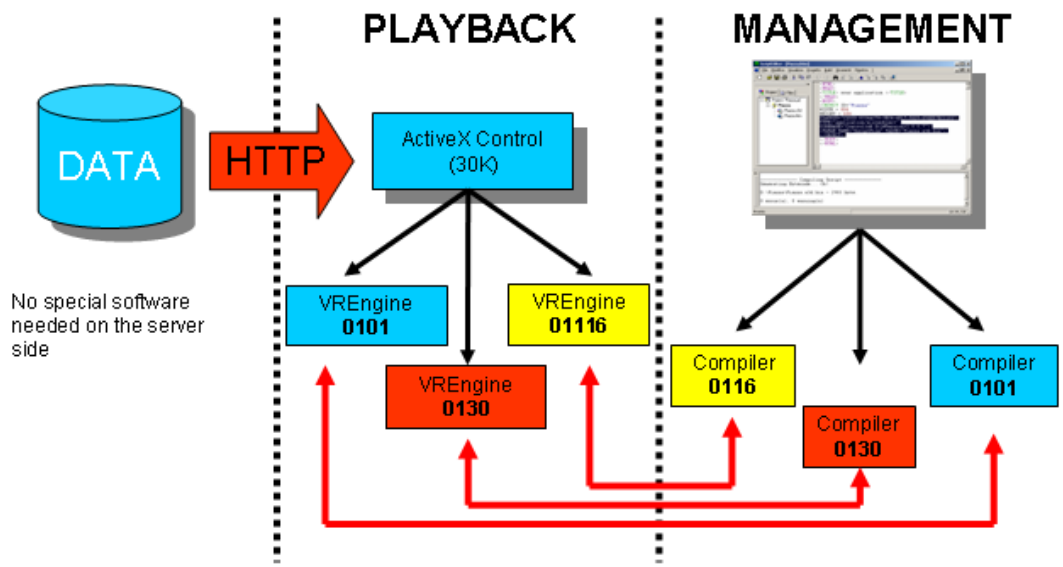


Figure 41: XVR Data Flow

XVR (Figure 41) is actually divided in two main modules: the ActiveX Control module, which hosts the very basic components of the technology, such as the versioning check and the plug-in interfaces, and the XVR Virtual Machine (VM) module, which contains the core of the technology, such as the 3d Graphics engine, the Multimedia engine and all the software modules managing the other built-in XVR features. It is also possible to load additional modules which offer advanced functionalities, like the support to VR devices, as we decided to keep them separated so that web applications, which usually do not need any of these advanced features, are not afflicted by additional downloading times. The XVR-VM, like many other Virtual Machines, contains a set of bytecode instructions, a set of registers, a stack and an area for storing methods. The XVR Scripting Language (S3D) allows specifying the behavior of the application, providing the basic language functionalities and the VR-related methods, available as functions or classes. The script is then compiled in a bytecode which is processed and executed by the XVR-VM.

When accessing a web page hosting an XVR application, after checking the version and, if needed, downloading the right version of the VM module, the

bytecode of the application is downloaded and, subsequently, so do all the data files related to the application, such as 3d models, textures, sounds etc. After the downloading phase the bytecode is executed: first, all the necessary initializations are performed, then the proper application starts. The modularity of XVR allow to easily adapt it both for web and for stand-alone applications, according to the specific needs, as it be considered as made of a central core (the XVR-VM) and several additional modules, the main being the ActiveX Control for the web access. In general an XVR program can be represented as a main loop which integrates several loops, each one running at its own frequency, such as graphics, physics, networking, tracking, and even haptics, at least for the hi-level control loop (Figure 42).

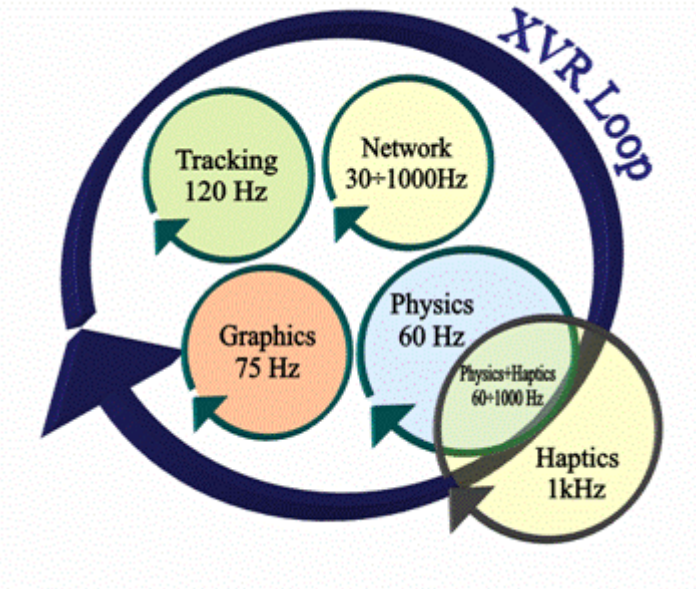


Figure 42: XVR Loop

3.2. S3D: The XVR Scripting Language

In general an XVR program is always based on 6 fundamental functions. These predefined functions constitute the basis of any project:

- *OnDownload()*

  The *OnDownload ()* function is performed at the very beginning and triggers the download of the data files needed from the application.

- *OnInit()*

The *OnInit ()* function is the place where to put the initialization code for the app. All the commands are executed sequentially. All the other functions are not active until *OnInit* () completes its execution.

- *OnFrame()*

  The *OnFrame ()* is the place for functions and methods that produce graphics output. This is the only function where the graphics context is visible. Placing graphics command outside this function would produce no results.

- *OnTimer()*

  The *OnTimer ()* function runs independently (i.e. at a different rate) by *OnFrame ()* and it's where to put commands that must be independent from the rendering task. As the timer is hi-res, it is possible to setup some parameters so that this function can be called up to 1k times per second.

- *OnEvent ()*

  The *OnEvent ()* function is independent from both *OnTimer()* and *OnFrame()*. It gets called whenever the application receives an event message. Event messages can be external (i.e. some Windows messages) or internal (i.e. generated anywhere in the XVR program). Events and messages are supported in XVR because they add flexibility to the programming environment for task where fixed timers are not the best option. If the application does not need them, this function can be ignored.

- *OnExit()*

  The *OnExit ()* function is called when the application quits or when the user close the page the application is in.

Beyond the basic functions, XVR offers lots of predefined classes, functions and data structures, and the user has the possibility to define new ones.

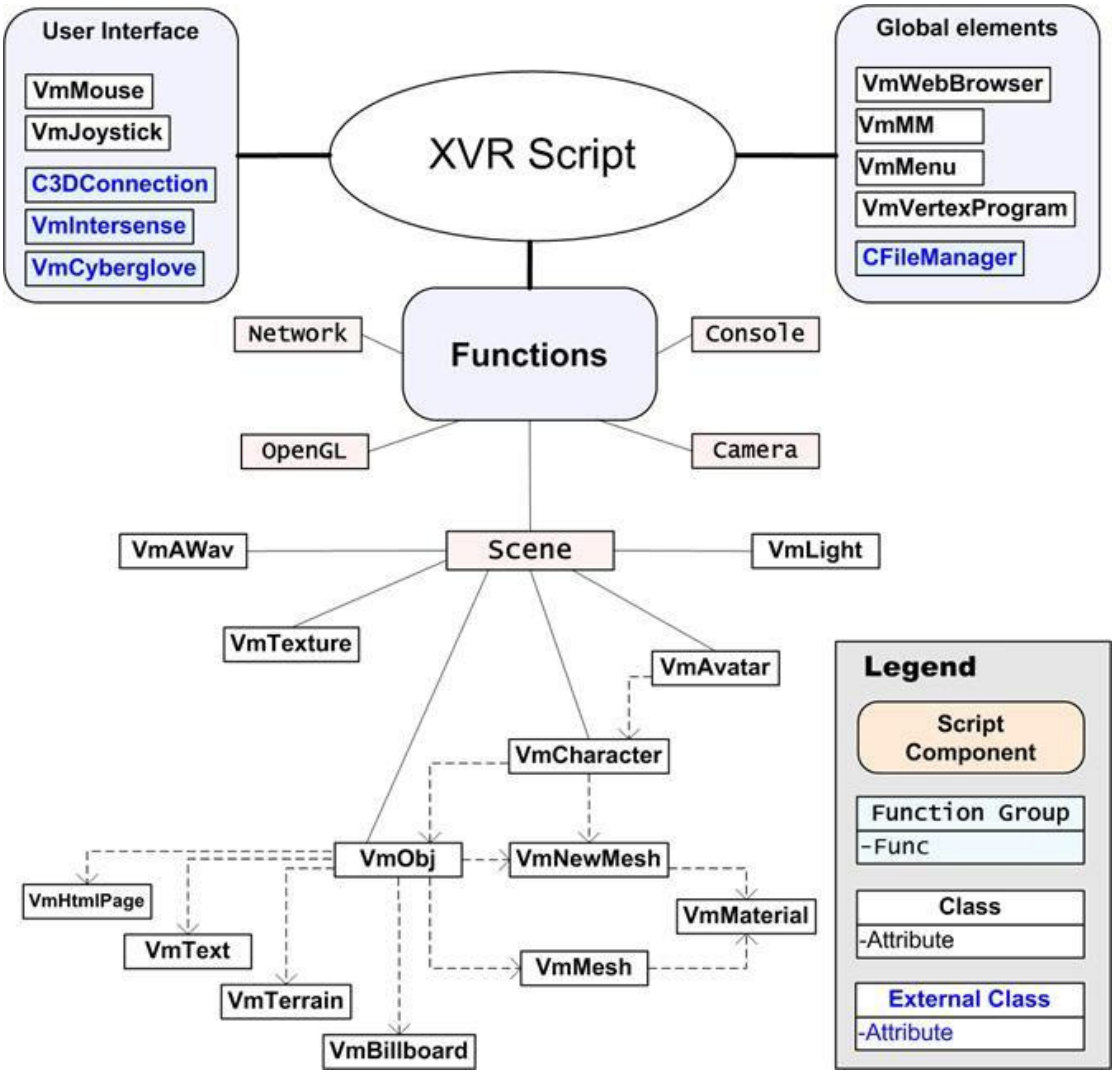


Figure 43: XVR classes and functions

3.2.1. Classes

The main set of classes and functions is related to the 3D graphics environment management. The Scene and Camera functions allow to, respectively, setup the graphical scene and the viewpoint properties. The latter functionalities are available also by means of the *CVmCamera* class.

The *VmLight* classes provide the functionalities related to lighting. The 3d models are managed by the *VmMesh* class, which handles the geometric properties of the model, and by the *VmObject* class, which deals with reference systems and geometrical transformations. Two superclasses based on the *VmObject* class, the *VmAvatar* and the *VmCharacter*, allow manipulating complex hierarchies of objects,

respectively with and without the support of real-time geometry deformation. The appearance of the objects is managed by the *VmMaterial* and *VmTexture* classes. Other graphics related classes are the *VmBillboard*, *VmText* and *VmTerrain* which deal with more specific components.

The integrated 3D engine, built on top of OpenGL, allows to manage the visual output not only on a standard graphical window (either web or local hosted), but also on more advanced devices such as Stereo Projection Systems and Head Mounted Displays. The perspective tuning for stereoscopy is supported and it is relatively simple to port applications from and to different visualization hardware. The engine uses state of the art algorithms of culling, simplification, normal mapping and image caching to achieve good real-time performances even with high-complexity models.

Although the S3D scripting language offers high-level functionalities to manage the 3d contents, it also allows mixing low-level functionalities to realize special effects or personal implementations not directly supported by the language. This is achieved through an almost complete wrapping of OpenGL functions that can be put straight in an S3D script. Moreover, XVR supports an even lower level of programming by the insertion of vertex programs to directly manipulate the graphics pipeline, a feature commonly available on the current 3d graphics boards. As in any other environment which allows mixing different programming levels, a particular attention must be put when using the lo-level functionalities, which can heavily modify the application graphical status and its flow.

Another set of classes is devoted to user interaction and communication. Among the built-in classes, the most commonly used are the *VmMouse* and the *VmJoystick* which, together with a set of keyboard related functions, handle the basic interaction with the application. The available additional external modules allow controlling also advanced interfaces, such as sensorized data gloves and body suits, magnetic, inertial and ultrasonic trackers, and 3d mice.

Network communication is handled by means of IP functions, which offer TCP connection-oriented or connectionless UDP functionalities, and html-related Data

functions, which allow the hosting web page to communicate with the XVR application so that they can reciprocally interact and modify their status. This means that commands can be sent to the application from the html page, which can therefore constitute the primary GUI for the application, and messages can be sent from the application to the html page, which can change its aspect or open popup windows etc.

A bi-dimensional graphical overlay output is provided by the Console functions. The GUI functions are completed by the *VmMenu* and *VmFileManager* classes whose functionalities can be trivially deduced.

Finally XVR offers a set of multimedia related classes. Among these, the *CVmMidi*, *CVmAvi* and *CVmMp3* add the support to play background music, speech and movies. The *CVmAWav* provides the 3d sound functionalities, giving the possibility to dynamically create sound sources and to position them in the 3d space. XVR supports also remotely located sound servers allowing the use of more than one planar 3d audio systems in order to realize a complete positional audio system able to localize the acoustical source not only in the plane surrounding the user, but rather in the whole space adding a better perception of the up-down directionality. Other functions exist, implementing various functionalities related to files, math and strings manipulation.

3.2.2. Functions

XVR offers a wide set of functions to manage several features of its environment. Most of them are related to the graphical scene, and often refer to global attributes which do not need to be accessed through dedicated classes.

Scene functions

These functions allow defining the graphical scene and its parameters. A scene is basically a graphical loop; it is delimitated by the *SceneBegin()* and *SceneEnd()* commands. All the graphical commands (including OpenGL functions) must be called inside a *SceneBegin()...SceneEnd()* block, or they will not produce any visible

effect. The scene is drawn according to the current camera settings, which are read upon calling *SceneBegin()* and remain constant until *SceneEnd()*. Therefore any change in camera properties is effective only in the next scene loop. The very basic properties (FOV, near plane Z, far plane Z) are assigned using the SET command. Additional properties may be configurable through scene functions. With the scene functions is also possible to realize split screens and stereoscopic visualization.

Camera Functions

With these functions it is possible to manage the camera (i.e. the viewpoint) and its properties. The camera setup affects how the frame image of the scene is produced. Up to 8 different cameras may be used, but only one at each time. The available functions allow to position and rotate the camera, to fix a specific target or orientation, to read camera movements from an external animation file and to retrieve its main properties.

OpenGL Functions

This set of functions constitutes a wrapping of several OpenGL functions which provide a low-level graphical programming layer, which can be mixed with the high-level XVR graphical functionalities. This allows realizing special «effects» not directly implemented in XVR or using particular modalities by expert programmers. The wrapper also adds polymorphism to OpenGL functions, providing only one version of each function instead of having different ones depending on the parameters type.

Texture Functions

These functions allow specifying some global parameters of the texture environment.

Console Functions

Console functions allow visualizing overlayed 2D text using system fonts. In addition to text, console functions offer also basic 2d drawing functionalities and support transparency. The console layer is drawn independently from the 3D scene,

and thus console functions may be inserted anywhere in the code (i.e. not necessarily inside a *SceneBegin()…SceneEnd()* block).

Network Functions

Different instances of XVR applications may communicate between them, or with an external host, making use of the XVR networking functions. It is possible to use both connectionless UDP and connection-oriented TCP protocols.

I/O Functions

I/O functions manage the communication with external devices (keyboard or serial devices) and with the HTML page hosting the application. The I/O with the HTML page is achieved by means of two functions: *DataIn()* (from HTML page to XVR) and *DataOut()* (from XVR to HTML page). In the HTML page the interaction must be opportunely managed by means of HTML-related scripting features (using JavaScript, VBScript etc.).

File Functions

This group of functions offers an interface to the file system and allows managing the download of data files from the network. It is possible to control the downloading status in order to implement asynchronous downloads.

Strings and Text Functions

This set of functions deal with strings manipulation/parsing, and with text output.

Math Functions

XVR, in addition to the basic mathematical functions natively available in the language as commands, provides a set of additional mathematical functions which also deal with matrices and vectors.

Timer Functions

This set of functions deal with the timer functionalities. XVR allow specifying a high resolution timer (up to 1ms).

Global settings Functions

These functions allow specifying some parameters of the global XVR environment, such as the Frame Rate, the Timer Resolution, the Cursor Shape, the Audio 3d properties, or more specifically regarding the global graphical environment, such as the Ambient Light, the Fade properties, the Background etc.

Other Functions

Various remaining XVR functions not grouped at the categories above.

3.3. XVR Development Studio

XVR Developer Studio is an Integrated Development Environment which allows the programmer to create and manage XVR projects, to author the script code, to compile the script in a byte code to be interpreted by the XVR run-time Virtual Machine and to execute this code either in an HTML test page or via the XVRGlut application.

The IDE includes also a debugger. The IDE provides a wizard dialog. The dialog allows creating new resources through the wizard process. XVR Studio Developer inherits the concept of workspace from Eclipse.

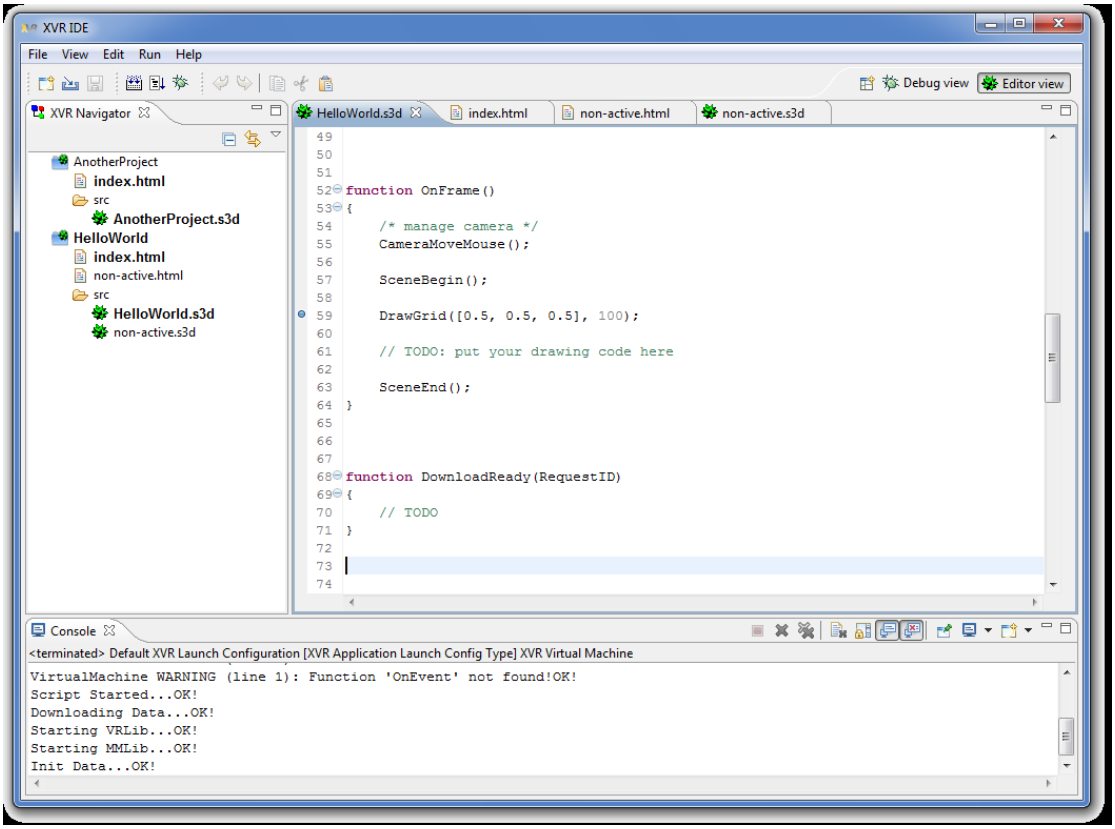


Figure 44: The XVE Development Studio IDE

3.3.1. Wizards

Using the new wizard command you can access the wizard dialog. The dialog allows creating new resources through the wizard process. The wizard is accessible both from the file menu and the main toolbar. XVR Developer Studio provides several wizards used to create different kinds of resources.

As depicted in the Figure 45 actually there are five different wizard categories:

- General: used to create empty project, any type of file and folders
- OpenGL Shader files: used to create shaders
- SVN: used to get resources directly from an SVN repository
- XVR Project Wizard: used to create new XVR projects
- Other: other wizard, in particular the category allows to create CSS and HTML resources

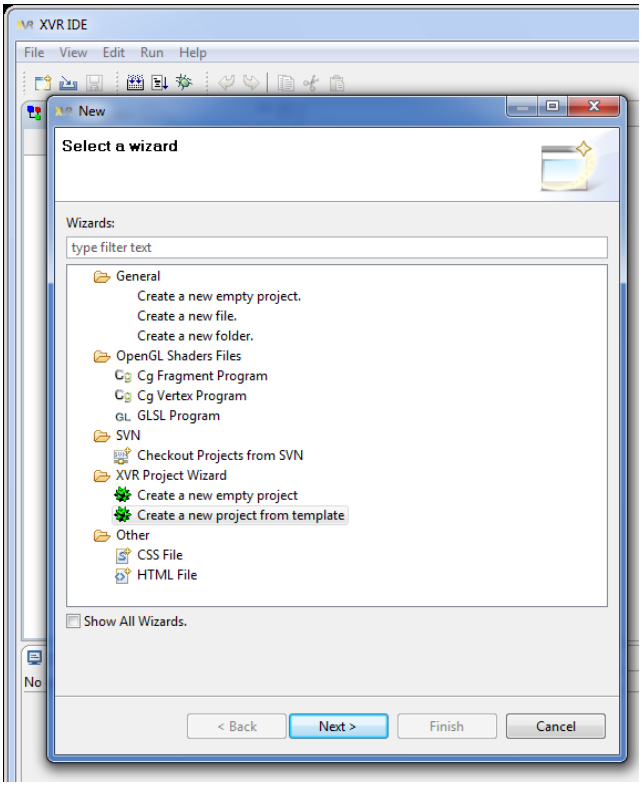


Figure 45: The XVR wizards.

XVR Project Wizard

The XVR project wizard category provides two different entries. The first one creates a new XVR empty project. The users have to specify the name for the new project and the folder where the project will be created.

The other wizard allows a user to create a new XVR projects from templates. As the former, the wizard requires the user to enter a name for the project and the directory where the project has to be created. Furthermore the user has to choose between six different templates.

An error message will be displayed on top of the dialog if one or more parameters entered are not valid. The user can not finish the wizard procedure until all the entries are valid.

Once finished the wizard creation process, the new XVR project will be available in the navigator view.

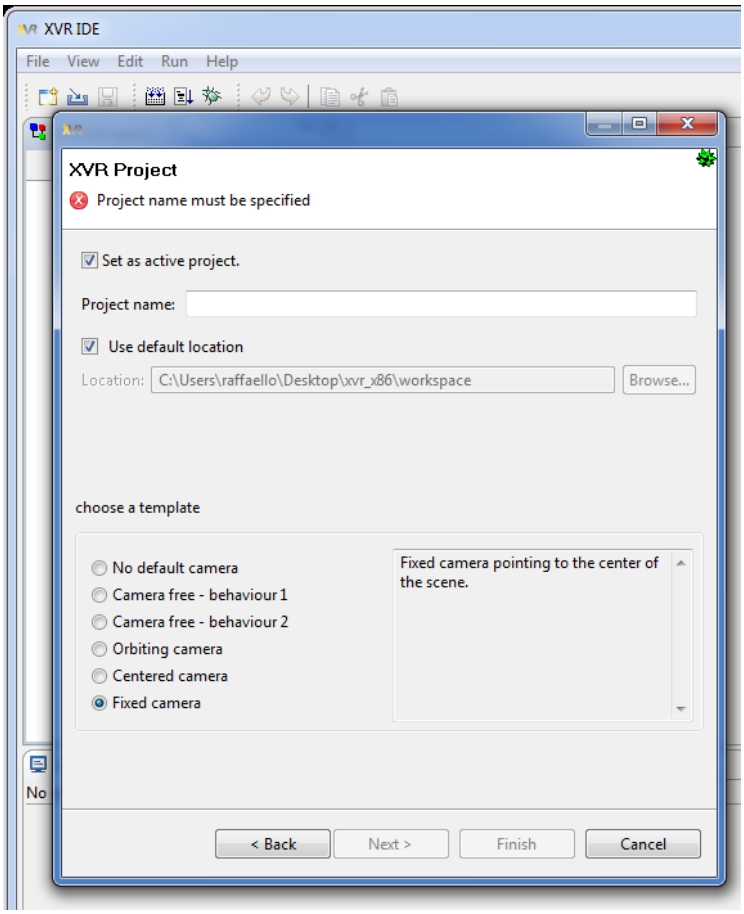


Figure 46: The XVR project wizard

3.3.2. Perspectives

A perspective is a visual container for a set of views and editors (parts). These parts exist wholly within the perspective and are not shared. A perspective is also like a page within a book. It exists within a window along with any number of other perspectives and, like a page within a book, only one perspective is visible at any time.

XVR Developer Studio has two default perspectives. The main perspective is called Editor Perspective. When you start XVR Developer Studio, the editor perspective is used by default.

Editor Perspective

The editor perspective is composed by:

- a navigator view on the left
- an outline view on the left
- a console view on the bottom
- one or more editor in the center

Debug Perspective

When a project is ran in debug mode, once the first breakpoint is reached, the editor asks to the user if he would like to switch to the debug perspective view. Once the debug session ends the IDE automatically switch the perspective back to the Editor Perspective. The user can also switch the perspective using the perspective toolbar located in the upper right corner of the editor.

In the side picture the switch perspective dialog is shown

The debug perspective has been designed to make easier debugging. It provides the following views more than those provided by the editor perspective:

- an expression view on the right
- a variables view on the right
- a debug view on the bottom

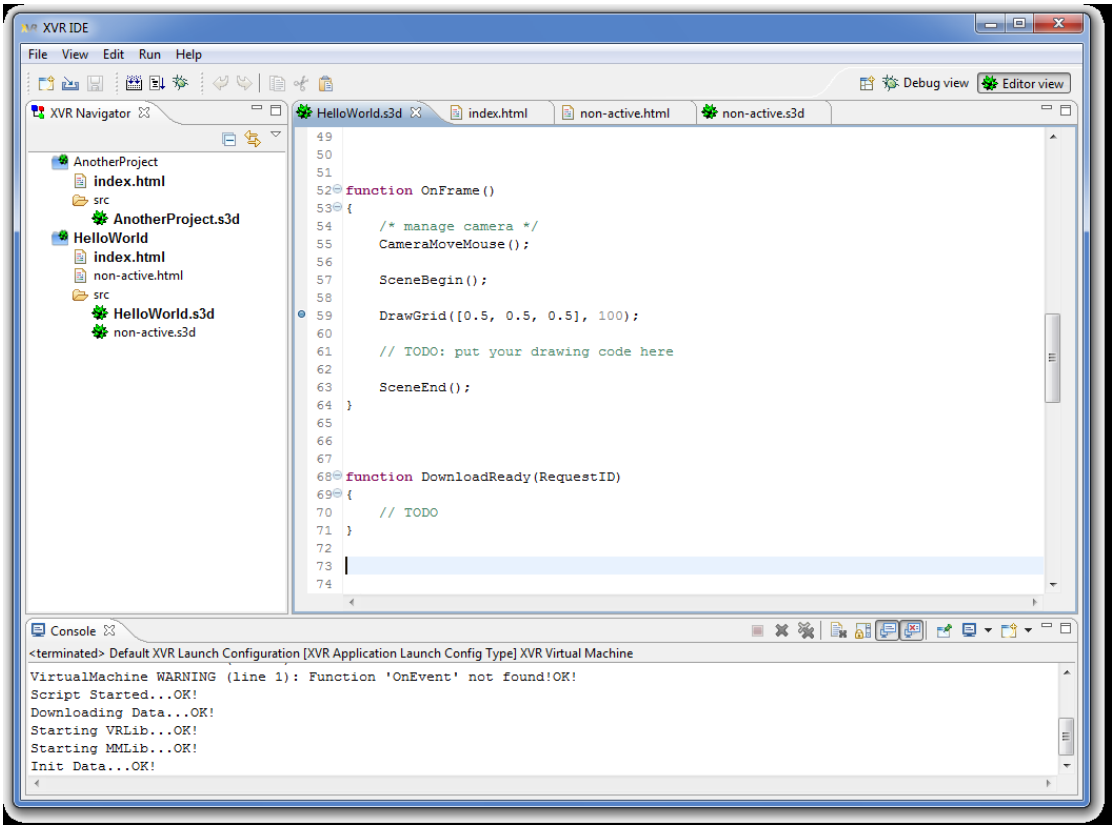


Figure 47: Editor perspective

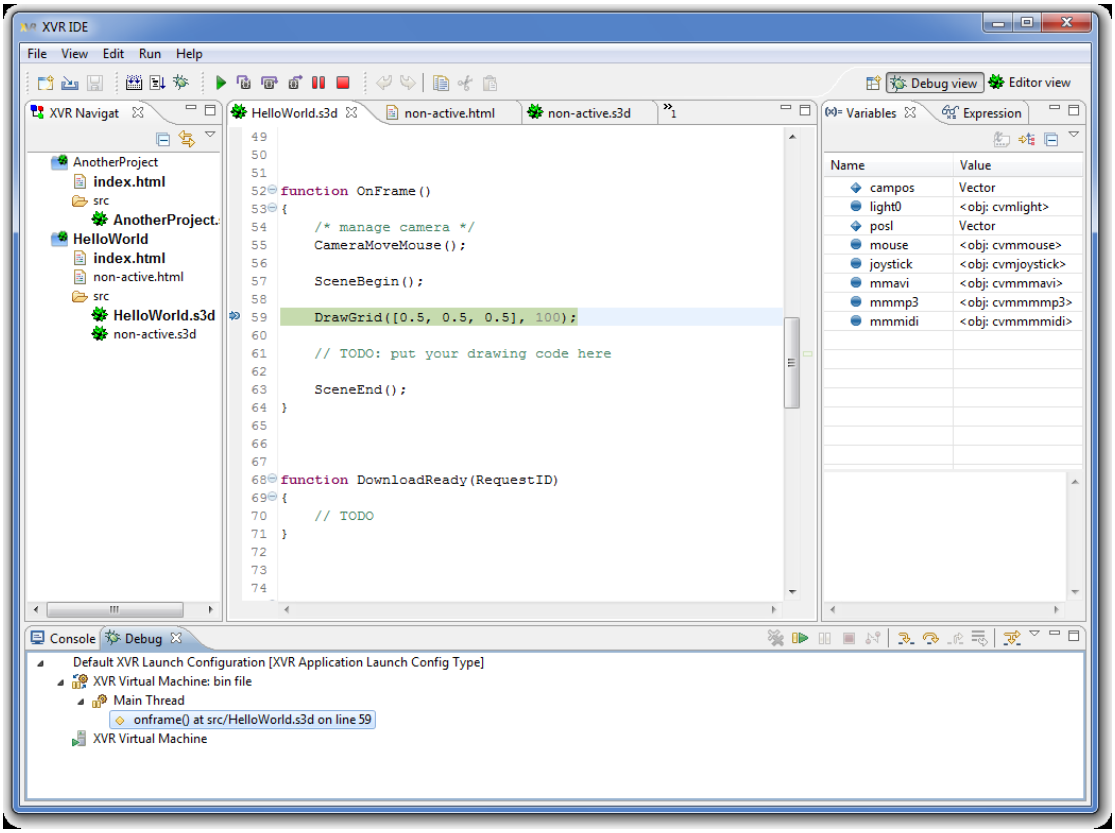


Figure 48: Debug perspective

3.3.3. Menus

File menu

The file menu allows to; *Create* new projects, files, folders, etc.., using wizards, Import external projects, *Save* the active editor and *Exit* the application.

View Menu

The view menu allows to:

- Select a view to display (e.g. Outline and Navigator). If the view has been previously closed it will be reopened, otherwise it will just focused.
- display the editor preferences dialog(see IDE preferences)

Edit menu

The edit menu allows to; Undo/redo the last/previous edit action performed in the active editor, Copy, cut and paste, Search and replace text in all the files contained in the workspace, Search and replace text in the selected editor.

Run menu

The run menu allows to:

- Build the active project. The result of the build process will be printed on the console view (see Console view)
- Run the active project. Using this command the project will be first compiled
- run the active project without build it before
- Run the active project in debug mode. Using this command the project will be first compiled.

Help menu

The help menu allows to; Display help and Display the XVR Developer Studio project "about".

3.3.4. Views

An editor is typically used to edit or browse a document or input object. Modifications made in an editor follow an open-save-close lifecycle model. A view is typically used to navigate a hierarchy of information, open an editor, or display properties for the active editor. In contrast to an editor, modifications made in a view are saved immediately. The layout of editors and views within a page is controlled by the active perspective.

Console view

The console view can contains one or more different console. A console allows writing and displaying to the user what happens during his work. When the IDE is started there are not console, but each console will be created when the editor needs to write contents to it.

XVR Studio developer uses two different consoles:

- The first one, called XVR CONSOLE displays the result of the last project build
- The other one displays the output coming from the last run.

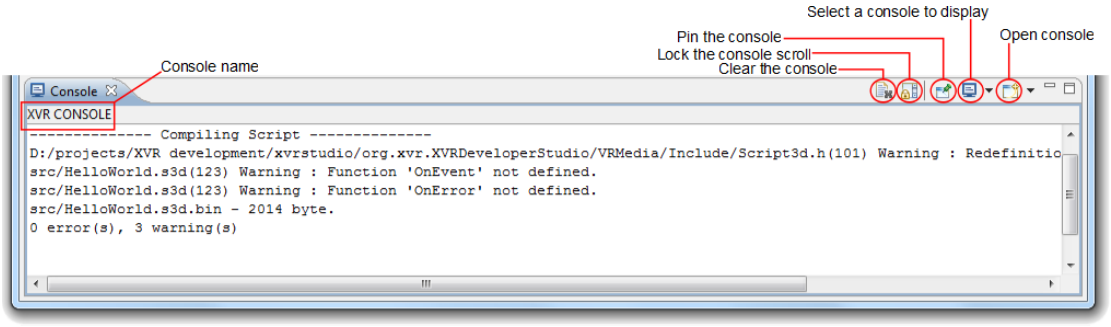


Figure 49: XVR console

Navigator view

The navigator view displays all the projects contained in the workspace. Each project corresponds to the root of a tree that reflects the resources contained in the project.

If you double-click on a folder, the tree node is expanded/collapsed to display/hide its content. If you double-click on a leaf of the tree the IDE will open it with an editor selected on the basis of the file extension.

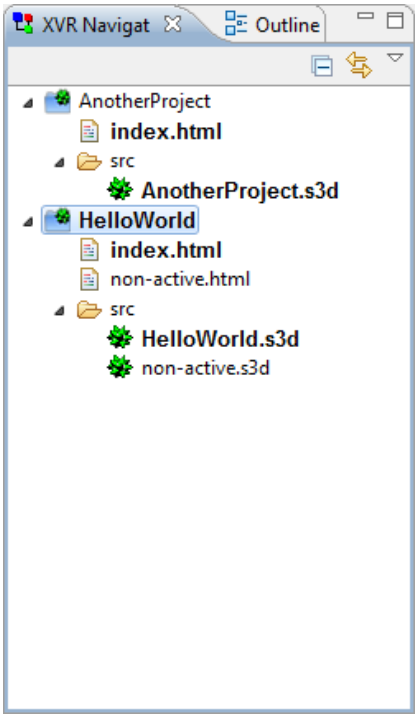


Figure 50: Navigator view

The navigator has a contextual menu accessible by right-clicking on it which allows performing several actions. Among the others it allows to:

- if you click on the root of a project or on a resource inside of it, to set the active project
- if you click on an HTML file, to set the active HTML file which will be used when launching a run on Internet Explorer
- if you click on an S3D file, to set the active S3D file which will be used as the main entry point of the application while the project is compiled

Outline view

The outline view allows to inspect the active document in the editor listing the elements of which is composed by using a tree like structure. The elements displayed depend on which type of document you are outlining.

Debug view

The debug view displays the running process launched from the editors. When the process is blocked on a breakpoint, the view displays on which breakpoint the

execution is blocked. The view allows also controlling the process flow with a set of buttons placed in the toolbar on the top right corner of the view.

Variables view

Once the process is blocked on a breakpoint, the variables view displays all the variables available in the scope and allows to inspect the value of these.

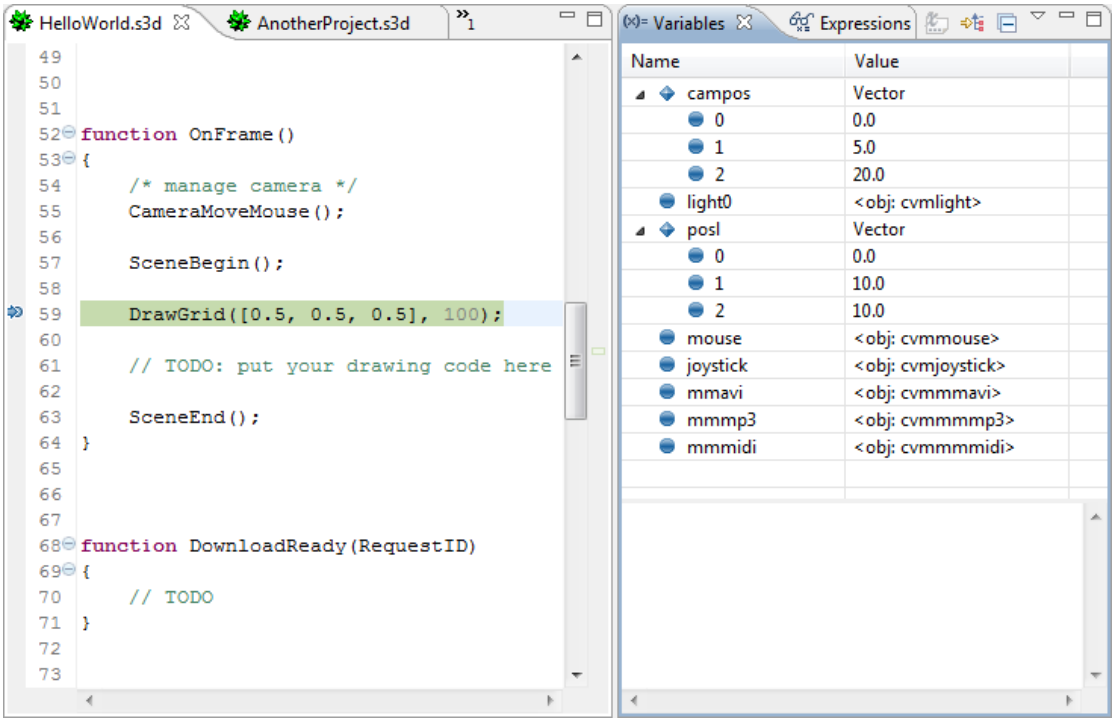


Figure 51: Variables view

Expressions view

Once the process is blocked on a breakpoint, the expressions view allows executing:

- Algebraic expressions using also the values available in the scope.
- External functions. If the editors fails to execute the function or it is not allowed to execute such a function an error message is displayed as returned value. Actually only the function glGet is allowed and you have to specify the integer value of the OpenGL constant you want to query the value of.

3.4. XVR Browser

XVR can use the Internet Explorer rendering engine in order to render the 3D scene (photo). In order to achieve that, the XVR application needs to be embedded inside a web page (Figure 52).

```
<OBJECT
ID="AppName"
WIDTH = 800
HEIGHT = 800
CLASSID = "CLSID:5D589287-1496-4223-AE64-65FA078B5EAB"
TYPE = "application/x-oleobject" align="left" border="2"
CODEBASE = "http://client.vrmedia.it/XVRPlayer.cab#Version=1,0,0,900">
<PARAM NAME="ScriptName" VALUE=" AppName.s3d.bin">
<PARAM NAME="EngineVersion" VALUE="0136">
<PARAM NAME="BackgroundColor" VALUE="#000066">
<PARAM NAME="EngineParam" VALUE "DEPTH=24">
<PARAM NAME="UserParam" VALUE "…">
</OBJECT>
```

Figure 52: Embedding XVR Application into html page.

These lines will be automatically added if using XVRStudio to generate the HTML page which hosts the XVR application.

The parameters which can be specified by the user are:

Param name	Value - Meaning	Value - Format
`ScriptName`	The name of the .bin file containing the XVR application bytecode.	String ("Filename.bin")
`EngineVersion`	The number of the XVR Engine version needed to execute the specified bytecode.	4 numeric characters ("0140")
`BackgroundColor`	The default background color of the XVR 3d graphical context. The format is the same of the HTML.	Same as in HTML ("#RRGGBB", hex format)

ForceUpdate	Forces the download of the specified version of the XVR Engine without checking the local version.	None ("")
UserParam	A string containing one of more user-defined parameters which will be passed to the OnDownload () and OnInit () functions.	String ("Myparams 0 1 2")
EngineParam	A string containing one of more parameters which will be used to initialize the engine. The supported parameters are:	String ("DEPTH=24;STEREO")
	NOLOGO	Hides the rotating cube logo during the download phase.
	STEREO	Enables, if available, the support to the Quadbuffer Stereo OpenGL mode.
	DEPTH=nn	Sets the depth of the DEPTH buffer to nn bits. Depending on the graphics board, the maximum depth may be 16, 24, 32.
	STENCIL=nn	Activates the Stencil buffer reserving nn bits.
	FOV=nn	Sets the 3d scene Field Of View to nn degrees.

NEAR=nn	Sets the 3d scene Near plane distance to nn.
FAR=nn	Sets the 3d scene Far plane distance to nn.
QUIET	Suppresses warning messages.

3.5. XVR Tracking

XVR supports 3 different types of tracking data reading;

- Reading from a Joystick device
- Reading from an external DLL API
- Reading tracking data VRPN (network) packets

Our InterSense tracker supports all the above methods of reporting tracking data via the InterSense Server interface.

3.5.1. Using the tracker as a VRPN device

VRPN protocol

VRPN (Virtual-Reality Peripheral Network) is a device-independent and network-transparent system for accessing virtual reality peripherals in VR applications. VRPN is a set of classes within a library and a set of servers that are designed to implement a network-transparent interface between application programs and the set of physical devices (tracker, etc.) used in a virtual-reality (VR) system. The idea is to have a PC or other host at each VR station that controls the peripherals (tracker, button device, haptic device, analog inputs, sound, etc.). VRPN provides connections between the application and all of the devices using the appropriate class-of-service for each type of device sharing this link. The application remains unaware of the network topology. Note that it is possible to use VRPN with devices that are directly connected to the machine that the application is running on, either using separate control programs or running all as a single program.

VRPN also provides an abstraction layer that makes all devices of the same base class look the same; for example, all tracking devices look like they are of the type VRPN Tracker. This merely means that all trackers produce the same types of reports. At the same time, it is possible for an application that requires access to specialized features of a certain tracking device (for example, telling a certain type of tracker how often to generate reports), to derive a class that communicates with this type of tracker. If this specialized class were used with a tracker that did not understand how to set its update rate, the specialized commands would be ignored by that tracker. The current system types are Analog, Button, Dial, ForceDevice, Sound, Text, and Tracker. Each of these abstracts a set of semantics for a certain type of device. There are one or more servers for each type of device, and a client-side class to read values from the device and control its operation. It also provides for

- Time-stamping of data
- Clock-synchronizing of clients and servers
- Multiple simultaneous accesses to peripheral devices
- Automatic reconnection of failed servers
- Storage and playback of sessions

The VRPN client (application-side) library has been tested on a various computer systems. There are drivers for a large number of trackers and VR peripheral devices and is supported by the most VE Development applications.

3.5.2. Using the tracker as joystick

The InterSense trackers data can be accessed as data from a Joystick device. A joystick is an input device consisting of a stick that pivots on a base and reports its angle or direction to the device it is controlling.

Joysticks (Figure 53) are often used to control video games, and usually have one or more push-buttons whose state can also be read by the computer. Most joysticks are two-dimensional, having two axes of movement (similar to a mouse), but one

and three-dimensional joysticks do exist. A joystick is generally configured so that moving the stick left or right signals movement along the X axis, and moving it forward (up) or back (down) signals movement along the Y axis. In joysticks that are configured for three-dimensional movement, twisting the stick left (counter-clockwise) or right (clockwise) signals movement along the Z axis. These three axes - X Y and Z - are, in relation to an aircraft or a head tracker, roll, pitch, and yaw.

Figure 53: A joystick pointing device

An analog joystick is a joystick which has continuous states, i.e. returns an angle measure of the movement in any direction in the plane or the space (usually using potentiometers) and a digital joystick gives only on/off signals for four different directions, and mechanically possible combinations (such as up-right, down-left, etc.). Additionally joysticks often have one or more fire buttons, used to trigger some kind of action. These are simple on/off switches. A hat switch is a control on some joysticks. It is also known as a POV (point of view) switch. It allows one to look around in their virtual world, browse menus etc. For example, many flight simulators use it to switch the player's views.

Some joysticks have haptic feedback capability. These are thus active devices, not just input devices. The computer can return a signal to the joystick that causes it to resist the movement with a returning force or make the joystick vibrate.

The XVR VE application can be configured to access the joystick's pitch, roll, and yaw tracking data using the built-in *CVmJoystick* class (Figure 54). This class manages joystick devices, allowing to retrieving data from them. It is also possible to create a virtual joystick which can be used to replicate remote joystick actions.

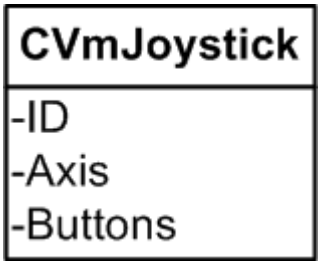


Figure 54: CVmJoystick predefined class

3.5.3. Accessing tracker data directly from the DLL

A Dynamic-link, or DLL, is Microsoft's implementation of the shared library concept in the Microsoft Windows and OS/2 operating systems. These libraries usually have the file extension DLL, OCX (for libraries containing ActiveX controls), or DRV (for legacy system drivers). The file formats for DLLs are the same as for Windows EXE files — that is, Portable Executable (PE) for 32-bit and 64-bit Windows, and New Executable (NE) for 16-bit Windows. As with EXEs, DLLs can contain code, data, and resources, in any combination. DLLs provide a mechanism for shared code and data, allowing a developer of shared code/data to upgrade functionality without requiring applications to be re-linked or re-compiled.

The Intrertrax2 tracker provides a DLL API (Figure 55) for accessing tracker data from C, C++, Visual Basic and other applications supporting readings from DLL libraries.

```
ISD_STATION_DATA_TYPE
```

This data structure is used to return current data for a station, including position, orientation, time stamp, button and analog channel state. It is passed to `ISD_GetTrackingData` as part of `ISD_TRACKING_DATA_TYPE`

```
typedef struct
{
    ISD_STATION_STATE_TYPE Station[ISD_MAX_STATIONS];
}
ISD_TRACKER_DATA_TYPE;



typedef struct
{
    BYTE    TrackingStatus;
    BYTE    NewData;
    BYTE    CommIntegrity;
    BYTE    BatteryState;
    float   Euler[3];
    float   Quaternion[4];
    float   Position[3];
    float   TimeStamp;
    float   StillTime;
    float   BatteryLevel;
    float   CompassYaw;
    Bool    ButtonState[MAX_NUM_BUTTONS];
    short   AnalogData[ISD_MAX_CHANNELS];
    BYTE    AuxInputs[ISD_MAX_AUX_INPUTS];
```

```
    float   AngularVelBodyFrame[3];
    float   AngularVelNavFrame[3];
    float   AccelBodyFrame[3];
    float   AccelNavFrame[3];
    float   VelocityNavFrame[3];
    float   AngularVelRaw[3];
    DWORD   Reserved[64];
}
ISD_STATION_DATA_TYPE;
```

TrackingStatus

Tracking status byte. Available only with IS-900 firmware versions 4.13 and higher, and isense.dll versions 3.54 and higher. It is a value from 0 to 255 that represents tracking quality.

NewData

TRUE if this is new data. Every time ISD_GetData is called this flag is reset.

CommIntegrity

Communication integrity of wireless link.

BatteryState

Wireless devices only 0=n/a, 1=low, 2=ok.

Euler

Orientation in Euler, returned in degrees.

Quaternion

Orientation in Quaternion form.

Position

Station position in meters.

TimeStamp

Only if requested, in seconds.

StillTime

InertiaCube and PC-Tracker products only.

BatteryLevel

Battery Voltage, if available.

CompassYaw

Magnetometer heading, computed based on current orientation.

ButtonState

Only if requested.

AnalogData

Only if requested. Current hardware is limited to 10 channels, only 2 are used. The only device using this is the IS-900 wand that has a built-in analog joystick. Channel 1 is x-axis rotation, channel 2 is y-axis rotation. Values are from 0 to 255, with 127 representing the center.

AuxInputs

Only if requested.

AngularVelBodyFrame

Angular rotation speed in sensor body coordinate frame. This is the processed angular rate, with current biases removed, rad/sec. This is the angular rate used to produce orientation updates.

AngularVelNavFrame

Angular rotation speed in world coordinate frame, with boresight and other transformations applied, rad/sec.

AccelBodyFrame

Acceleration in sensor body coordinate frame, meter^2/sec. Only factory calibration is applied to this data, gravity component is not removed.

AccelNavFrame

Acceleration in the navigation (earth) coordinate frame, meters/sec^2. This is the accelerometer measurements with calibration, and current sensor orientation

applied, and gravity subtracted. This is the best available estimate of acceleration.

VelocityNavFrame

meters/sec, 6-DOF systems only.

AngularVelRaw

Raw gyro output, only factory calibration is applied. Some errors due to temperature dependent gyro bias drift will remain.

Figure 55: Intersense DLL API tracker reading data structure

The XVR VE application supports importing C functions from external DLLs through the build in CVmExternDll class.

When importing the DLL XVR will use the library as an object, and the function will became the method of the object. In order to load the library, the CVmExternDLL object with the name of the library is used.

```
library = CVmExternDLL("myLib.dll");
```

This command creates a CVmExternDll object. This object refers to the library, but now is empty and you have to add method with the __AddFunction(). In case the DLL is not available it generates a fatal error. The DLL is also not available if DLL dependencies (other DLLs) are not available. Since Engine 150 optional second parameter of CVmExternDLL is a Boolean that allows specifying to nicely return a void in case of missing DLL.

This function takes the return value, the name of the exported function, and the type of the parameter.

```
library.__AddFunction(C_FLOAT, "myFunction", C_PFLOAT, C_INT);
```

Now the function can be used like a method of your library.

```
b = library.myFunction(a, 3);
```

The allowed types of parameters are;

C_VOID	void, used only for return value
C_INT	signed 32 bit int
C_FLOAT	32 bit float
C_DOUBLE	64 bit double
C_PCHAR	an array of char
C_PSTR	the same as above
C_PINT	an array of int
C_PFLOAT	a vector of float
C_PDOUBLE	an array of double (Since Engine 150)
C_PFLOAT_1	force the vector of float to be one element long
C_PFLOAT_X	same as above where X is a number from 1 to 16

The previous tables indicate that there is an incompatibility between the data types that the InterSense DLL API provides and the data types than the XVR can read from an external DLL. Thus the API cannot be used as is for tracking data readings (Figure 56).

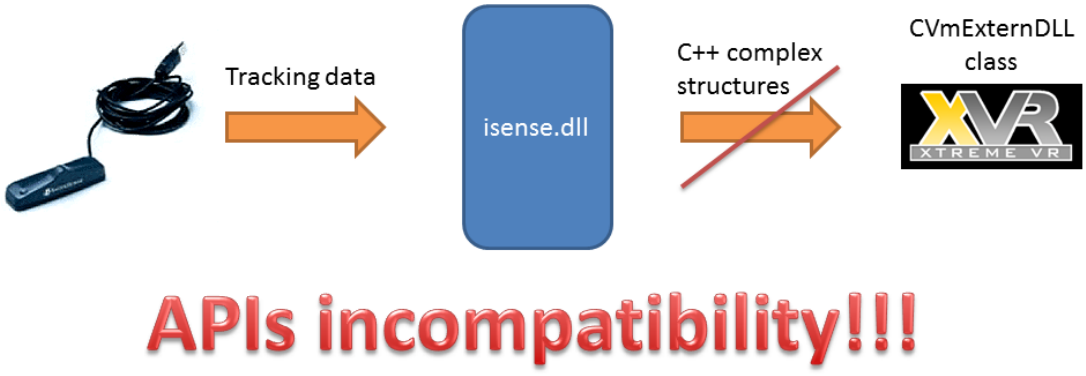


Figure 56: Accessing tracker data directly from the DLL API

3.6. Moving the camera

The viewpoint, also called camera, is managed through this set of functions. Up to eight different camera setups may be handled at the same time, but only one is used to render the current scene. The available functions allow to position and rotate the camera, to fix a specific target or orientation, to read camera movements from an external animation file and to retrieve all these properties.

function TrackerMove()

```
{
        var rScale = -360;
        var move = 0.0;
        static var xpos=0.0 , ypos=0.0, zpos=0.0;
        static var x_pos=0.0, z_pos=0.0;
var cammat = array(16);
        var jx,jy,jz;
        jx=(library.getX()+180)/360;
        jy=(library.getY()+180)/360;
        jz=0.5;
        CameraSetRotation(rScale*jy,[1,0,0]);
        CameraRotate(rScale*jx,[0,1,0]);
        CameraRotate(-rScale*jz,[0,0,1]);
}
```

Figure 57: Rotating the camera with the tracker

3.7. XVR Stereo rendering

3.7.1. Quad buffered Stereo rendering

XVR supports the standard quad buffered OpenGL way to produce stereo images. Using this feature is extremely easy, but you need to have a graphics card that supports quad buffered OpenGL such as the NVIDIA Quadro family or the ATI Fire GL family.

To activate quad buffer support you just need to add this in the .HTML file associated to the project (place it next to the other XVR Control params):

```
<PARAM NAME="EngineParam" VALUE="STEREO">
```

3.7.2. XVR side-by-side stereoscopic rendering

XVR has built in support to adjust the perspective correction needed for stereo visualization. Stereoscopic output in XVR was achieve by using the Side-by-Side stereo rendering and use the Horizontal Span of the graphics card driver to send the 2 different outputs to the 2 VGA channels of the HMD.

Inside the OnFrame function the following code sections were used:

```
SceneBeginRel( 0.0,0.0 , 0.5,1.0 , VR_STEREO_LEFT );  //Rendering  the left half of the screen (left eye)
    Rendering code();
    …
SceneEnd();


SceneBeginRel( 0.5,0.0 , 0.5,1.0 , VR_STEREO_RIGHT); // Rendering  the right half of the screen (right eye)
    Rendering code();
    …
SceneEnd();
```

Figure 58: Stereoscopic side-by-side rendering in XVR

The HMD displays were combined as a single display using the display driver Horizontal span feature. Each monitor of the HMD has a 1024x768 pixels resolution so the resulting combined large screen has a resolution of 2048x768 pixels. Half of the pixels (1024x768) are displayed at each monitor.

DualView vs. Horizontal Span

Graphics cards in windows support two types of desktop and application visualizations using two monitors.

Dual view:

The two monitor can be configures independently. Different resolutions and settings can be applied and. The two monitors are logically placed side by side and windows can be dragged from one monitor to the other. Applications can either be aware of the existence of two monitors or can only be aware of the monitor they are initially launched in.

Span (vertical or horizontal)

The two displays are combined to one virtual display. There are two options of placement; placed side-by-side (horizontal span) or placed one on top of the other (vertical span). Applications are not aware of the excistence of two separate displays.

This mode has been discontinued after windows xp.

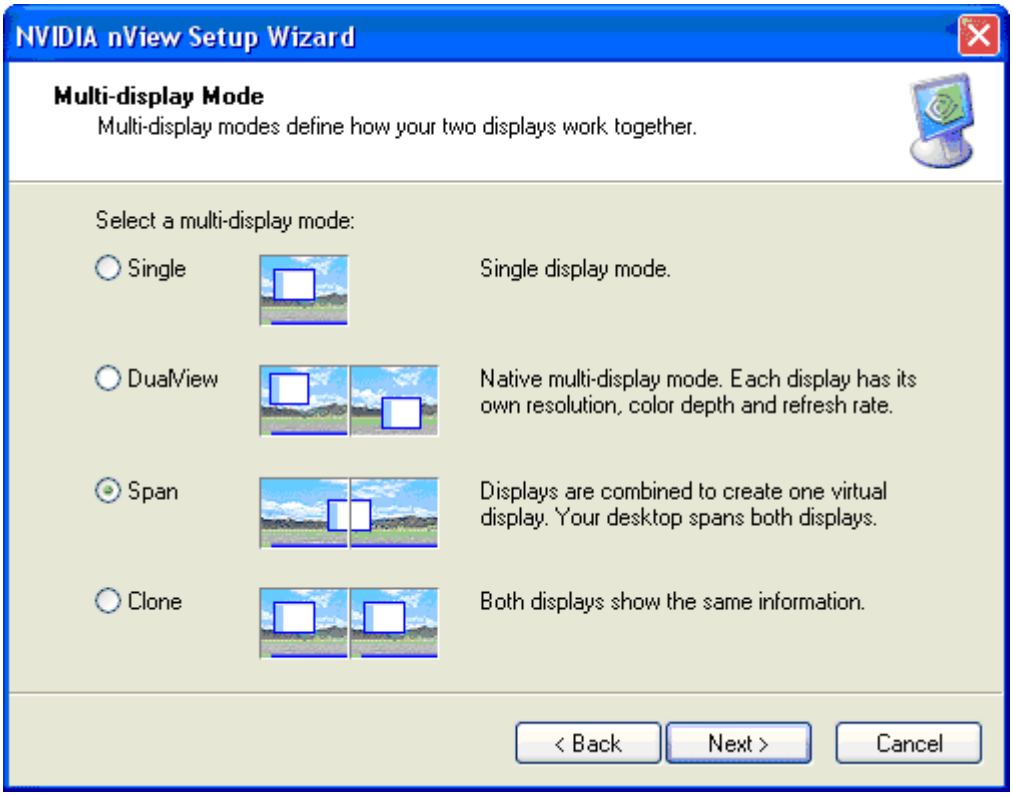


Figure 59: Multi display modes

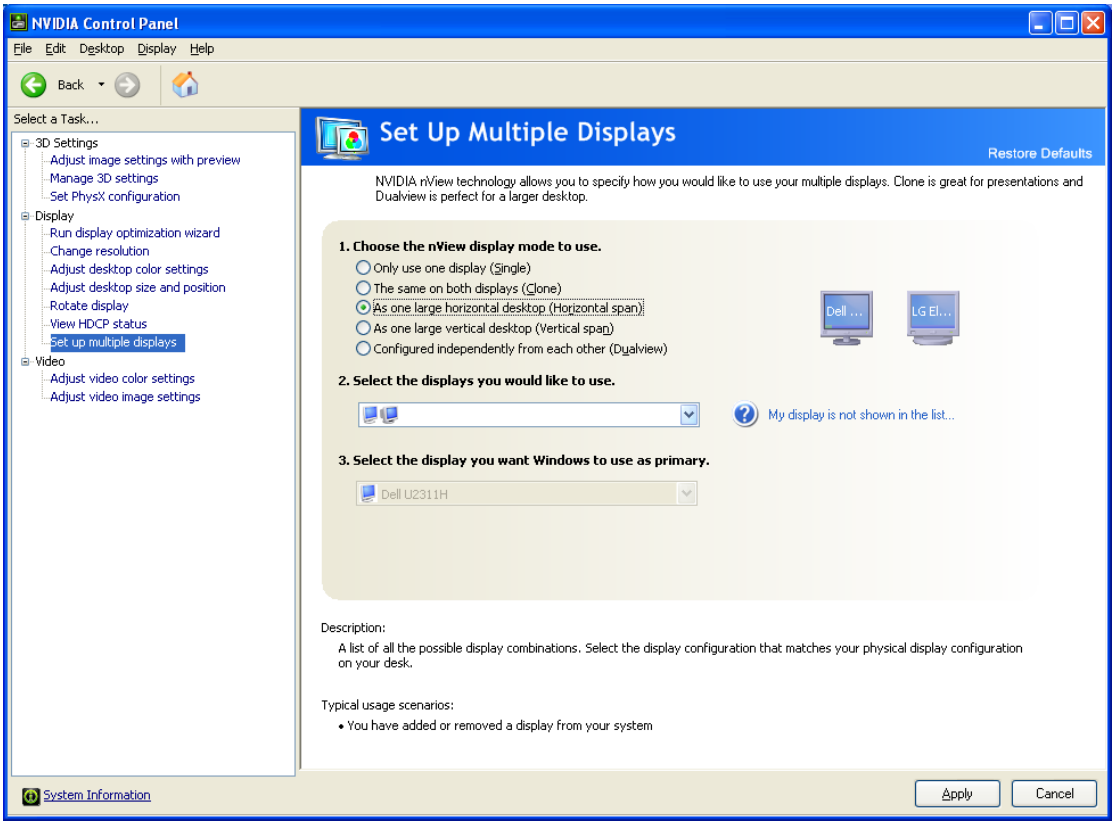


Figure 60: Using the two HMD displays as one (horizontal span)

3.8. Chapter Summary

This chapter provided information regarding the XVR application framework by analyzing the framework development language the integrated development environment and other software components. We also presented examples of using the framework.

In the following chapter we will present the hardware and software setup for the latency measurement mechanism and the results of latency minimization techniques to our system.

CHAPTER 4.Hardware and Software setup for Latency Measurements and Minimization

In this chapter, we will describe the latency measurement system proposed in this thesis and how this system is used. In the following sections we will analyze particular parts composing the system as shown in Figure 61, Figure 71, Figure 72. Along with the hardware setup of our latency measurement system we will also discuss the software setup and reorganizations used for interoperability between the hardware setup and the VE applications and for latency minimization. In the last section of this chapter the data from the latency measurements from out system as well as the results of the minimization techniques used, are presented.

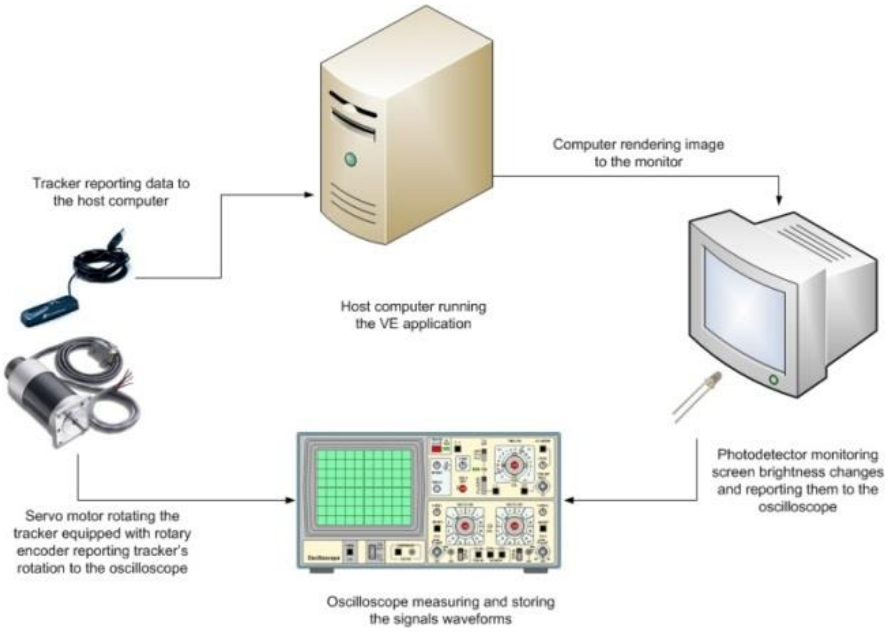


Figure 61: Overview of the data-acquisition system for measuring the end-to-end latency

4.1. Tracker

For our measurements we had an Intersense Intertrax2 and an Intersense InsertiaCube3, head trackers available. Both are portable high performance trackers. The Intertrax2 combines readings from multiple sensors, 3 Gyroscopes, 2 magnetometers and 2 accelerometers to produce 3 degrees of freedom (pitch, yaw and roll) head movement reports with relative angular resolution of 0.02degs. The InsertiaCube3 has 0.03degs resolution and it can provide future position predictions

The internal (hardware) latency of both trackers is 4ms. The Intertrax2 connects to the host computer through a USB connection while the InsertiaCube3 connects through a parallel port interface (there is also a USB interface converter).

Through preliminary measurements there was not found any particular difference in terms of latency between the two trackers. Overall the Intertrax2 tracker was found to be more stable than the InsertiaCube3 and so the final measurements described in the present chapter and the subsequent evaluation experiment discussed in CHAPTER 5 were conducted using this tracker (Intertrax2).

4.2. Rotary Encoder

A rotary encoder, is an electro-mechanical device, attached to a rotating object (i.e. a wheel or motor), used for converting the angular position or motion of a shaft or axle to an analog or digital signal (Encoders n.d.) (Repas n.d.). Rotary encoders are categorized in incremental and absolute in terms of output. Incremental encoders output provides information about the motion of the shaft. This information can be further processed elsewhere into information such as acceleration, speed, distance, rotations per minute (RPM) and position. Absolute encoders output specifies the current position of the shaft. In such terms absolute encoders are angle transducers. Rotary encoders are utilized in many applications requiring precise shaft unlimited rotation like industrial controls, robotics, special purpose photographic lenses, computer input devices (such as optomechanical mice and trackballs), and rotating radar platforms.

Absolute digital encoders produce a distinctive digital code for each unique angle of the shaft (Encoders n.d.). An incremental rotary encoder provides cyclical outputs on encoder rotation. Absolute encoders mostly have multiple code rings with various binary weightings which provide data representing the absolute position of the encoder within one turn. This kind of encoder is referred to as a parallel absolute encoder. The distinctive feature of the absolute encoder is that it reports the absolute position of the encoder to the electronics directly upon power-up with no requisite for indexing. A typical incremental encoder works differently by providing an A and

a B pulse output that does not provide any usable count information in their own right; rather, the counting is done in the external electronics. The starting point where of the counting depends on the counter in the external electronics rather than on the position of the encoder. In order that the position information, provided by the encoder is useful, this position must be referenced to the device to which it is attached, typically using an index pulse. The distinctive feature of the incremental encoder is that it reports an incremental position change of the encoder to the calculating electronics (Mitchell Electronics n.d.).

The basic types of absolute rotary encoders are optical and mechanical. In mechanical absolute encoders a metal disc that contains a set of concentric rings with openings is fixed to an insulating disc. The disc is rigidly fixed to the shaft. A row of sliding contacts is fixed to a stationary object so that each contact wipes against the metal disc at a varying distance from the shaft. Some of the contacts touch metal, while others fall in the gaps where the metal has been cut out, as the disc rotates with the shaft. The metal sheet is connected to an electric current source, and each contact is connected to a discrete electrical sensor. The metal pattern is designed such way that each possible position of the axle creates a distinctive binary code in which some of the contacts are connected to the current source (i.e. switched on) and others are not (i.e. switched off). In optical absolute encoders, the optical encoder's disc is glass made or plastic with transparent and opaque areas. A light source and photo detector array reads the optical pattern that result from the disc's position at any one time. The angle of the shaft can be determined by a controlling device such as a microprocessor or microcontroller that reads the code. The absolute analog type produces a unique dual analog code that can be translated into an absolute angle of the shaft (by using a special algorithm).

An example of a binary code, in a simplified encoder with only three contacts, is shown underneath.

Table 1: Absolute encoder standard binary encoding

Sector	Contact 1	Contact 2	Contact 3	Angle
1	off	off	off	0° to 45°
2	off	off	ON	45° to 90°
3	off	ON	off	90° to 135°
4	off	ON	ON	135° to 180°
5	ON	off	off	180° to 225°
6	ON	off	ON	225° to 270°
7	ON	ON	off	270° to 315°
8	ON	ON	ON	315° to 360°

In general, where there are n contacts, the number of distinct positions of the shaft is$2n$. In this example, n is 3, so there are 2^3 or 8 positions.

In the example above, as the disc rotates a standard binary count is produced by the contacts. Though, this has the disadvantage that if the disc breaks between two adjacent sectors, or the contacts are not perfectly aligned, it can be not possible to determine the angle of the shaft. However in a real-world device, the contacts are never perfectly aligned, so each switches at a different moment. If contact 1 switches first, followed by contact 3 and then contact 2, for example, the actual sequence of codes is:

off-on-on (starting position)

on-on-on (first, contact 1 switches on)

on-on-off (next, contact 3 switches off)

on-off-off (finally, contact 2 switches off)

In order, the sectors corresponding to these codes in the table are 4, 8, 7 and then 5. So, from the sequence of codes produced, the shaft appears to have jumped from sector 4 to sector 8 and then gone backwards to sector 7, then backwards again to sector 5, which is where we expected to find it. In many situations, this behavior is undesirable and could cause system failure. (I.e. in an encoder were used in a robot arm, the controller would think that the arm was in the wrong position, and try to correct the error by turning it through 180°, possibly damaging the arm).

Gray encoding is used to avoid the problem above. Gray encoding is a system of binary counting in which adjacent codes differ in only one position. For the three-contact example given above, the Gray-coded version would be as follows.

Table 2: Ablolute encdoer gray encoding

Sector	Contact 1	Contact 2	Contact 3	Angle
1	off	off	off	0° to 45°
2	off	off	**ON**	45° to 90°
3	off	**ON**	**ON**	90° to 135°
4	off	**ON**	off	135° to 180°
5	**ON**	**ON**	off	180° to 225°
6	**ON**	**ON**	**ON**	225° to 270°
7	**ON**	off	**ON**	270° to 315°
8	**ON**	off	off	315° to 360°

In this example, only one of the contacts changing its state from on to off or vice versa is involved in the transition from sector 4 to sector 5, like all other transitions, involves. This means that the sequence of incorrect codes shown in the previous illustration is impossible.

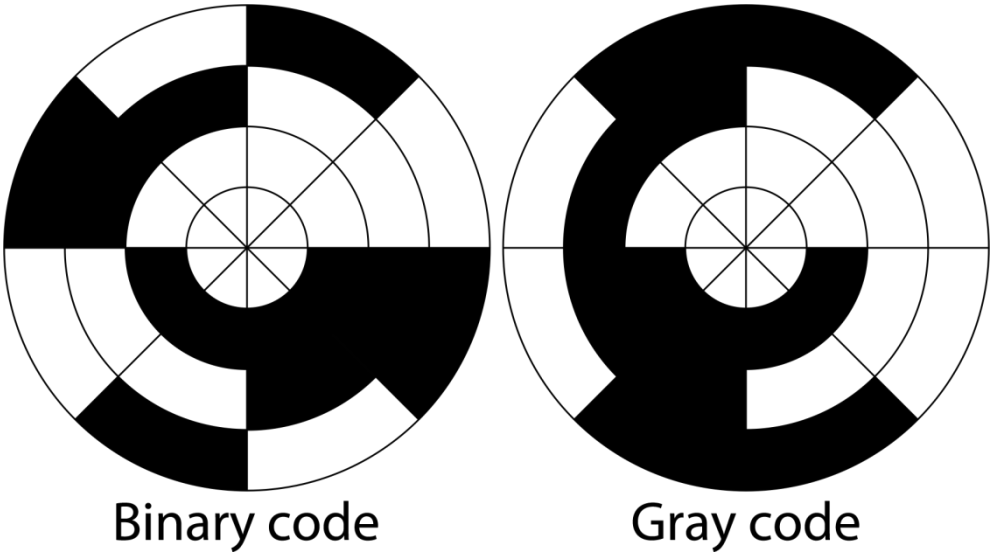


Figure 62: Rotary encoder for angle-measuring devices marked in 3-bit binary (left) and 3-bit gray (right). The inner ring corresponds to Contact 1 in the table. Black sectors are "on". Zero degrees is on the right-hand side, with angle increasing counterclockwise

When an incremental rotary encode is rotated only cylindrical output is provided. Incremental rotary encoders can be either mechanical or optical. The mechanical type is typically used as digital potentiometers on equipment including consumer devices and requires debouncing. Because mechanical switches require debouncing, mechanical encoders are limited in the rotational speeds they can handle.

Incremental encoders are used to track either linear or rotary motion and can be used to determine position and velocity. Very accurate measurements can be made because the direction can be determined.

They employ two outputs called A & B, which are 90 degrees out of phase and are called quadrature outputs.

The state diagram:

Table 3: Incremental encoder coding for clockwise rotation

Phase	A	B
1	0	0
2	0	1

3	1	1
4	1	0

Table 4: Incremental encoder coding for counter-clockwise rotation

Phase	A	B
1	1	0
2	1	1
3	0	1
4	0	0

The two output wave forms are 90 degrees out of phase, which is all that the quadrature term means. These signals are decoded to produce a count up pulse or a countdown pulse. For software decoding, the A & B outputs are read, either via an interrupt on any edge or polling, and the table above is used to decode the direction. For example, if the last value was 00 and the current value is 01, the device has moved one half step in the clockwise direction. The mechanical types would be debounced first by requiring that the same (valid) value be read a certain number of times before recognizing a state change.

In case the encoder is turning too fast, an invalid transition may occur, such as 00->11. Then there is no way to know which way the encoder turned; if it was 00->01->11, or 00->10->11. In case of even faster turning, a backward count may occur. Example: consider the 00->01->11->10 transition (3 steps forward). If the encoder is turning too fast, the system might read only the 00 and then the 10, which yields a 00->10 transition (1 step backward).

This same principle is used in ball mice to track whether the mouse is moving to the right/left or forward/backward.

Single output rotary encoders cannot be utilized to sense motion direction. They can be used for systems that measure rate-of-movement variables such as velocity and RPM and in certain applications to measure distance of motion

Our encoder was an absolute rotary encoder. The encoder was capable of selection between single and multi-turn modes using an input signal. The encoder was capable of 14-bit resolution when used in singleturn mode. It was powered using a standard 5V DC supply. The complete datasheet of the encoder can be found in APPENDIX D.

4.3. Rotational Mechanism

The rotational mechanism of our apparatus was based on a direct current (DC) electric motor.

An electric motor is an electromechanical device that converts between electrical and mechanical energy (Fink and Beaty 1999). A DC motor is designed to run on direct current electric power. DC electric motors, depending on how they generate motor, are categorized into brushed or brushless DC electric motors. A brushed DC electric motor can have low cost and easy motor speed control by varying the supply voltage. In such terms a brushed DC motor was ideal for our latency measurement mechanism.

The rotary encoder was attached to the rotating axis of the electro motor (Figure 69). Our head tracker was attached to another axis. Rotation of the electro motor axis was transferred to the rotating axis of the head tracker using a roller chain using 1-1 gear ration, so that rotation of the electromotor was resulting in the same rotational changes in the rotary encoder and the head tracker.

Figure 63: The rotational motor

4.4. Photodiode

A photodiode is a type of photodetector. It senses light. Photodiodes are capable of converting light into either current or voltage, depending upon the mode of operation (Nic, Jirat and Kosata 2006).

For our measurements we used a visible light sensitive planar silicon photodiode in recessed ceramic package (Figure 64), (Figure 65). The package incorporates an infrared rejection filter. These diodes have very high shunt resistance and have good blue response. Detailed electro-optical characteristics of the photodiode can be found in APPENDIX E.

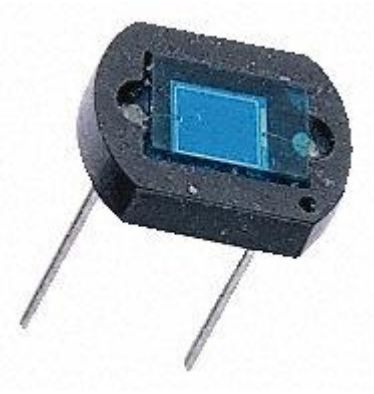


Figure 64: The EG & G Vactec visible light photodiode used for our measurements

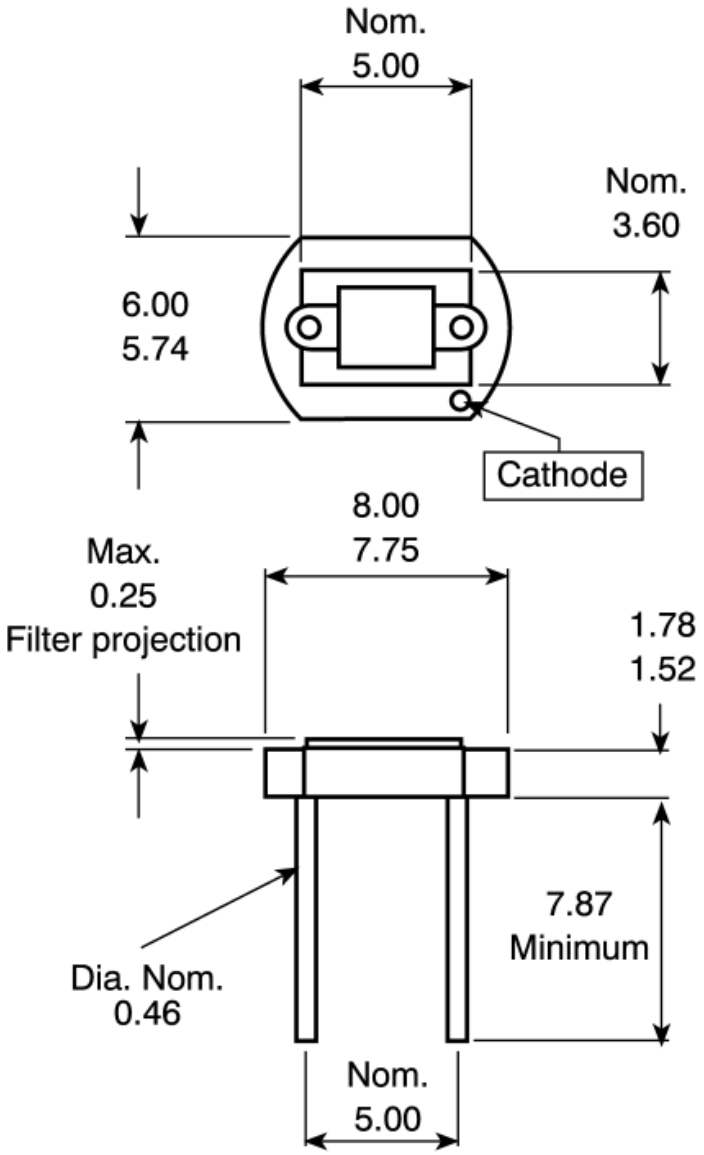


Figure 65: The photodiode mechanical characteristics

4.5. Other parts of the digital circuit

4.5.1. Digital-to analog converter

A digital-to-analog converter (DAC or D-to-A) is a device used to convert a digital signal (binary code) to an analog signal (current or voltage) (Integrated n.d.). A DAC converts an abstract finite-precision digital number (usually a fixed-point binary number) into the corresponding voltage. In particular, DACs are often used to convert finite-precision digital time series data to a continually varying (i.e. analog)

electric signal. Typically a DAC converts the numerical data into a concrete sequence of impulses that are subsequently processed by a reconstruction filter using some form of interpolation to fill in data between the impulses. Other DAC methods (e.g., methods based on Delta-sigma modulation) produce a pulse-density modulated signal that can then be filtered in a similar way to produce a smoothly varying signal. As per the Nyquist–Shannon sampling theorem, a DAC can reconstruct the original signal from the sampled data provided that its bandwidth meets certain requirements (e.g., a baseband signal with bandwidth less than the Nyquist frequency). Digital sampling introduces quantization error that manifests as low-level noise addition to the reconstructed signal.

The sequence of numbers usually updates the analogue voltage at uniform sampling intervals instead of impulses. These data are persisted to the DAC, typically with a clock signal that causes each number to be latched in sequence, at which time the DAC output voltage changes rapidly from the previous value to the value represented by the currently latched number. This has an effect that the output voltage is held in time at the current value until the next input number is latched resulting in a piecewise constant or 'staircase' shaped output. This is equivalent to a zero-order hold operation and has an effect on the frequency response of the reconstructed signal. Multiple harmonics above the Nyquist frequency are caused by the fact that DACs output a sequence of piecewise constant values or rectangular pulses. Usually, these harmonics can be removed with a low pass filter that acts as a reconstruction filter in applications that require it.

Types

The most common types of electronic DACs are (Integrated n.d.):

- The pulse-width modulator:
- Oversampling DACs
- The binary-weighted DAC
- The R-2R ladder DAC
- The thermometer-coded DAC

- Hybrid DACs
- The segmented DAC

Performance

DACs are very important to system performance. The most important characteristics of these devices are (Integrated n.d.):

- **Resolution**: This is the number of possible output levels the DAC is designed to reproduce. This is usually stated as the number of bits it uses, which is the base two logarithm of the number of levels. For instance a 1 bit DAC is designed to reproduce 2 (21) levels while an 8 bit DAC is designed for 256 (28) levels. Resolution is related to the effective number of bits (ENOB) which is a measurement of the actual resolution attained by the DAC.
- **Maximum sampling frequency:** This is a measurement of the maximum speed at which the DACs circuitry can operate and still produce the correct output. As stated in the Nyquist–Shannon sampling theorem, a signal must be sampled at over twice the frequency of the desired signal. For instance, to reproduce signals in the entire audible spectrum, which includes frequencies of up to 20 kHz, it is necessary to use DACs that operate at over 40 kHz. The CD standard samples audio at 44.1 kHz, thus DACs of this frequency are often used. A common frequency in cheap computer sound cards is 48 kHz — many work at only this frequency, offering the use of other sample rates only through (often poor) internal resampling.
- **Monotonicity:** Very important characteristic for DACs used as a low frequency signal source or as a digitally programmable trim element. Refers to the ability of a DAC's analog output to move only in the direction that the digital input moves (i.e., if the input increases, the output doesn't dip before asserting the correct output.) **THD+N:** A very important DAC characteristic for dynamic and small signal DAC applications. Measurement of the distortion and noise introduced to the signal by the DAC. Expressed as a

percentage of the total power of unwanted harmonic distortion and noise that accompany the desired signal.

- **Dynamic range:** This is a measurement of the difference between the largest and smallest signals the DAC can reproduce. Dynamic range is expressed in decibels. Usually related to DAC resolution and noise floor.

Other measurements, such as jitter and phase distortion, can also be very important for some applications of DACs.

Figures of merit

- Static performance (Integrated n.d.):
    - Differential nonlinearity (DNL); Indicator of how much two adjacent code analog values deviate from the ideal 1 LSB step.
    - Integral nonlinearity (INL); indicator of how much the DAC transfer characteristic deviates from an ideal one. Ideally a straight line; INL shows how much the actual voltage at a given code value differs from that line, in LSBs (1 LSB steps).
    - Noise; ultimately limited by the thermal noise generated by passive components (i.e. resistors). Usually a little less than 1 μV (microvolt) of white noise for audio applications and in room temperatures. This limits performance to less than 20~21 bits even in 24-bit DACs.
    - Offset
    - Gain
- Frequency domain performance
    - Spurious-free dynamic range (SFDR) Indicator of the ratio between the powers of the converted main signal and the greatest undesired spur. Measured in dB.
    - Signal-to-noise and distortion ratio (SNDR) Indicator of the ratio between the powers of the converted main signal and the sum of the noise and the generated harmonic spurs. Measured in dB.

- i-th harmonic distortion (HDi) Indicator the power of the i-th harmonic of the converted main signal
- Total harmonic distortion (THD); The sum of the powers of all HDi
- If the maximum DNL error is less than 1 LSB, then the D/A converter is guaranteed to be monotonic. However, many monotonic converters may have a maximum DNL greater than 1 LSB.

- Time domain performance:
  - Glitch energy
  - Time nonlinearity (TNL)
  - Response uncertainty

AD7840

For our system we used the AD 7840, a complete 14-Bit Voltage Output DAC, matching our 14-bit encoder (Figure 66). The AD7840 is a fast, complete 14-bit voltage output D/A converter. It consists of a 14-bit DAC, 3 V buried Zener reference, DAC output amplifier and high speed control logic. It provides the complete function for creating AC signals and DC voltages to 14-bit accuracy. It features an on-chip reference, an output buffer amplifier and 14-bit D/A converter. The AD7840 is capable of 14-bit parallel and serial interfacing. In the parallel mode, data setup times of 21 ns and write pulse widths of 45 ns make the AD7840 compatible with modern 16-bit microprocessors and digital signal processors. In the serial mode, the part features a high data transfer rate of 6 MHz. In our system we used the parallel interface matching out encoder.

The analog output from the AD7840 provides a bipolar output range of ± 3 V. The AD7840 is fully specified for dynamic performance parameters such as signal-to-noise ratio and harmonic distortion as well as for traditional DC specifications. Full power output signals up to 20 kHz can be created.

Full specifications of the AD7840 encoder can be found in APPENDIX F.

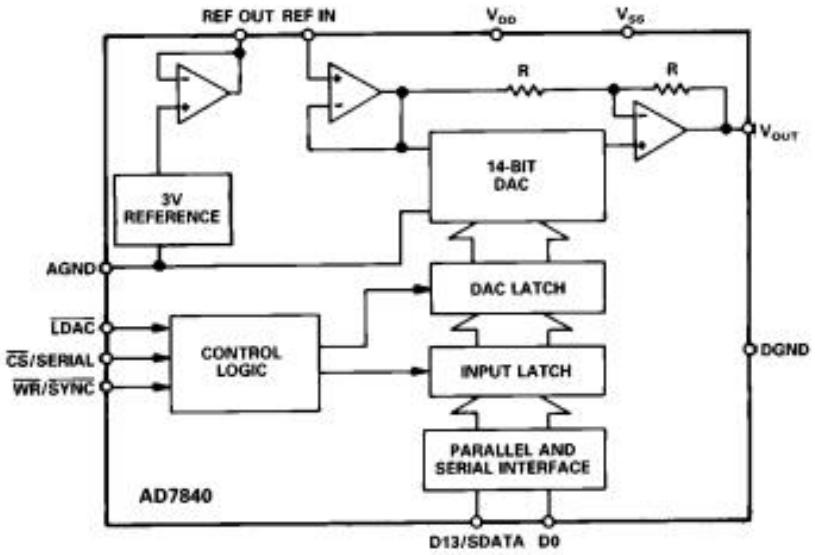


Figure 66: Functional block diagram of the digital to analog encoder used in our latency measurement system.

4.5.2. Current-to-voltage converter

A current-to-voltage converter is an amplifier that converts current to voltage. A common application of current-to-voltage-converters is in receivers for optical communications (Current-to-voltage converter n.d.). They convert the current generated by a photodetector into a voltage signal for further amplification. The input ideally has zero impedance. The input signal is a measured as a current. The output may have low impedance, or may be matched to a driven transmission line in high-frequency applications. The output signal is measured as a voltage. The gain, or ratio of output to input, is expressed in units of ohms because the output is a voltage and the input is a current. When constructed as a simple operational amplifier circuit (Figure 67), the gain is equal to the negative of feedback resistance. An operational amplifier is a DC-coupled high-gain electronic voltage amplifier with a differential input and, usually, a single-ended output. An operational amplifier produces an output voltage usually hundreds of thousands times larger than the voltage difference between its input terminals.

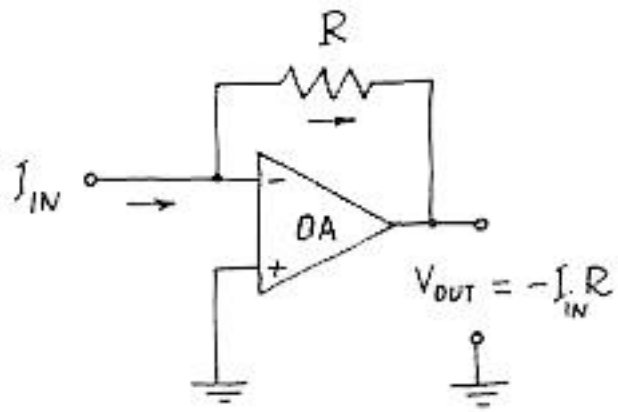


Figure 67: Operational amplifier current-to-voltage converter

4.5.3. LM555 timer

For the timing of the Digital-to-Analog Converter in our system a LM555 digital timer circuit was used (Figure 68).

A timer is a specific type of clock which can be utilized for controlling the sequence of an event or process (dictionary n.d.). Although a stopwatch counts upwards from zero for measuring elapsed time, a timer counts down from a specified time interval. Timers can be mechanical, electromechanical, electronic (quartz), or even software. All modern computers include digital timers of one kind or another. When the set period expires some timers simply indicate so (e.g., by an audible signal), while others operate electrical switches, such as a time switch, which cuts electrical power.

Electronic timers are quartz clocks controlled by electronic circuits. Electronic timers have higher precision than mechanical timers and furthermore are less expensive than most mechanical and electromechanical timers.

The 555 timer IC is an integrated circuit (chip) used in many timers, pulse generations and oscillators.it is a low price, high stability and easy to use.

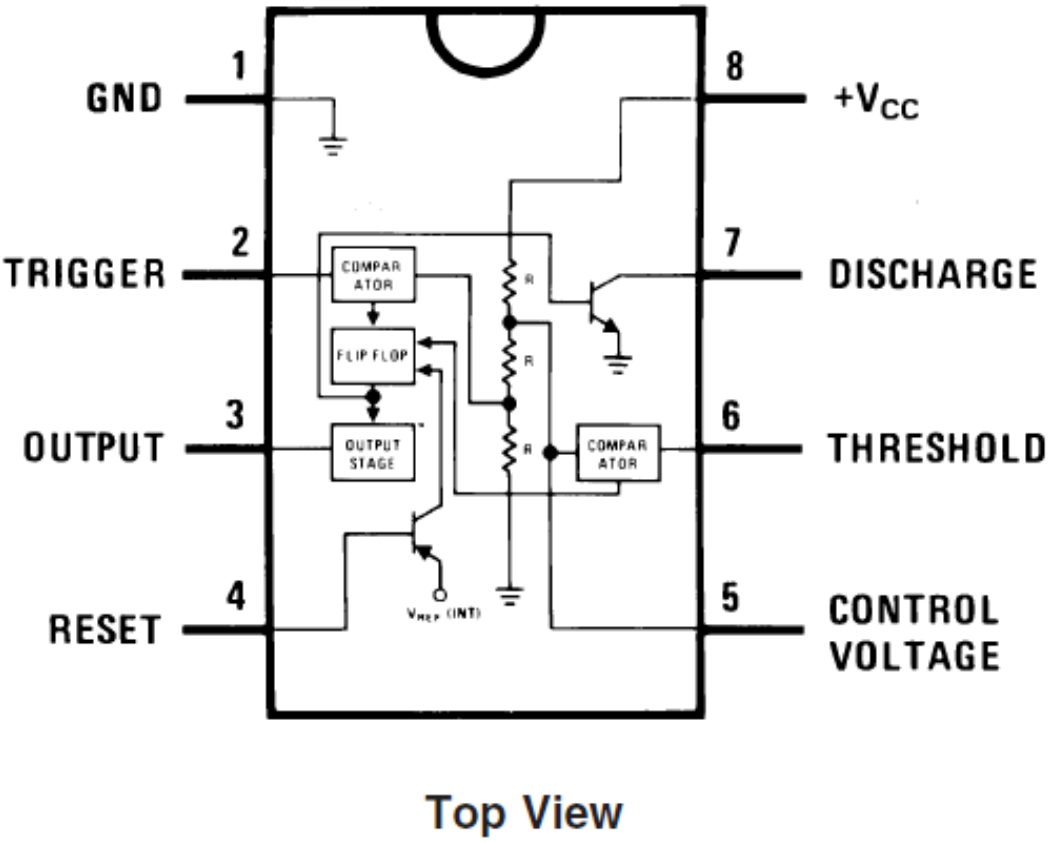


Figure 68: LM555 timer connection diagram

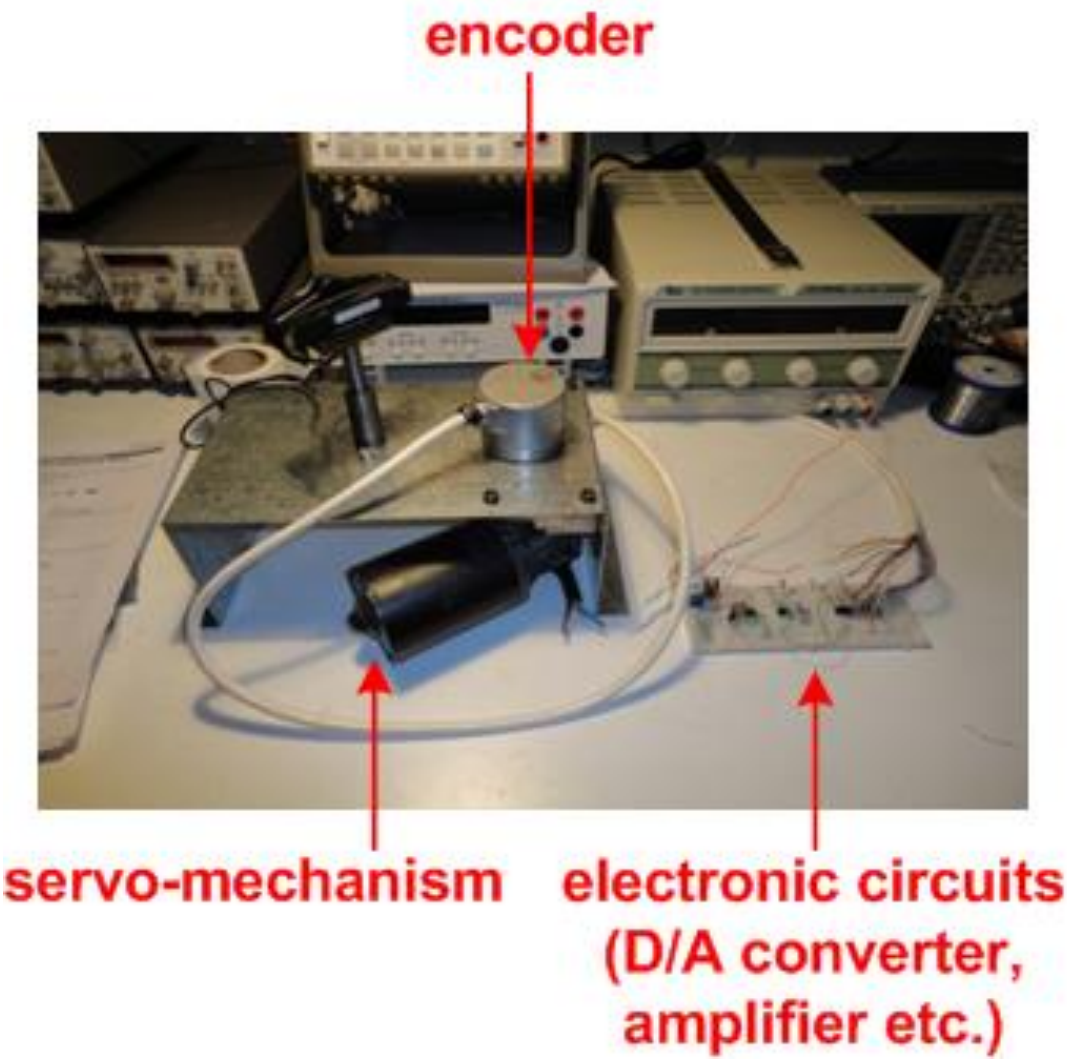


Figure 69: The laboratory-built prototype comprising the tracker rotation mechanism with the shaft encoder attached and the signal-conditioning electronic circuits

4.6. Oscilloscope

An oscilloscope is an electronic test instrument used to observe constantly varying signal voltages (Kularatna 2003). The signals are usually displayed as a 2-dimensional graph of one or more electrical potential differences using the vertical or *y* axis, plotted as a function of time, (horizontal or *x* axis). Although an oscilloscope displays voltage on its vertical axis, any other quantity that can be converted to a voltage can be displayed as well. Oscilloscopes are commonly used to observe the exact wave shape of an electrical signal. In addition to the amplitude of the signal, an

oscilloscope can show distortion, the time between two events (such as pulse width, period, or rise time) and relative timing of two related signals. (Kularatna 2003)

A typical oscilloscope is shown in Figure 70. It is divided into four sections: the display, vertical controls, horizontal controls and trigger controls.

The display; usually an LCD or CRT panel laid out with both horizontal and vertical reference lines. It is also referred as the graticule. Additionally to the screen, most display sections also include with three basic controls; a focus knob, an intensity knob and a beam finder button.

The vertical section; it controls the amplitude of the demonstrated signal. It carries a Volts-per-Division ($Volts/Div$) selector knob, an AC/DC/Ground selector switch and the vertical (primary) input for the instrument. Additionally, this section is typically equipped with the vertical beam position knob. And controls the time base or "sweep" of the instrument. The primary control is the Seconds-per-Division (Sec/Div) selector switch. This section also includes a horizontal input for plotting dual x-y axis signals. The horizontal beam position knob is usually located in this section.

The trigger section controls the initial event of the sweep. The trigger can be set automatically to restart on each sweep or it can be configured to respond to internal or external events. The main controls of this section are the source and coupling selector switches. An external trigger input (EXT Input) and level adjustment are typically also included.

In addition to the basic instrument, most oscilloscopes are supplied with a probe as shown in (Figure 70: Basic oscilloscope). The probe connects to any input on the instrument and typically has a resistor of ten times the oscilloscope's input impedance. This results in a .1 (-10X) attenuation factor, but helps to isolate the capacitive load presented by the probe cable from the signal being measured. Some probes have a switch allowing the operator to bypass the resistor when appropriate (Kularatna 2003).

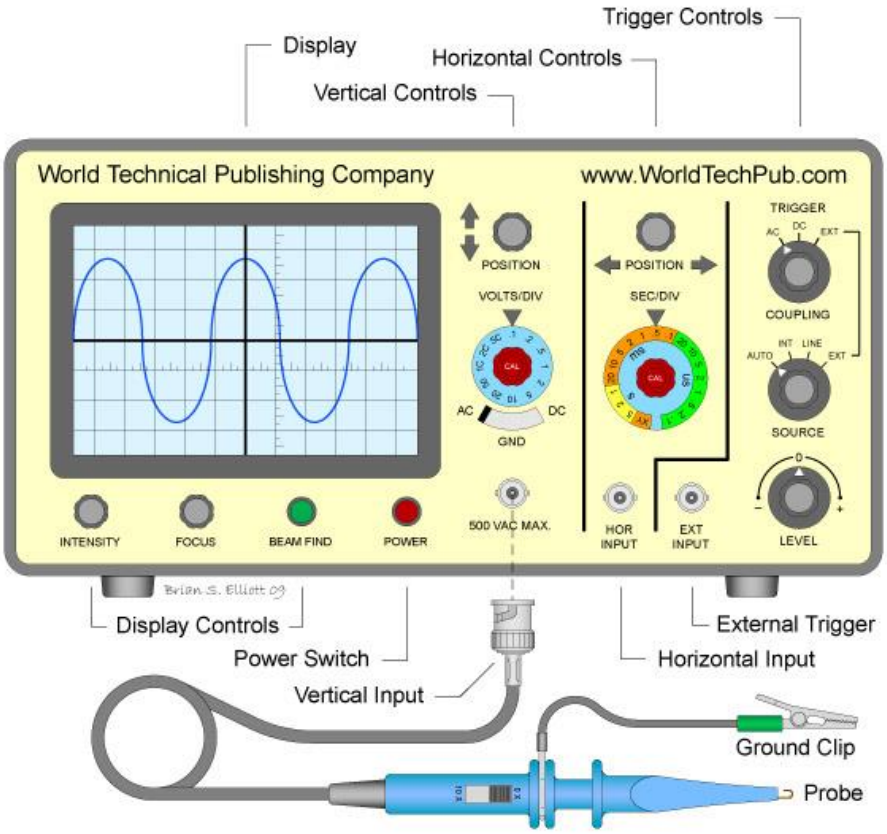


Figure 70: Basic oscilloscope (WorldTechPub n.d.)

The signal to be measured is fed to one of the input connectors. These connectors are usually a coaxial connector such as a BNC or UHF type. For lower frequencies Binding posts or banana plugs may be used. If the signal source has its own coaxial connector, then a simple coaxial cable is used; otherwise, a specialized cable called a "scope probe", supplied with the oscilloscope, is used. In general, for routine use, an open wire test lead for connecting to the point being observed is not satisfactory, and a probe is generally necessary. General-purpose oscilloscopes usually present an input impedance of 1 mega-ohm in parallel with a small but known capacitance such as 20 pico-farads. This allows the use of standard oscilloscope probes. Scopes for use with very high frequencies may have 50 ohm inputs, which must be either connected directly to a 50 ohm signal source or used with Z_0 or active probes. Less-frequently-used inputs include one (or two) for triggering the sweep, horizontal deflection for X-Y mode displays, and trace brightening/darkening, sometimes called "Z-axis" inputs.

The digital storage oscilloscope, abbreviated as DSO, is currently the preferred type for most industrial applications. It replaces the unreliable storage method used in analog storage scopes with digital storage. Digital storage can store data without degradation as long as required. It also allows complex processing of the signal by high-speed digital signal processing circuits (Kularatna 2003).

Bandwidth is a measure of the range of frequencies that can be displayed; it refers primarily to the vertical amplifier, although the horizontal deflection amplifier has to be fast enough to handle the fastest sweeps. The bandwidth of the oscilloscope is limited by the vertical amplifiers and the CRT (in analog instruments) or by the sampling rate of the analog to digital converter in digital instruments. The bandwidth is defined as the frequency at which the sensitivity is 0.707 of the sensitivity at lower frequency (a drop of 3 dB). The rise time of the fastest pulse that can be resolved by the scope is related to its bandwidth approximately (Spitzer and Howarth 1972):

Bandwidth (Hz) x rise time (sec) = 0.35

For example, an oscilloscope intended to resolve pulses with a rise time of 1 nanosecond would have a bandwidth of 350 MHz For a digital oscilloscope, a rule of thumb is that the continuous sampling rate should be ten times the highest frequency desired to resolve; for example a 20 Megasamples/second rate would be applicable for measuring signals up to about 2 MHz.

Modern oscilloscopes have triggered sweeps to display events with unchanging or slowly (visibly) changing waveforms, which occur at times that may not be evenly spaced, Compared to simpler oscilloscopes with sweep oscillators that are always running, triggered-sweep oscilloscopes are markedly more versatile. A triggered sweep starts at a selected point on the signal, providing a stable display. In this way, triggering allows the display of periodic signals such as sine waves and square waves, as well as non-periodic signals such as single pulses, or pulses that don't recur at a fixed rate. With triggered sweeps, the scope will blank the beam and start to reset the sweep circuit each time the beam reaches the extreme right side of the

screen. For a period of time, called holdoff, (extendable by a front-panel control on some better oscilloscopes), the sweep circuit resets completely and ignores triggers. Once holdoff expires, the next trigger initiates a sweep. The trigger event is usually the input waveform reaching some user-specified threshold voltage (trigger level) in the specified direction (going positive or going negative—trigger polarity). In some cases, variable holdoff time can be really useful to make the sweep ignore interfering triggers that occur before the events one wants to observe. In the case of repetitive, but quite-complex waveforms, variable holdoff can create a stable display that can't otherwise practically be obtained. Trigger holdoff defines a certain period following a trigger during which the scope will not trigger again. This makes it easier to establish a stable view of a waveform with multiple edges which would otherwise cause another trigger. Triggered sweeps can display a blank screen if there are no triggers. To avoid this, these sweeps include a timing circuit that generates free-running triggers so a trace is always visible. Once triggers arrive, the timer stops providing pseudo-triggers. Automatic sweep mode can be de-selected when observing low repetition rates.

Some oscilloscopes offer single sweeps. The sweep circuit is manually armed (typically by a pushbutton or equivalent) "Armed" means it's ready to respond to a trigger. Once the sweep is complete, it resets, and will not sweep until re-armed. This mode, combined with an oscilloscope, captures single-shot events.

Types of trigger include (Kularatna 2003):

- **External trigger,** a pulse from an external source connected to a dedicated input on the scope.
- **Edge trigger,** an edge-detector that generates a pulse when the input signal crosses a specified threshold voltage in a specified direction. These are the most-common types of triggers; the level control sets the threshold voltage, and the slope control selects the direction (negative or positive-going).
- **Video trigger,** a circuit that extracts synchronizing pulses from video formats such as PAL and NTSC and triggers the timebase on every line, a specified

line, every field, or every frame. This circuit is typically found in a waveform monitor device; although some better oscilloscopes include this function.

- **Delayed trigger,** which waits a specified time after an edge trigger before starting the sweep. As described under delayed sweeps, a trigger delay circuit (typically the main sweep) extends this delay to a known and adjustable interval. In this way, the operator can examine a particular pulse in a long train of pulses.

Four our measurements we used the Agilent DSO1012A digital oscilloscope (Figure 71). This is a 2 channel 100 MHz oscilloscope. The first channel is connected to the DAC converter thus it is monitoring the signal from the rotary encoder and the second channel is monitoring the signal from the photodetector.

The DSO1012A features:

- 100 MHz bandwidth
- 2 analog channels
- 2 GSa/s sample rate half channel, 1 GSa/s each channel
- 20 kpts memory half channel, 10 kpts each channel

It also offers features as:

- Sequence mode for easier debug
- Recording and playback of up to 1000 occurrences of a trigger event for further examination. Waveforms can be stored to internal or external memory (USB flash drive).
- Digital filtering on waveforms.
- Ability to apply a real-time digital filter the input source waveform to eliminate unwanted frequencies from the display. Digital filtering selections include low-pass, high-pass, band-pass and band-reject filters. Frequency limits are selectable between 250 Hz and the full bandwidth of the oscilloscope.
- Advanced triggering

Triggering options including edge, pulse width, composite video, pattern and alternate channel trigger modes. These modes ensure capturing and viewing hard-to-find signal conditions.

Technical sheets of the DSO1012A can be found at the APPENDIX G.

Figure 71: The DSO1012A oscilloscope used for our measurements

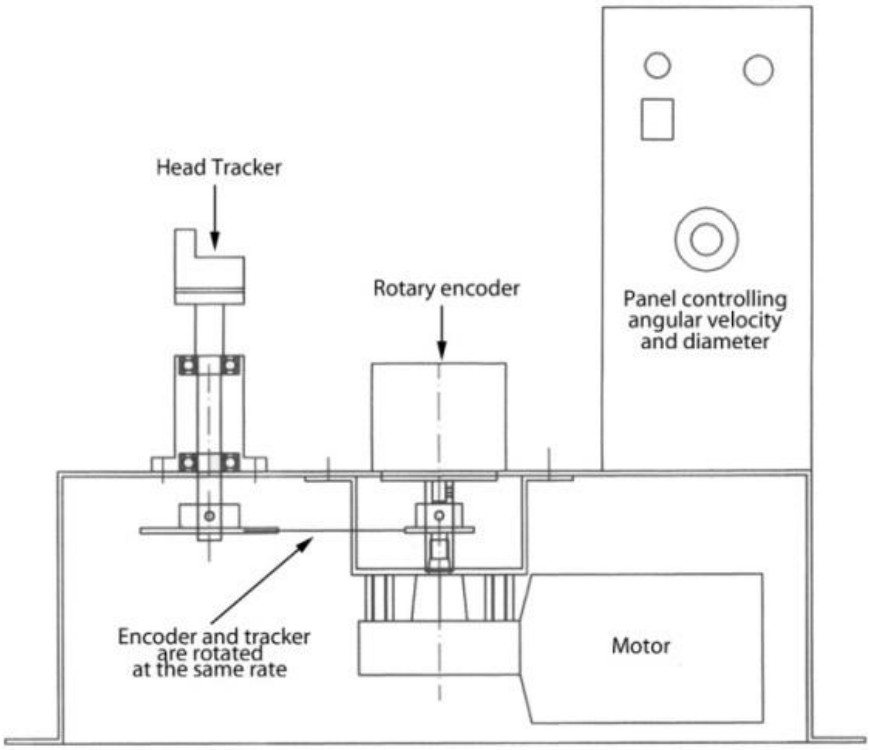


Figure 72: The diagram of the servo-mechanism which is used to control the tracker rotation

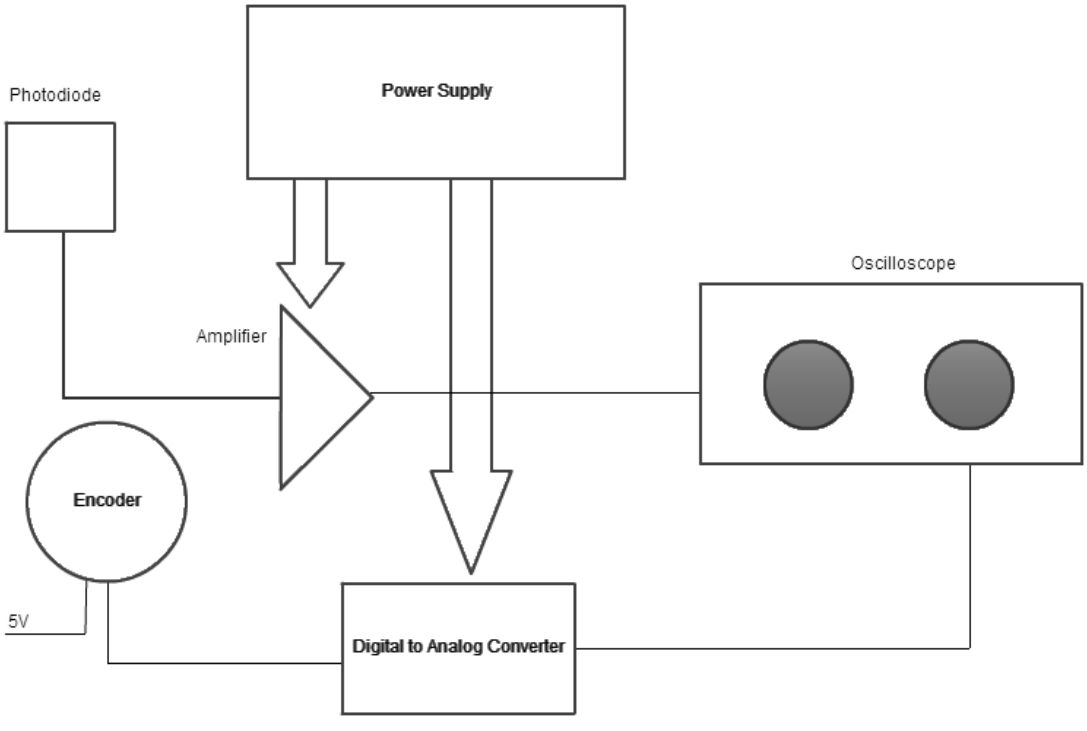


Figure 73: Digital circuit overview

4.7. Calculating display metrics

As described in section 2.2.2 in order the scene is correctly projected to the HMD, metrics of the scene have to be calculated in order that the correct parameters are applied to the VR application (XVR).

```
SceneSetParam(VR_TRACKER_POSITION,0.0,0.0,1);
        SceneSetParam(VR_SCREEN_SIZE,1,0.75);
        SceneSetParam(VR_EYE_SEPARATION,0.065);
```

Figure 74: Calculating display metrics

4.8. Adjusting stereo parallax

Stereo parallax can be readjusted on request by recalculating display metrics.

```
function OnEvent()
{
        if(Keypressed(VK_NUMPAD1))
        {       scrsz-=0.2;
                SceneSetParam(VR_TRACKER_POSITION,0.0,0,scrsz);
                SceneSetParam(VR_SCREEN_SIZE,scrsz,scrsz*0.75);
        }
        else if(Keypressed(VK_NUMPAD2))
        {
                scrsz+=0.2;
                SceneSetParam(VR_TRACKER_POSITION,0.0,0,scrsz);
                SceneSetParam(VR_SCREEN_SIZE,scrsz,scrsz*0.75);
        }
}
```

Figure 75: Re-adjusting stereo parallax

4.9. Reading from the Tracker

4.9.1. Using the tracker as joystick device to read head position

Using the plug 'n' play Microsoft Joystick driver

When plugged in a Windows pc the Intretrax2 tracker is immediately recognized as a joystick device and a driver from Microsoft provides tracking data access (Figure 77). As we measured the end-to-end tracking latency of this configuration we found that it was significantly high (~200ms) possible to low tracking data update rate from the Microsoft driver, thus this configuration was unacceptable for our low-latency system.

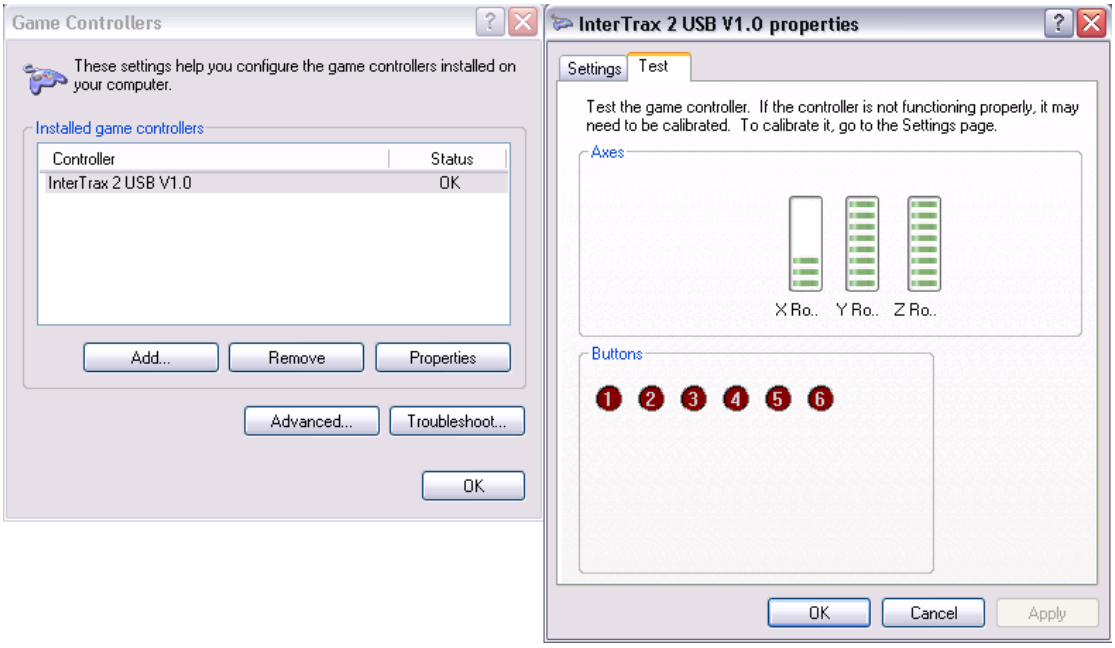


Figure 76: When inserted, Windows recognize the Intrtrax2 tracker as a joystick

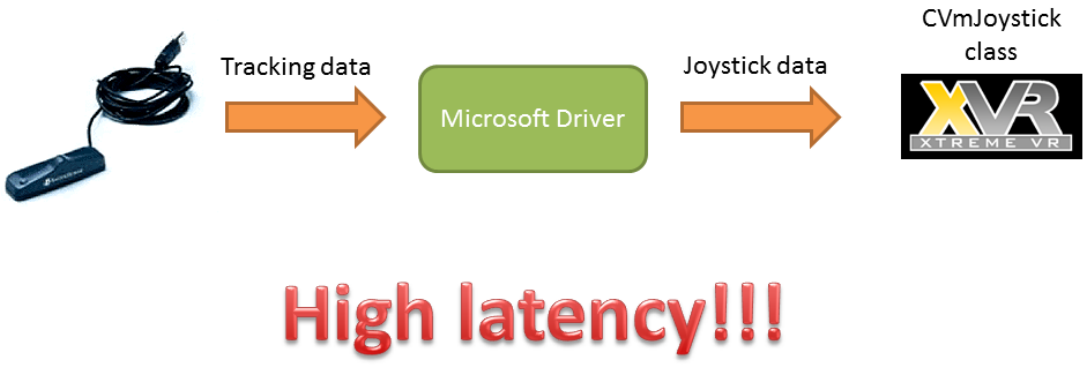


Figure 77: Accessing tracker data as Joystick readings with the plug 'n' play Microsoft driver

Using the InterSense server joystick emulation driver

InterSense Server provides a joystick interface for reading tracker data from the Intertrax2 tracker. This driver is different from the "plug 'n' play" Microsoft driver and it comprises of two intermediate interfaces the InterSense Joystick Interface Driver, which is the back-end interface and the InterSense Joystick Driver which is the front-end interface (Figure 78).

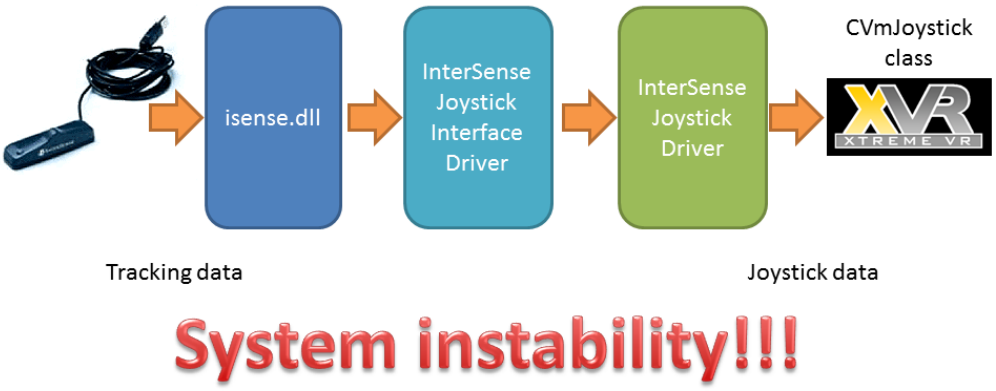


Figure 78: Accessing tracker data as Joystick readings with the InterSense Joystick Driver interface

The driver had a very good performance in terms of end-to-end latency but it caused system instability when running demanding, in terms of computational power, VE applications. The driver was crushing causing operating system crashes (Blue Screens of Death - BSOD) after which the system was rebooting.

The problems with latency and stability discussed in this section made the joystick interface of the tracker unusable for out VE application.

4.9.2. Using the C++ Intersense tracker API

InterSense provides SDK and DLL/shared library, as well as troubleshooting tools. The API can be used by the application software to initialize and retrieve data from the InterSense devices using the InterSense library (isense.dll / libisense.so / libisense.dylib). This library and API is provided to simplify communications with all models of InterSense tracking devices. It can detect, configure, and get data from up to 32 trackers, which may have multiple (up to 8) stations in some cases, such as the IS-900 processor. The library maintains compatibility with existing devices, and

also makes the applications forward compatible with all future InterSense products. The library is intended to be backwards compatible, in the sense that software written for older versions of the DLL should generally run without recompilation using the current version.

API organizes data in C++ structures and passes pointer to return values. Below we can see the call to the API that returns the tracking data from the tracker.

```
SD_GetTrackingData()

Bool      ISD_GetTrackingData(      ISD_TRACKER_HANDLE      handle,
ISD_TRACKING_DATA_TYPE *Data )
```

Get data from all configured stations. Data is places in the ISD_TRACKING_DATA_TYPE structure. TimeStamp is only available if requested by setting TimeStamped field to TRUE. Returns FALSE if failed for any reason.

- Handle

Handle to the tracking device. This is a handle returned by ISD_OpenTracker() or ISD_OpenAllTrackers().

- Data

Pointer to a structure of type ISD_TRACKER_DATA_TYPE. See below for structure definition. Orientation data order is Yaw, Pitch, and Roll for Euler angles and W, X, Y, Z for quaternions.

```
ISD_TRACKING_DATA_TYPE

typedef    struct    {                        ISD_STATION_DATA_TYPE
Station[ISD_MAX_STATIONS]; } ISD_TRACKING_DATA_TYPE;

typedef struct {

BYTE    TrackingStatus;

BYTE    NewData;
```

```
BYTE    CommIntegrity;
BYTE    BatteryState
float   Euler[3];
float   Quaternion[4];
float   Position[3];
float   TimeStamp;
float   StillTime;
float   BatteryLevel;
float   CompassYaw;
Bool    ButtonState[ISD_MAX_BUTTONS];
short   AnalogData[ISD_MAX_CHANNELS];
BYTE    AuxInputs[ISD_MAX_AUX_INPUTS];
float   AngularVelBodyFrame[3];
float   AngularVelNavFrame[3];
float   AccelBodyFrame[3];
float   AccelNavFrame[3];
float   VelocityNavFrame[3];
float   AngularVelRaw[3];
BYTE    MeasQuality;
BYTE    bReserved2;
BYTE    bReserved3;
BYTE    bReserved4;
DWORD   TimeStampSeconds;
DWORD   TimeStampMicroSec;
DWORD   OSTimeStampSeconds;
DWORD   OSTimeStampMicroSec;
```

```
float   Reserved[55];

float   Temperature;

float   MagBodyFrame[3]; } ISD_STATION_DATA_TYPE;
```

This API combines the performance in terms of latency of the Intersence interface driver (which in fact is built upon this API) and is comparable in stability with the Microsoft Driver. It also provides the extensibility to build a compatible interface for the XVR VE application framework.

4.10. Intermediate DLL API for InterSense trackers

In order to overcome the problems and software incompatibilities described in previous sections we developed an intermediate DLL tracker data access API for the InterSense tracking devices and the XVR development Studio. The Intermediate API is a DLL that serves as an intermediate proxy between the original InterSense API isense.dll and the XVR Development Studio external DLL access API (CVmExternDLL). It is also expandable to other VE applications that are incompatible with the complex data structures of the InterSense DLL API and need to simpler data types.

Intermediate DLL API uses the InterSense API to read tracking data directly from the tracking device and converts the complex data structures to simpler readable from the CVmExternDLL. The Intermediate DLL API was written in C++ (Figure 79).

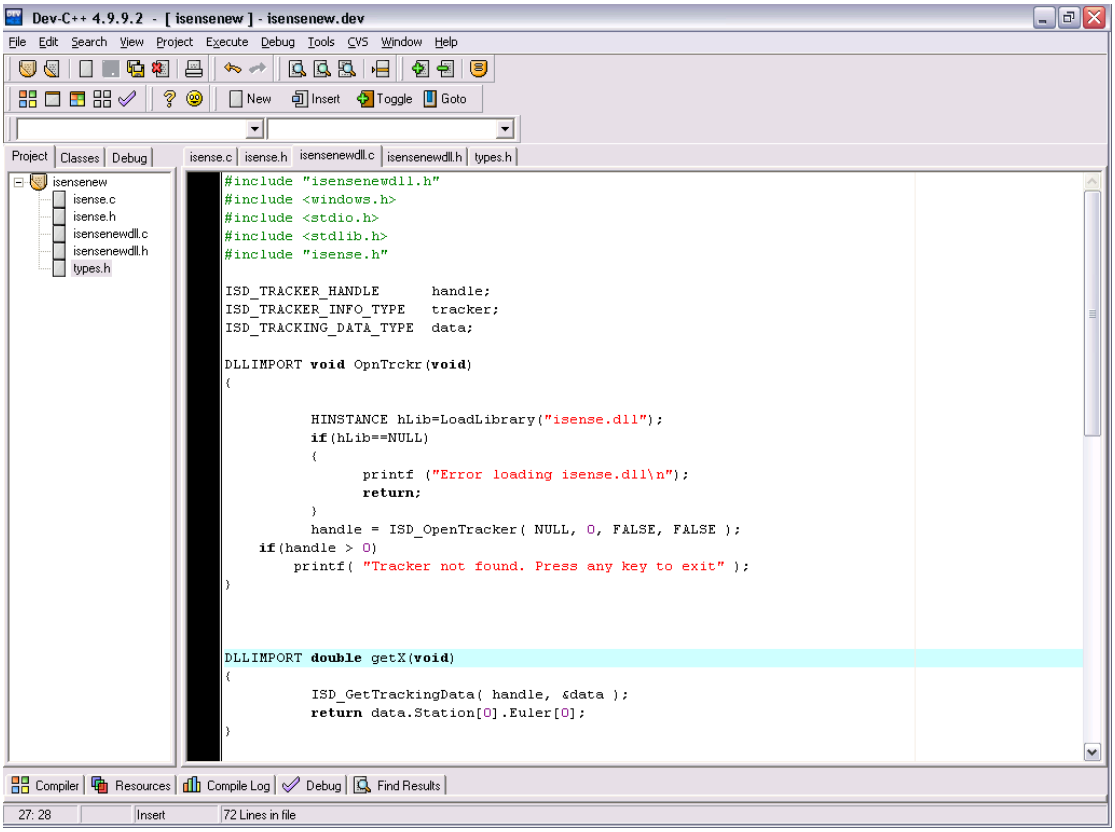


Figure 79: Intermediate DLL API development

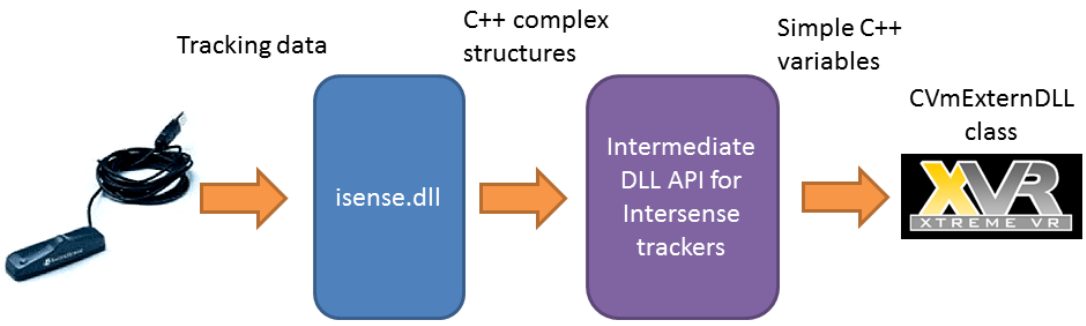


Figure 80: Accessing tracker data as using the Intermediate DLL API for Internense trackers

The Intermediate ALL API for InterSense tracers exports the following functions;

DLLIMPORT void OpnTrckr(void);

Opens the default tracker

DLLIMPORT double getX(void);

Returns the X axis value of the tracker as a double

DLLIMPORT double getY(void);

Returns the Y axis value of the tracker as a double

DLLIMPORT double getZ(void);

Returns the Z axis value of the tracker as a double

DLLIMPORT void ClsTrckr(void);

Closes the default tracker

The intermediate DLL API for InterSense tracker also provides input delaying functionality. This functionality was useful for the experiments described in a later section for which we needed to add a constant amount of latency to our head tracking input in our VE application in order to assess the latency impact on memory awareness states. Delaying of head tracking input is achieved by aging tracker reports in a circular buffer (Figure 81) to increase latency without affecting frame rate or tracking rate.

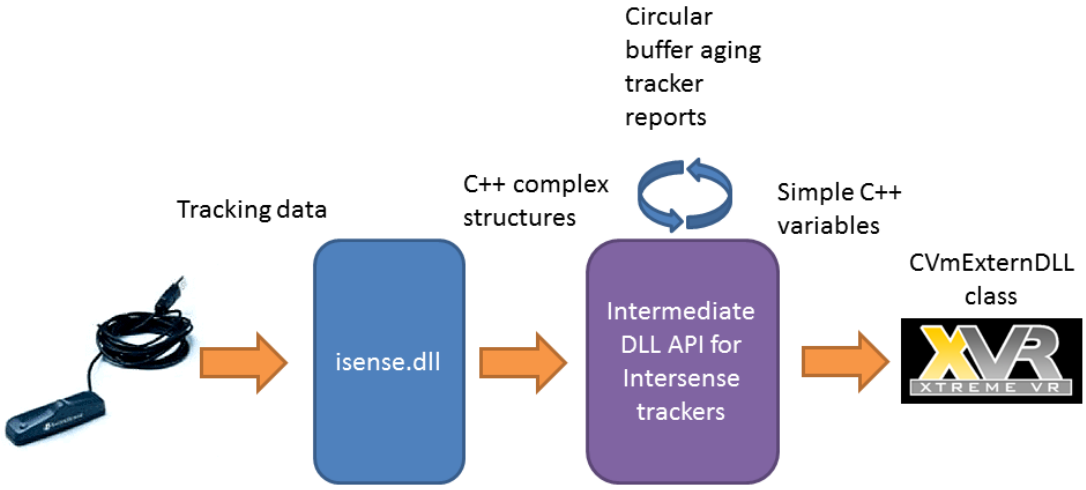


Figure 81: Aging tracker reports using a circular buffer

In order to use the aging feature of the Intermediate DLL API for InterSense trackers, a multiple versions of DLL with a circular buffer enabled were compiled. The size of the circular buffer also varied from version to version. The appropriate DLLwas loaded at the XVR VE application (Figure 82).

```
Library1=CVmExternDLL("isensenew.dll");

Library2=CVmExternDLL("isensenew_delayed12.dll");
```

```
Library3=CVmExternDLL("isensenew_delayed25.dll");
```

Figure 82: Loading both DLL versions with and without delayed reporting

4.11. Latency Minimization

"Triple buffering" and its consequent added latency were eliminated by removing the graphics system's frame timing by disabling the feature at the graphics card control panel (Figure 83). Instead of using the OpenGL v-sync dependent setting, we directly coupled the timing of our simulation-graphics application to the display VSync through a combination of custom software and hardware measures. This removes the extra frame of latency, but maintains a steady frame rate and prevents image tearing. First, the OpenGL software control panel setting responsible for synchronizing the buffer swap with the vertical blank interval is turned off. While this has the effect of deactivating the additional third buffer, double buffering remains intact and all drawing still occurs to the back buffer. However, since the hardware VSync is not being used, the draw cycle is no longer locked to 60 Hz, and runs instead at a higher rate (130-300 Hz, depending on VE image complexity) (Figure 84). Because, the buffer swap is no longer tied to the display's vertical blank interval, images are swapped into the front buffer as soon as they are completed. This results in image tearing where portions of successive separate images generated during each 60 Hz interval appear as distinct horizontal bands within the same display frame.

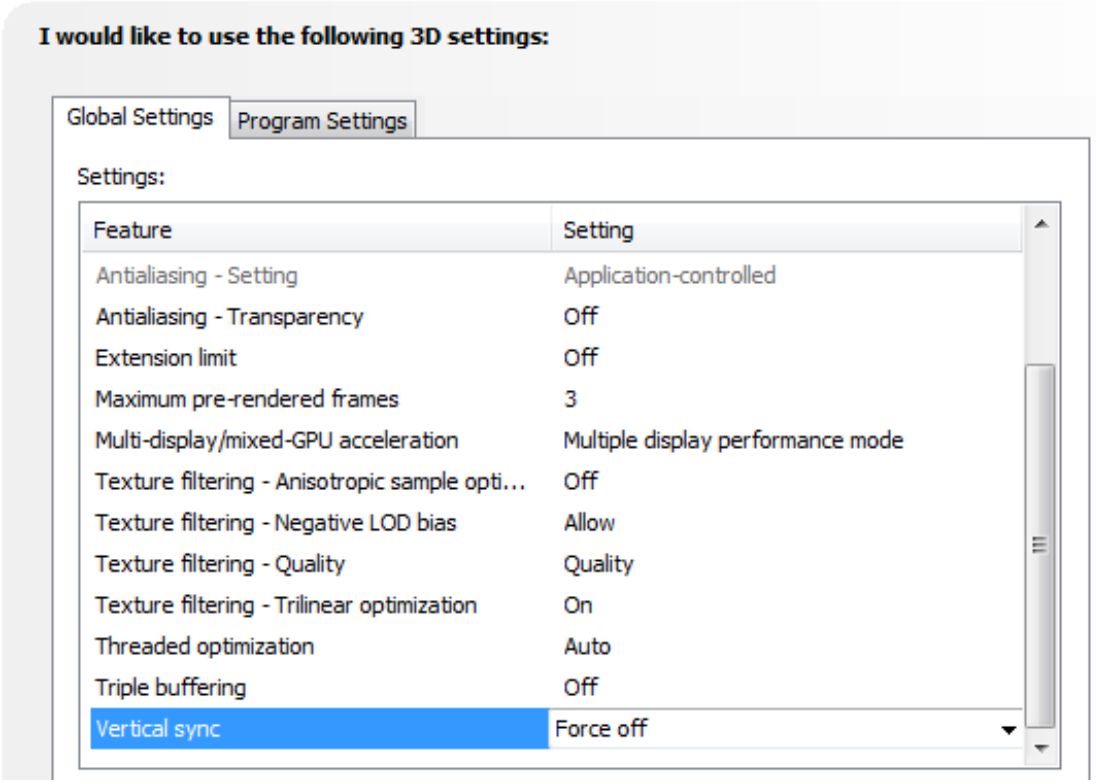


Figure 83: Turning of triple buffering and Vertical Sync from the graphics card control panel

```
44
45
46      SetFrameRate(9999);
47      SceneSetParam(VR_VERTICAl_SYNC,1);
48
49
```

Figure 84: Turning off VSync from the VE application and setting frame rate to maximum

4.12. Measurements

In our system, the end-to-end-latency is measured using a variation of the techniques described in section **2.4**, designed to accurately measure latency of the Intertrax2 and the InsertiaCube3 head tracker, by utilizing a low-cost, custom-made portable measuring mechanism with relative angular resolution of 0.02° and internal latency of 2ms. Previous methods used a pendulum which was moving a 6 DOF (Degrees of Freedom) positional tracker about the 3-rotational axes (i.e. roll, pitch and yaw) and along the 3 positional axes (i.e. x, y and z). The orbit of the tracker movement was forming an arc and a photodiode or encoder was reporting crossings

through a point. In the proposed system, the tracker is capable only to perform rotational 3-DOF movement tracking resulting in higher accuracy because of translational error control. In particular, the tracker movement at the measurements is restricted to rotational only, on one axis. The data-acquisition system for measuring the end-to-end latency is illustrated in Figure 61. A custom made modular servo-mechanism, depicted in Figure 63, with a 14-bit, parallel-output digital rotary encoder attached to its shaft rotates the tracker back-and-forth within a preset threshold angle. The 14-bit resolution of the encoder matches the angular resolution of the tracker used in this study. The angular velocity and arc of the movement are fully controllable through a power supply and a double-pole/double-throw switch. The encoder output signals are interfaced to a D/A converter and then passed to the oscilloscope. The XVR VE application is configured such that passing through a threshold angle results in VE changes. Both the tracker and the VE application are zero-calibrated prior to the measurements.

The original scene is photorealistically illuminated using pre-computed radiosity textures and stereoscopically rendered, using XVR's side-by-side stereoscopic rendering feature. The VE represents a room as described in section 5.1. The polygon count of the scene was ~140,000 polygons. A box is superimposed at every frame on the uppermost left corner of the screen (Figure 5). The application is configured to change the color of the box from black to white and vice versa at each threshold crossing of the tracker. A photodiode with spectral sensitivity in the visible light is attached to the front of the monitor and it is used to measure the brightness changes of the superimposed box. Rather than using the Head Mounted Display (HMD) system to be utilized for future experimental work, we used a standard LCD monitor configured to refresh at 60Hz similar to our Kaiser Electro-optics Pro-View 50 Head Mounted Display described in CHAPTER 2 that was also used in experiments about the effect of latency in immersive simulations described in 0. The small dimensions of the HMD displays make it hard to attach a photodiode on it. The refresh rate of both LCD and HMD displays is similar (60 Hz). Each monitor is configured to display the VE at 1024*1024 768 resolution matching the resolution of the HMD displays. The

photodiode output signal is amplified using an operational amplifier-based current-to-voltage converter. The laboratory-built prototype comprising the tracker rotation mechanism with the shaft encoder attached and the signal-conditioning electronic circuits is depicted in a digital oscilloscope with waveform storage capability is used to measure and store parallel digital samples of the D/A converter and amplifier output signals, corresponding to the tracker position and the display brightness level, respectively. An example of these signals is illustrated in Figure 86.

The oscilloscope used in the experimental setup is configured to acquire 10.000 samples at a time frame of 1 second (i.e. one sample per 1/10 of a millisecond).

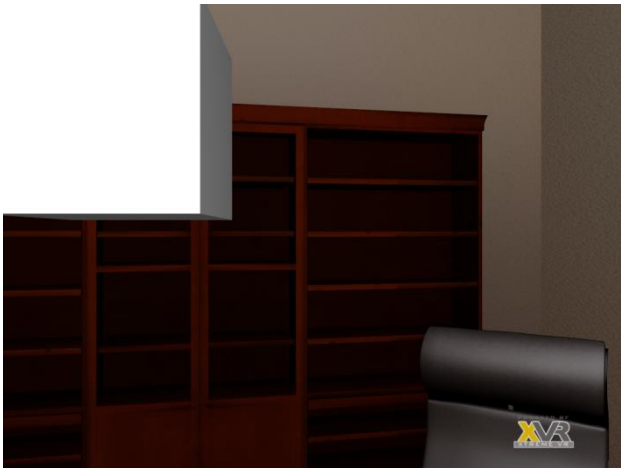


Figure 85: Test scene for latency measurements

The samples acquired by the oscilloscope are downloaded to a PC through a USB communication interface. Implemented software compares the individual values of the signals measured, in order to calculate the time-shift between the passing of the tracker through the threshold angle and the black-to-white transition of the polygons. This time-shift is equal to the end-to-end latency of the system. Estimates (mean ± standard deviation) of the VE latency were derived from averaging measurements of a hundred back-and-forth threshold crossings by our rotation mechanism; fifty of them when moving the tracker from right to left and fifty vice-versa. The estimated latency of our system was measured to be 90ms ±10% before minimization processes were applied as described below, inclusive of the latency induced by the refresh rate of the screen (Figure 87).

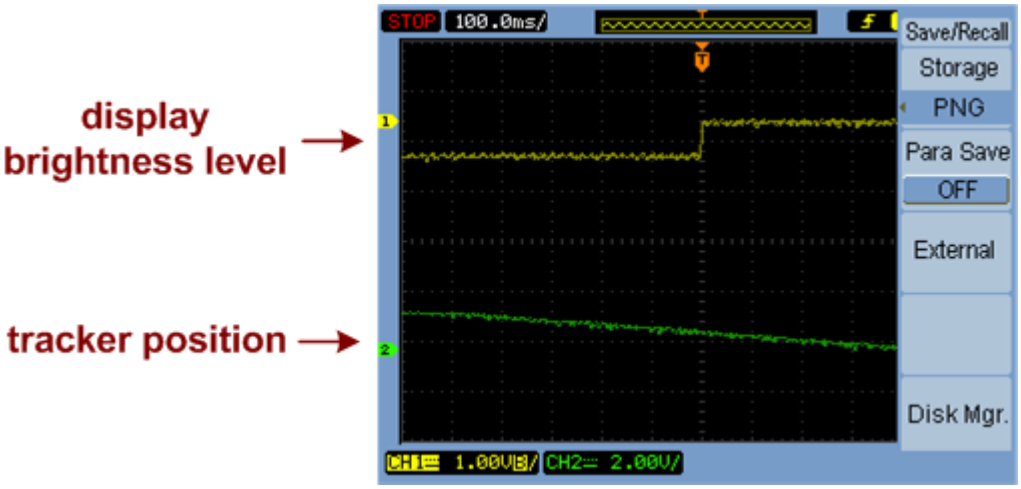


Figure 86: An example of the signals measured using the oscilloscope, corresponding to the tracker position and the display brightness level, respectively

4.12.1. Captured data

The DSO1012A offers data capturing to a USB flash memory as CSV data. Each capture stores 10000 continuous samples at specified frame. A frame of 1 second was chosen

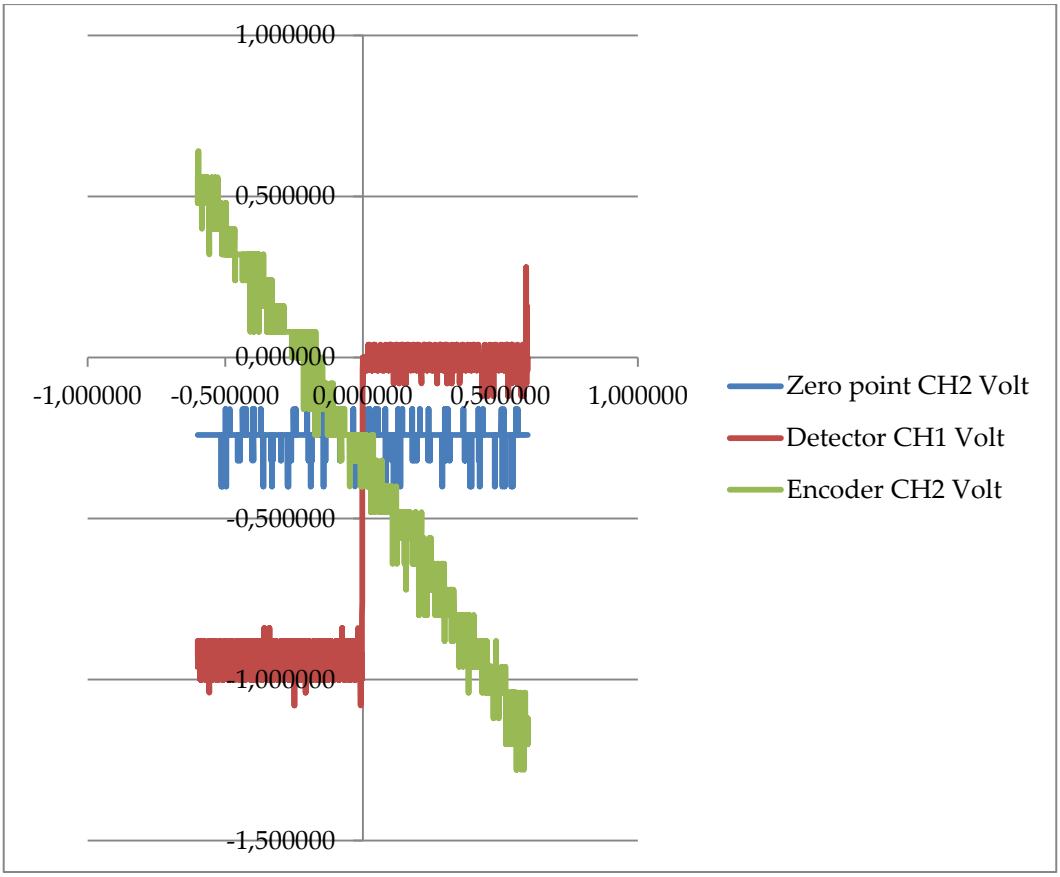


Figure 87: Raw captured data plot

4.12.2. High frequency noise removal (moving average)

Moving average is a type of finite impulse response filter used to analyze a set of data points by creating a series of averages of different subsets of the full data set. Given a series of numbers and a fixed subset size, the moving average can be obtained by first taking the average of the first subset. The fixed subset size is then shifted forward, creating a new subset of numbers, which is averaged. This process is repeated over the entire data series. The moving average is the plot line connecting all the averages. A moving average is a set of numbers, each of which is the average of the corresponding subset of a larger set of data points. A moving average may also use unequal weights for each data value in the subset to emphasize particular values in the subset. A moving average is a type of convolution and so it can be viewed as an example of a low-pass filter used in signal processing. When used with non-time series data, a moving average filters higher frequency components without any

specific connection to time, although typically some kind of ordering is implied. Viewed simplistically it can be regarded as smoothing the data.

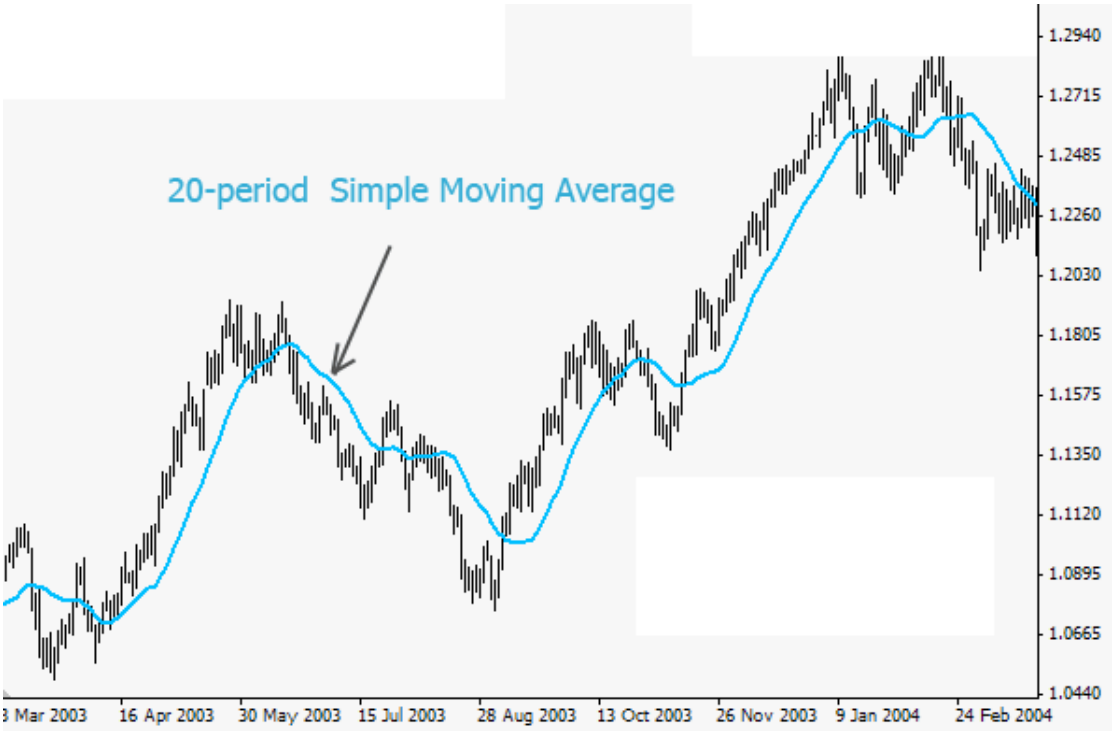


Figure 88: Moving average on a set of data.

4.12.3. Linear interpolation

Linear interpolation is a method of curve fitting using linear polynomials. It is heavily employed in mathematics (particularly numerical analysis), and numerous applications including computer graphics (Meijering 2002). It is a simple form of interpolation.

The linear interpolant is the straight line between two given points, given by the coordinates (x_0, y_0) and(x_1, y_1). For a value x in the interval(x_0, x_1), the value y along the straight line is given from the equation

$$\frac{y - y_0}{x - x_0} = \frac{y_1 - y_0}{x_1 - x_0}$$

which can be derived geometrically from Figure 89: Given the two red points, the blue line is the linear interpolant between the points, and the value y at x may be

found by linear interpolation. It is a special case of polynomial interpolation with $n = 1$.

Solving this equation for y, which is the unknown value at x, gives

$$y = y_0 + (x - x_0)\frac{y_1 - y_0}{x_1 - x_0} = y_0 + \frac{(x - x_0)y_1 - (x - x_0)y_0}{x_1 - x_0}$$

which is the formula for linear interpolation in the interval (x_0, x_1). Outside this interval, the formula is identical to linear extrapolation.

This formula can also be interpreted as a weighted average. The weights are inversely related to the distance from the end points to the unknown point; the closer point has more influence than the farther point. Thus, the weights are $\frac{x-x_0}{x_1-x_0}$ and $\frac{x_1-x}{x_1-x_0}$, which are normalized distances between the unknown point and each of the end points.

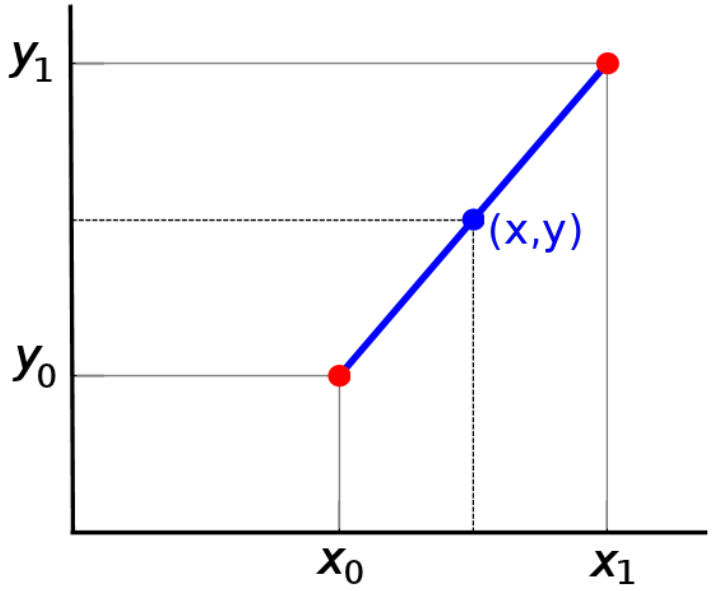


Figure 89: Given the two red points, the blue line is the linear interpolant between the points, and the value y at x may be found by linear interpolation

Linear interpolation on a set of data points (x_0, y_0), $(x_1, y_1)\ldots$ (x_n, y_n) is defined as the concatenation of linear interpolants between each pair of data points. This results in a continuous curve, with a discontinuous derivative (in general), thus of differentiability class C^0.

$$R_T = f(x) - p(x)$$

where p denotes the linear interpolation polynomial defined above

$$p(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0)$$

It can be proven using Rolle's theorem that if f has a continuous second derivative, the error is bounded by

$$|R_T| \leq \frac{(x_1 - x_0)^2)}{8} \max_{x_0 \leq x \leq x_1} f''|(x)|$$

The approximation between two points on a given function gets worse with the second derivative of the function that is approximated.

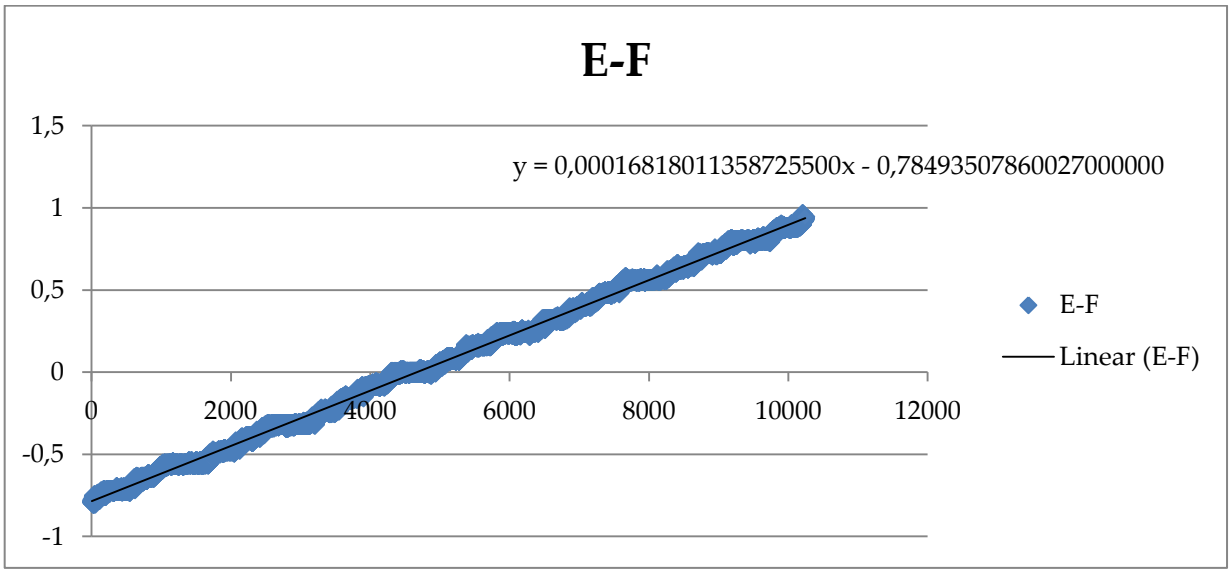


Figure 90: Moving average (blue) and linear interpolation (black)

4.12.4. Results

The estimated latency of our system was measured to be 90ms ±10% before minimization processes were applied as described below, inclusive of the latency induced by the refresh rate of the screen.

After disabling added latency sources described in section 2.4.2, the end-to-end latency of our system was measured again using the same measurement technique described above. The new estimated latency of our VE was measured to be slightly below 50ms ±10%, a reduction of almost 50%, inclusive of the latency induced by the refresh rate of the screen. The estimated latency and the latency reduction is less or comparable to previous work (Hill, Adelstein and Ellis 2004), in this case using a more complex environment of high polygon count, accurate measurements via an oscilloscope and a custom-made, low-cost, portable system.

4.13. Chapter Summary

I this chapter we presented the hardware and software setup for the latency measurement mechanism presented in this thesis. We analyzed the digital circuit components and setup of the measurement mechanism and introduced the Intermediate DLL API used for interoperability between our head tracker hardware and the VE application.

CHAPTER 5. Latency Experiment

5.1. The 3D Scene for Latency Experiments

In order to explore whether the cognitive impact of latency is severe for spatial awareness or whether there is adaptation to latency occurring, we investigated the effect of latency on 3D spatial cognition, spatial awareness states and 3D mental models and imagery.

A 3d scene depicting an apartment was created. The 3d apartment scene included 3 sub-areas, the office sub-area, the kitchen sub-area and the lounge sub-area. The 3d scene contained objects both consistent and inconsistent to each sub-area. In order to define the type and the degree of consistency of objects which could be found in an apartment scene results from a previous study (Zotos, Mania and Mourkoussis 2009) were used. This study (Zotos, Mania and Mourkoussis 2009), in order to define which objects were consistent and which objects were inconsistent in relation to the context of the scene (type of consistency), a set of questionnaires were designed that contained a list of objects asking participants to *"Rate each object for how likely the object would to be appear in a room like this."* The methodology was similar to the one described in the (Brewer and Treyens 1981) paper and pilot questionnaires were created containing a list of objects related to three sub-areas in the apartment of this (Zotos, Mania and Mourkoussis 2009) study:

- An office area
- A lounge area
- A kitchen area

5.1.1. Creating the Scene

An eight by eight meters virtual house was chosen as the rendered displayed environment, divided in four zones according to the experiment's specifications. The four zones designed was the lounge, office and kitchen area.

3D models were created or downloaded from 3D models' repositories and were placed in a scene with the help of an industry-standard 3D modeling software

(Autodesk 3ds Max 2011). The final result can be seen in Figure 95, which depicts the scene in lit mode (after calculating the lighting space of the scene).

5.1.2. Radiosity Solution

As described in a previous section radiosity calculates diffuse reflections in a scene and results in a finely divided geometrical mesh. Heat transfer theory describes radiation as the transfer of energy from a surface when that surface has been thermally excited. This encompasses both surfaces that are basic emitters of energy, as with light sources and surfaces that receive energy from other surfaces and thus have energy to transfer.

The radiosity algorithm was utilized in order to simulate light propagation and render the scene. In order to apply the radiosity solution, the following parameters should be defined:

- *Number of iterations:*

  The maximum number of radiosity iterations. The radiosity engine bounces rays around the scene and distributes energy on surfaces. Between the iterations, the engine measures the amount of variance (noise between surfaces) that was computed. As the number of algorithmic iterations of surface light propagation increases, it improves the radiosity shading accuracy and polygon count. For our solution we used 3 iterations.
- *Minimum and maximum mesh size:*

  Defines the minimum mesh size that faces are not divided smaller than. Maximum mesh size is the size of the largest faces after adaptive subdivision. For our solution we used maximum mesh of 0.4m and minimum mesh of 0.01m.
- *Initial quality (%):*

  This parameter sets the quality percentage at which to stop the Initial Quality stage, up to 100%. For example, if the initially quality is set to 80%, the result is a radiosity solution that is 80% accurate in relation to total energy

distribution. A quality of 80 to 85% is usually sufficient for good results. For our solution we used 85%.

- *Global subdivision settings:*

  This parameter turns on the radiosity mesh for the entire scene.

- *Regather indirect illumination:*

  In addition to recalculating all the direct lighting, the algorithm recalculates the indirect lighting at each pixel by regathering illumination data from the existing radiosity solution. Using this option, the most accurate, artifact-free images can be produced, adding, however, a considerable amount of rendering time.

- *Photometric lights:*

  Photometric lights use photometric (light energy) values that enable the user to more accurately define lights as they would be in the real world. The user can create lights with various distribution and color characteristics, or import specific photometric files available from lighting manufacturers.

Radiosity algorithms display view-independent diffuse inter-reflections in a scene assuming the conservation of light energy in a closed environment. The surfaces of objects are divided into patches or elements. Despite transmitting energy to others, a patch will also reflect the energy from other meshes that arrives on its surface into the scene. These processes will be iterated until energy equilibrium in the closed space is achieved. Radiosity produces color-bleeding effects from one surface to another, shades inside the shadow area and creates soft-edge shadow with penumbrae along shadow boundaries. All of these results imitate the physical propagation of light in the real environment. The number of algorithmic iterations of surface light propagation as increases improves the radiosity shading accuracy and polygon count.

Figure 92 (right) and Figure 95 show the scene after applying the radiosity solution compared to the initial geometry used as input (Figure 91 and Figure 93) and the direct illumination, flat shaded rendering of the scene in Figure 92 (left) and Figure 94.

Figure 91: Top view of the experimental scene, without shading

Figure 92: Flat shaded version of the experimental scene (left) vs. the radiosity solution (right) (top view)

Figure 93: Rendering of experimental scene with no shading

Figure 94: Flat shaded rendering of the experimental scene

Figure 95: Radiosity solution rendering of the experimental scene

5.1.3. Exporting geometry for XVR

XVR studio offers plugins for exporting the geometry from the 3d modeling applications to the AAM file format.

AAM (once the acronym of Ascii Animated Mesh, now waiting for a better meaning) is the XVR native format for the description of triangular meshes. Its features are:

- optimized for loading times in XVR
- 1:1 correspondent to XVR data structures
- available in ASCII (to easily allow inspection and manual modifications) or binary (for faster loading)
- support to multitexturing, animation, smoothing groups, user properties, shaders, skinning
- exporter plugins for 3D Modeling Applications

Scenes created with 3d modelers may be provided with a highly realistic precomputed lighting, as off-line rendering can make use of lighting models much more sophisticated of those suitable for real-time. It is anyway possible to export

AAM models preserving this rendering information, by "rendering on textures". Basically some textures are created which may be:

- Complete maps (material + diffuse map + lighting map, combined in one single texture layer)
- Lighting maps (which result in two texture layers, the "traditional" diffuse map + lighting map)

For our VE the following procedure was used:

- The whole 3d scene was selected and selected and was "Rendered to Texture" item from the Rendering menu.
- In the "Output" frame, a complete map on the diffuse channel texture map type was generated
- Shadows, Direct Light and Indirect Light were turned On
- Map size of 256x256 was used for baked textures of most of the objects except from the walls the ceiling and the floor were maps size of 2048x2048 was used
- The 3d application generated baked texture with the TIFF picture format. The TIFF textures were converted to JPEG textures in order to be used with the XVR application.
- Geometry of the scene was exported to the AAM file format using the XVR AAM exporter plugin.

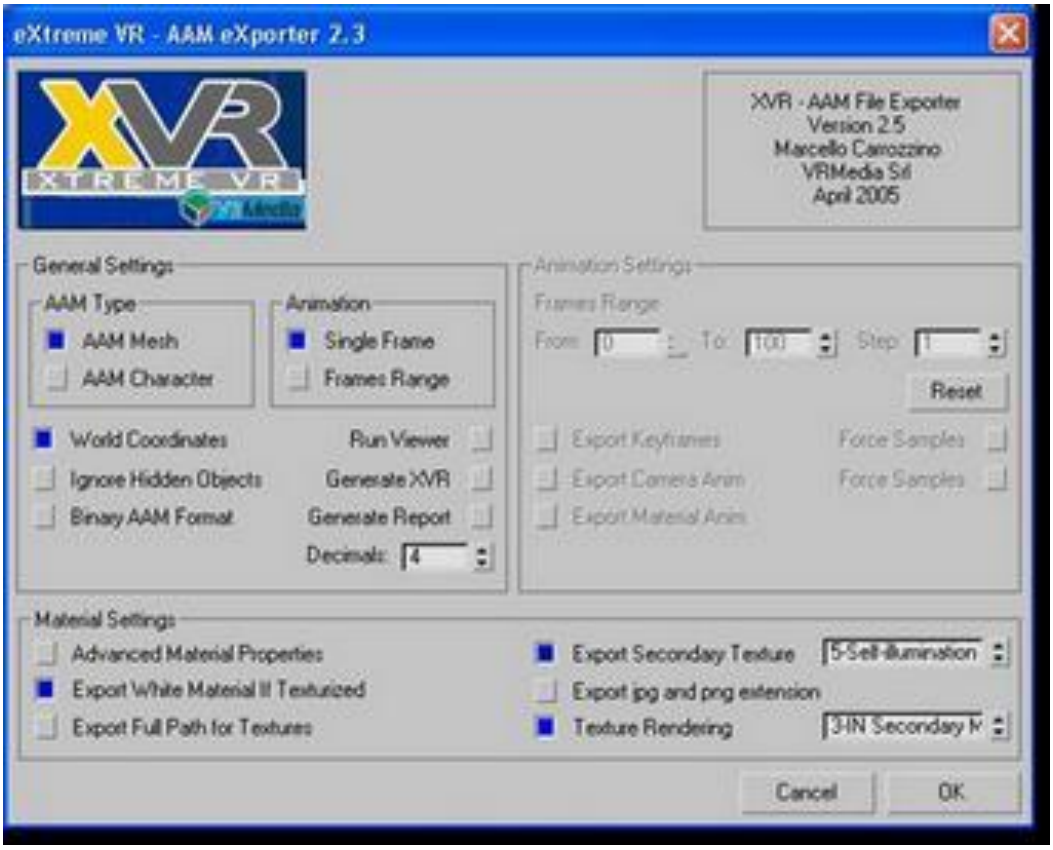


Figure 96: The AAM exporter plugin window

5.2. Pilot studies

In order to determine the appropriate number of objects contained in the VE and the memory task and the appropriate exposure time of the participants in the main experiment scene, a pilot study was conducted in October 2011. 20 participants belonging to the research population of the Technical University of Crete were recruited, with their age ranging from 18-28.

Two versions of the same VE scene of an apartment consisting of 3 sub-areas (kitchen, office, lounge) described in 5.1 were used in the pilot studies. The first version contained 30 objects, 10 placed at each of the three sub-areas and the other contained 24 objects, 8 placed at each of the sub-areas. In both versions half of the objects placed at each of the sub-areas were consistent with the context of each sub-area and the other half were inconsistent (see 5.1).

Participants were exposed to a Virtual Environment setup close to the setup of the main study to follow except of the 2 latency conditions (instead of 3).

From the beginning of the pilot study it was clear that the participants were having difficulties in the memory placement task in the 30-objects version scene, therefore we gave up this version and continued the pilot studies with the 24-objects version.

The remaining group of participants were exposed to the scene with the 24 objects which was decided to be used to the subsequent main experiment.Preliminary analysis was conducted within the results of this group.

5.2.1. Preliminary Results

The accuracy of memory was measured by counting the number of correct positions of objects (out of a possible 24). Awareness state data was considered in terms of prior probabilities. Prior probabilities reflect on the following: "Given that the response of a participant is correct (correct placement of object), what is the probability that the participant has chosen a particular awareness state?" Prior probabilities were obtained by calculating the proportions of correct answers falling in each of the three memory awareness categories for each participant.

Total Correct

The total number of objects that were identified in the correct location was counted for each participant (Table 5).

Table 5: Number of correct responses and standard deviations as a function of viewing condition (no latency, high latency) and schema consistency (consistent, inconsistent)

	No latency (n=4)		**High latency (n=4)**	
	Consistent	**Inconsistent**	**Consistent**	**Inconsistent**
Total correct (out of 24)	7.00 (3.92)	6.75 (3.40)	5.75 (1.71)	5.50 (3.70)

Trends in the data indicate that, of these participants, more items were being correctly recalled in the correct location with no latency (M= 6.88) than with hi latency (M= 5.63). This seemingly does not depend upon whether the objects are consistent or inconsistent. These pilot data are based on a small number of participants (n=4) and are at this stage inappropriate for further parametric statistical analysis. The reliability of this trend will be verified using a 2x2 mixed analysis of variance (ANOVA) with viewing condition (no latency, hi latency) entered as a between subjects variable and the context consistency of the objects (consistent, inconsistent) entered as a within subjects variable in the main experiment.

Confidence

Confidence reports *(No confidence, Low confidence, Moderate confidence, Confident, Certain)* were converted to numerical values ranging from 1 assigned to *'No confidence'* and 5 assigned to *'Certain'*. Mean values are presented in Table 6.

Table 6: Mean confidence rating and standard deviation as a function of viewing condition (Hi latency, No latency) and context consistency (consistent, inconsistent)

	No latency (n=4)		**High latency (n=4)**	
	Consistent	**Inconsistent**	**Consistent**	**Inconsistent**
Confidence (5-point scale)	3.52	3.56	2.77	3.35
	(.58)	(.50)	(.46)	(.71)

Trends in the data indicate that, of these participants, confidence ratings were slightly higher for responses to inconsistent objects (M=3.46) than consistent objects (M=2.90), and that confidence ratings were slightly higher in the no latency condition (M=3.54) than the hi-latency condition (M=2.81). Importantly, there are suggestions of

an interaction, with confidence ratings generally lower in the hi-latency condition for consistent objects (M=2.27) than any other.

As per the above section, these pilot data are based on a small number of participants (n=4) and are at this stage inappropriate for further parametric statistical analysis.

Awareness states

The proportion of correct responses assigned to each awareness state is displayed in **Table 3**.

Table 7: Proportion of correct responses and standard deviations as a function of viewing condition (Hi latency, No latency), context consistency (consistent, inconsistent) and reported awareness state (Type A, Type B, Guess)

	No latency (n=4)		**High latency (n=4)**	
	Consistent	**Inconsistent**	**Consistent**	**Inconsistent**
TYPE A	0.55 (.46)	.67 (.34)	.24 (.17)	.77 (.30)
TYPE B	0.10 (.21)	.33 (.34)	.14 (.10)	.19 (.32)
Guess	0.34 (.45)	.00 (.00)	.63 (.38)	.04 (.07)

Trends in the data indicate that, of these participants, more inconsistent objects were associated with a remember response (M= 0.72) than consistent objects (M=0.39), and a similar pattern is found for *'know'* responses (M=.26 vs. M=0.12). Naturally, consistent objects were therefore associated mainly with guess responses (M=0.48) compared to guess responses for inconsistent objects (M=0.02).

In general there are no indications in this data set that the proportion of remember responses differ greatly between the no latency condition (M=0.55) and the hi latency condition (M=0.49), with similar indications with the proportion of know responses in the no latency condition (M=0.10) and the hi latency condition (M=0.12),

as well as the proportion of guess responses in the no latency condition (M=0.34) and the hi latency condition (M=0.38).

However, there appears to be the first signs of an interaction between the latency condition (no latency, high latency) and the object consistency (consistent, inconsistent) across the three reported awareness states. In particular, there are a disproportionately large proportion of guess responses for consistent objects in the high latency condition (M=0.63), and correspondingly a disproportionately low proportion of remember responses for consistent objects in the same latency condition (M=0.24).

These pilot data are based on a small number of participants (n=4) and are at this stage inappropriate for further parametric statistical analysis. The reliability of these trends will be verified with a series of $2x2$ mixed analysis of variance (ANOVA) for each awareness state with viewing condition (no latency, high latency) entered as a between subjects variable and the context consistency of the objects (consistent, inconsistent) entered as a within subjects variable in the main experiment.

Discussion of pilot data

The preliminary analyses of the pilot data indicated that latency in the VE simulation may influence the accuracy of memory for objects in that environment. The correct objects and their locations may be remembered more accurately when there is no latency than when there is high latency. The pilot data also indicated that object consistency with the visual scene may have an influence too. Interestingly, the proportion of correct responses that had a vivid *'remember'* experience was greater when the objects were inconsistent with the environment, than when they were consistent. This appears to potentially interact with latency, with a disproportionate number of correct responses to consistent objects in the high latency condition being associated with guesses. Confidence scores interacted with latency and object consistency in a similar way, with lower confidence scores for responses to consistent objects in the high latency condition.

It has been noted previously by some of these authors that pure accuracy measurements are an imperfect measure of the memorial experience in VE simulations. This has to some extent been predicated by consistently high performance on accuracy tasks that is underpinned by differential patterns in actual memorial experience (Mania, Troscianko, et al. 2003), (Mania, Badariah and Coxon 2010), (Bennett et al. 2010). In this initial pilot exploration there are some suggestions that, unlike previous variations, variations in latency may impact upon overall accuracy. This can be explained simply as additional perceptual processing resources that may have been dedicated to interpreting and updating the internal mental scene as a result of the latency can instead be redistributed to processing the objects within it. That is to say, navigating a world in which there is no latency may minimize the perceptual resources needed to cope with this unnatural motion and instead allow these to be re-distributed to other perceptual tasks such as object recognition.

In terms of the memorial experiences that underpin these recollections, past studies have indicated that low visual fidelity environments, or low interactivity, may be more attentionally demanding because of their novelty or variation from 'real' resulting in more vivid remember responses (Mania, Badariah and Coxon 2010). That is to say that deviation from 'real' may capture attention. This is typified in the current experiment in conditions where the objects are not ones you might expect in the environment (inconsistent objects) for which there are clear indications that this may lead to more vivid 'remember' experiences of seeing them in the VE simulation. Potentially of more interest are the measurements of memorial experiences associated with inconsistent objects when there is a high latency. This combination of conditions is potentially the least consistent with reality in that the interactivity, the latency, and the objects are inconsistent with reality. Interestingly this combination produced the highest proportion of 'remember' responses in this exploratory data set which is consistent with the attentional hypothesis that has been put forward based upon consistency with reality. More broadly, the suggestion is tentatively supported that vivid recollective experiences occur more frequently when there is a match between the novelty of the object being remembered and the novelty

of the environment it is in. That is, that objects and their environments are processed in an interactive way that is determined by consistency (Davenport and Potter 2004).

Nevertheless, the present data also pose an interesting challenge for interpretation. There are initial indications that participants had particular difficulty with consistent objects in the hi-latency condition, with generally lower accuracy, lower confidence ratings and a disproportionate amount of guess responses. One possibility is that interacting with a high latency VE simulation is particularly demanding of perceptual processing resources, such that any remaining resources are devoted to processing and interpreting objects that 'pop-out' by varying from reality at the expense of interpreting those that are consistent. This would suggest at least two stages at which attentional demands may influence processing of objects in similar VE simulations. The first stage may be based upon additional processing demands that arise from the VE environment. If these demands are interactive (e.g. latency) then these make more demands of processing resources than those that are less interactive (e.g. radiosity). The second stage then makes use of the remaining processing resources. Where these are novel aspects of the environment that vary from 'real' may receive more attention than those that are consistent. If sufficient resources are available then both novel and non-novel items may be attended to for processing. This interpretation if of course tentative and rests on a number of assumptions that would require further testing if this result was found with a larger sample size. A larger sample size, though, could eliminate such effects and showcase that perceptual adaptation is occurring in a manner that subjects adapt to added latency and communicate similar object recognition performance irrespectively of the presence of high latency levels. We describe the main complete formalized experiment in the following section.

Our understanding of how such processes work within fully immersive environments, such as those that VEs provide, is only now beginning to be explored and it is possible, indeed likely, that there will be differences between real-world experiences and simulated scenes. In any case, the pilot study results presented here stimulate a number of considerations for further testing when the full-scale

experiments are conducted with the appropriate number of participants and parametric statistical manipulation of data is possible. There is some indication that a high saliency environment may have a profound effect upon memory for the objects and their locations within it.

5.3. The Main Experiment

5.3.1. Apparatus

The VEs were presented in stereo XGA resolution (2 channels of XGA (1024*768) resolution) on a Kaiser Electro-Optics Pro-View 50 Head Mounted Display with a Field-of-View comprising 50 degrees diagonal. An Intersense Intertrax2, three degree of freedom tracker was utilized for rotation. The viewpoint was set in the middle of the virtual room and navigation was restricted only to freely circle around that viewpoint (yaw) and to 180 degrees vertical head rotation (pitch). Participants were sitting on a swivel chair during exposure. The application ran on a standard PC with an average cost graphics card. Participants weren't allowed to move forward/backward and navigate through the scene.

5.3.2. Participants

60 participants (Figure 97) were recruited belonging to the research population of the Technical University of Crete, their age ranging from 18-28. The 60 participants formed 3 balanced for age and gender, groups of 20, corresponding to the three latency conditions. Participants in all conditions were naive as to the purpose of the experiment. All participants had normal or corrected to normal vision and no reported neuromotor or stereovision impairment. The experimental VE was set up in a dedicated experimental space on campus, which was darkened to remove any periphery disturbance during the exposure.

Figure 97: Photo of participant

5.3.3. Visual Content

The original scene was photorealistically illuminated using pre-computed radiosity textures and stereoscopically rendered, using XVR's side-by-side stereoscopic rendering feature. The VE represented a room as shown in Figure 98. The radiosity-rendered space was divided in three zones including a kitchen/dining area, an office area and a lounge area. The space was populated by objects consistent as well as inconsistent with each zone's context. Four consistent objects and four inconsistent objects populated each zone resulting in 24 objects located in the scene overall, 8 in each zone. The polygon count of the scene was ~180,000 polygons.

The between-subjects factor was *"minimized System Latency"* vs. "standard latency" vs. *"400ms added to minimum system latency"* and the within-subjects factor

was *"Context specific"* vs. *"Inconsistent objects"*. According to the experimental group that they were assigned to, participants completed a memory recognition task including self-report of spatial awareness states and confidence rating for each recognition after exposure to one out of the three experimental conditions.

- Minimized System Latency, referred as "Low Latency": A stereo-rendered radiosity simulation of a scene displayed on a stereo head-tracked HMD including consistent as well as inconsistent objects in each zone. The tracking latency utilized was the minimum system latency of around 50ms.
- Elevated Latency, A stereo-rendered radiosity simulation of a scene displayed on a stereo head-tracked HMD including consistent as well as inconsistent objects in each zone. The tracking latency utilized was around 90ms.
- 400ms latency added to Minimum System Latency, referred as "Hi Latency": A stereo-rendered radiosity simulation of a scene displayed on a stereo head-tracked HMD including consistent as well as inconsistent objects in each zone. The tracking latency utilized was the minimized system latency of around 50ms with added latency of approx. 400ms.

The experimental scene in all latency conditions consisted of the so-called *"Room frame"* objects: walls, floor, ceiling and doors. It also included standard objects such as desks, dining table, chairs, shelves etc. According to (Brewer and Treyens 1981), "the room frame contains the information about rooms that one can be nearly certain about before encountering a particular room". As mentioned the scene was populated by four consistent objects in each zone as well as four inconsistent objects for each zone. The list of objects was assembled based on an initial pilot study which explored which objects were expected to be found in each area and which were not (Zotos, Mania and Mourkoussis 2009). According to this study, 25 participants ranked the objects on the list. The consistency of each item was rated on a scale from 1 to 6 according to whether each object was expected to be found in each area or not, with 6 being the most expected, and 1 being the least. Based on these ratings, consistent objects were selected from the high end of the scale, and the inconsistent ones from the low end.

The objects were distributed over locations indicated in a testing blueprint similar to the one presented in Figure 98. Participants were required to select from a recognition list provided which object was present in each location. All twenty-four objects positioned in the house scene as follow:

Kitchen Area				
Consistent	Mixer	Saucepan	Toaster	Fruits
Inconsistent	Bike wheel	Shovel	Calculator	Baseball bat

Lounge Area				
Consistent	Remote Control	Flower vase	Ashtray	Magazine
Inconsistent	Tennis racket	Hammer	Warrior Helmet	Basketball

Office Area				
Consistent	Pencils	Laptop	Books	Printer
Inconsistent	Trumpet	Tennis ball	Cashier Machine	Sword

Figure 98: The experimental VE scene

5.3.4. Procedure

The experiment was set to be completed in three stages by each participant. Initially, a preliminary training phase took place requiring participants to wear the HMD device. During this stage, all appropriate adjustments were conducted accordingly for each individual. After participants familiarized themselves with the device and the stereoscopic VE then the second stage of the experiment was initiated which was the main phase of the experiment. The third stage took effect when participants finished the main experiment, requiring them to complete an online memory recognition questionnaire.

The experiments were conducted during May 2012. Participants were led to believe that the main experiment was just a practice phase before the actual main experiment took place, aiming to familiarize them with the HMD (see APPENDIX A for a detailed description). The reason for this was to prevent the participants from being aware of the experimental task prior to exposure and avoid the development of mnemonics. Participants were given identical instructions across conditions. The room where the experiment was taking place was kept dark during exposure.

The Inter Pupillary Distance (IPD) of each participant was measured prior to exposure and the stereo application's parallax was adjusted accordingly for each individual. All participants had time to feel comfortable with the apparatus and 3D environments during the practice phase. A practice scene was created which included 3D primitive shapes such as boxes and cylinders (Figure 99) and participants were let to look around the scene just like the main phase. After the practice phase, the main scene of the experiment was loaded while participants were still led to believe that this was just a practice phase thus, they were not aware of the experimental task to follow. Participants were instructed to look around the room at their own pace and to examine it in all directions for 210 seconds of exposure either to the low, the elevated or the high latency condition. The time of exposure was defined after detailed pilot studies which ensured that there were no apparent floor (the task being too hard) or ceiling (the task being too easy) effects.

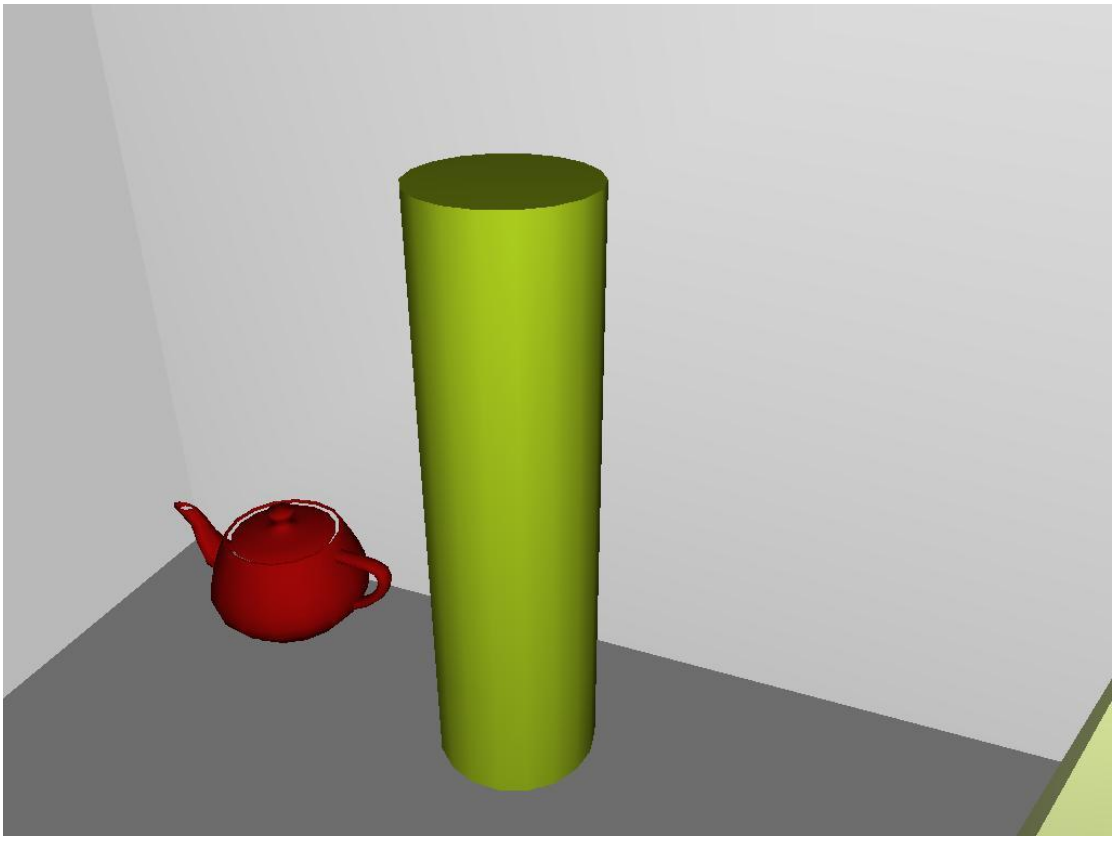


Figure 99: Simple 3d pattern scene used for calibration.

After the exposure, participants were led to another room and asked to complete a memory recognition questionnaire (APPENDIX B). Participants were not informed that they would subsequently complete a memory task. The questionnaires were administered within 1 minute after VE exposure.

A top view of the bare environment was provided including 24 numbered vacant object positions, 8 positions for each of the three sub-areas, in which an object had been present (Figure 100). In the memory recognition test administered, participants were required to select which object they considered they saw during exposure in each numbered position, selecting objects from an object recognition list as well as one out of 5 levels of confidence: *No confidence, Low confidence, Moderate confidence, Confident, Certain,* and two choices of awareness states: *TYPE A, TYPE B (description below)*. A recognition list was devised including a list of objects per scene zone. Each zone included in alphabetical order the eight present objects as well as

eight absent objects (four inconsistent and four consistent) in each zone. The list included a total of 48 objects;

- Twelve consistent objects that were present (4 at each sub-area)
- Twelve consistent objects that were absent (4 at each sub-area)
- Twelve inconsistent objects that were present (4 at each sub-area)
- Twelve inconsistent objects that were absent (4 at each sub-area)

Prior to the memory recognition task, awareness states were explained to the participants in the following terms:

- TYPE A means that you can recall specific details. For example, you can visualize clearly the object in the room in your head, in that particular location. You virtually 'see' again elements of the room in your mind, or you recollect other specific information about when you saw it.
- TYPE B means that you just 'know' the correct answer and the alternative you have selected just 'stood out' from the choices available. In this case you can't visualize the specific image or information in your mind.

Figure 100: The bare environment top view

The list of objects was assembled based on an initial pilot study which explored which objects were expected to be found in each area and which were not. 25 participants ranked the objects on the list. The consistency of each item was rated on a scale from 1 to 6 according to whether they expected to find each object in each area, or not with 6 being the most expected and 1 being the least. Based on these ratings, consistent objects were selected from the high end of the scale and the inconsistent ones from the low end.

5.3.5. Simulator Sickness

Simulator sickness which is a potential side effect of all HMDs in general, has been observed during the experiment presented here too. Participants reported several symptoms related to simulator sickness such as fatigue, headache, dizziness, visual discomfort, and nausea. There also exist other indirect effects of VEs on the visual system such as eyestrain, changes in binocular vision and visual acuity, balance, nausea, and motion sickness. Participants experienced the aforementioned symptoms to varying degrees. Various articles exist in related literature focusing on possible causes of simulator sickness such as system latency (DiZio, Lackner and Matin 1997), (Cobb, et al. 1999), limited Field-of-View (DiZio, Lackner and Matin 1997), Image scale factor (Draper, et al. 20001), etc. Apart from the previous factors that provoke simulator sickness, other aspects of the HMD that contribute to participant discomfort exist. The HMD itself weighted 1 Kg making some participants uncomfortable during the experiment. Additionally, as a result of improper adjustment of the Interpupillary Distance (IPD) participants perceive dissimilar imagery from their eyes. Nevertheless, the experiments reported here were conducted without any participant interrupting the procedure because of simulator sickness.

5.4. Statistical Analysis

This section presents the basic statistical principles employed in order to analyze the acquired memory recognition self-report.

5.4.1. Analysis of Variance

In statistics, ANalysis Of VAriance (ANOVA) is a collection of statistical models, and their associated procedures. ANOVA procedures are powerful parametric methods for testing the significance of the differences between sample means where more than two conditions are used, or even when several independent variables are involved (Coolican 1999). ANOVA is used to compare the variance between the two groups with the variability within each of the groups. This comparison is in the form of a ratio known as the F-test. A high value for F indicates a strong effect, i.e. the variance between groups is higher than the variance within the groups. The strength

of the effect is given by the p value. The p value represents the probability that there is no between groups variance. This is called the null hypothesis and is disproved if a value of p below 0.05 is returned.

5.5. Results and Discussion

5.5.1. Total Correct (Hits)

The total number of objects that were identified in the correct location were summed for each participant.

Table 8

	Hi Latency (n= 20)		**Mid Latency (n=20)**		**Low Latency (n=21)**	
	Consistent	**Inconsistent**	**Consistent**	**Inconsistent**	**Consistent**	**Inconsistent**
Total correct (out of 24)	5.10 (2.29)	5.90 (2.38)	5.40 (2.68)	6.40 (2.98)	5.81 (1.78)	6.62 (3.04)

The total number of objects identified in the correct location (displayed in Table 8) was analyzed using a 2x3 mixed ANOVA. Latency (hi, mid, lo) was entered as a between subjects variable, with the context consistency of the objects (consistent, inconsistent) entered as a within subjects variable. A minimum alpha level of .05 was used throughout the analyses to judge a reliable difference.

No reliable main effects of latency were found ($F(2,58)=0.54$, $p>.05$) and no interaction was found between latency and the context consistency of the objects ($F(2,58)=0.04$, $p>.05$). However, there was a main effect of the context consistency of the objects ($F(1,58)=6.85$, $p<.05$, partial eta-squared = 0.11). More objects were reported in the correct position when they were inconsistent with the context (Mean = 6.31, SD = 2.79) compared when to objects that were consistent with the context (Mean = 5.44, SD = 2.25).

5.5.2. Total Misses – Present Objects

The total number of objects that were incorrectly placed, but had been present in the room, were summed for each participant.

Table 9

	Hi Latency (n= 20)		**Mid Latency (n=20)**		**Lo Latency (n=21)**	
	Consistent	**Inconsistent**	**Consistent**	**Inconsistent**	**Consistent**	**Inconsistent**
Total misses (present)	3.05 (1.57)	3.65 (2.43)	3.30 (2.54)	2.45 (1.90)	3.57 (1.54)	2.90 (2.00)

The total number of misses for present objects (displayed in Table 9) was analyzed using a 2x3 mixed ANOVA. Latency (hi, mid, lo) was entered as a between subjects variable, with the context consistency of the objects (consistent, inconsistent) entered as a within subjects variable. A minimum alpha level of .05 was used throughout the analyses to judge a reliable difference. No reliable main effects of latency ($F(2,58)=0.50$, $p>.05$), context consistency of the objects ($F(1,58)=0.87$, $p>.05$), or an interaction between the two were found ($F(2,58)=1.92$ $p>.05$).

5.5.3. Total Misses – Absent Objects

The total number of objects that were selected which had not been in the room was summed for each participant.

Table 10

	Hi Latency (n= 20)	**Mid Latency (n=20)**	**Lo Latency (n=21)**

	Consistent	Inconsistent	Consistent	Inconsistent	Consistent	Inconsistent
Total misses (absent)	3.00 (2.36)	1.71 (1.23)	2.80 (2.04)	1.55 (1.35)	3.09 (1.34)	1.30 (1.03)

The total number of misses for absent objects (displayed in Table 10) was analyzed using a 2x3 mixed ANOVA. Latency (hi, mid, lo) was entered as a between subjects variable, with the context consistency of the objects (consistent, inconsistent) entered as a within subjects variable. A minimum alpha level of .05 was used throughout the analyses to judge a reliable difference.

No reliable main effects of latency were found ($F(2,58)=0.22$, $p>.05$) and no interaction was found between latency and the context consistency of the objects ($F(2,58)=0.35$, $p>.05$). However, there was a main effect of the context consistency of the objects ($F(1,58)=41.19$, $p<.001$, partial eta-squared = 0.42). Absent objects consistent with the context were incorrectly chosen more often (Mean = 2.97) than absent objects that were inconsistent with the context (Mean = 1.52).

5.5.4. Confidence

Confidence reports (No confidence, Low confidence, Moderate confidence, Confident, Certain) were converted to numerical values ranging from 1 assigned to 'No confidence' and 5 assigned to 'Certain'. Confidence for correct responses only was analyzed. One participant was removed from the dataset due to incomplete data.

Table 11

	Hi Latency (n= 20)		Mid Latency (n=19)		Lo Latency (n=21)	
	Consistent	**Inconsistent**	**Consistent**	**Inconsistent**	**Consistent**	**Inconsistent**

Total correct (out of xxx)	3.11 (0.87)	3.70 (1.04)	3.56 (0.95)	3.85 (0.72)	3.49 (0.74)	3.86 (0.72)

The average confidence ratings for correct responses (displayed in Table 11) were analyzed using a 2x3 mixed ANOVA. Latency (hi, mid, lo) was entered as a between subjects variable, with the context consistency of the objects (consistent, inconsistent) entered as a within subjects variable. A minimum alpha level of .05 was used throughout the analyses to judge a reliable difference.

No reliable main effects of latency were found ($F(2,57)=1.11$, $p>.05$) and no interaction was found between latency and the context consistency of the objects ($F(2,57)=0.55$, $p>.05$). However, there was a main effect of the context consistency of the objects ($F(1,57)=11.01$, $p<.005$, partial eta-squared = 0.16). Higher levels of confidence were reported when the correctly identified objects were inconsistent with the context (Mean = 3.80) compared to when objects that were consistent with the context were chosen (Mean = 3.39).

5.5.5. Awareness States

Table 12

	Hi Latency		**Mid Latency**		**Lo Latency**	
	Consistent	**Inconsistent**	**Consistent**	**Inconsistent**	**Consistent**	**Inconsistent**
TYPE A	0.29 (0.15)	0.43 (0.18)	0.33 (0.14)	0.39 (0.19)	0.31 (0.12)	0.41 (0.16)
TYPE B	0.17 (0.12)	0.12 (0.13)	0.15 (0.15)	0.13 (0.15)	0.17 (0.10)	0.10 (0.10)

The proportion of the total correct responses (displayed in Table 12) were analyzed with two separate 3x2 mixed ANOVAs for each awareness state (Type A, Type B). Latency (hi, mid, lo) was entered as a between subjects variable, with the context consistency of the objects (consistent, inconsistent) entered as a within subjects variable. A minimum alpha level of .05 was used throughout the analyses to judge a reliable difference.

No reliable main effects of latency were found for either Memory A awareness states ($F(2,58)=0.004$, $p>.05$) or Memory B awareness states ($F(2,58)=0.004$, $p>.05$). Similarly, no interactions were found between latency and the context consistency of the objects for either Memory A awareness states ($F(2,58)=1.42$, $p>.05$) or Memory B awareness states ($F(2,58)=0.54$, $p>.05$). However, there was a main effect of context consistency on Memory A awareness states ($F(1,57)=9.08$, $p<.005$, partial eta-squared = 0.14) and a main effect of context consistency on Memory B awareness states ($F(1,58)=4.44$, $p<.05$, partial eta-squared = 0.07). When objects were correctly recognised in the correct location, a higher proportion of these correct responses were reported as Memory A awareness state (remember) with inconsistent objects (Mean = .41) compared to with consistent objects (Mean = .31). Conversely, participants reported a higher proportion of correct responses as Memory B awareness states (know) with consistent objects (Mean = .16) compared to with inconsistent objects (Mean = .12).

The results of the experiment indicate the following

- There is a statistical difference between the type of response people gave, participants tended to respond that they had a vivid memory when they correctly placed an object (Mem A, Mean = 4.42) rather than a feeling of knowing (Mem B, Mean = 1.46).
- There is a statistical difference between the category of object also, participants tended to correctly place objects that were inconsistent (Mean = 3.15) compared to objects that were consistent (Mean = 2.72).

- There was also a significant interaction between the category of the object and the type of response. A higher number of vivid memories were reported with inconsistent objects (Mean = 5.06) than consistent objects (Mean = 3.77).

The latency manipulation is not making any difference to any part of the analysis. Despite the indications in the pilot study, analyses of the full study haven't revealed any influence of the latency manipulation. This means, that there is adaptation occurring of users to latency which results in similar object memory recognition in both latency conditions. However, these data do demonstrate an interaction between the memorial experience people are having (Mem A/Mem B) and the type of object (consistent/inconsistent).

It is also worth noting that the data for the 'awareness states' analysis were not normally distributed. If you more stringent criteria are applied to judge significance (e.g. at least $p<0.01$) won't change the results much but would remove any difference between consistent and inconsistent objects in terms of Memory B type responses.

5.6. Chapter Summary

This chapter presented an evaluation study of the effect of head tracking latency in spatial cognition in immersive simulations. The task utilized for this evaluation is memory recognition, drawn from formalized methodologies of memory psychology.

In order to evaluate spatial awareness while exposed to a rendered scene with 3 different levels of latency; minimized latency, medium latency and high-latency level. A scene was designed which included several areas and sub-areas and a memory recognition task was completed. The scene was designed so that it contains objects with differing levels of scene consistency based on the area they were placed.

Furthermore, the results of the research project have been presented and discussed in this chapter. The following chapter draws conclusions from the findings, discusses limitations of the current research and makes recommendations for the future.

CHAPTER 6. Conclusion

6.1. Main Contributions

In this thesis we presented a custom-made mechanism of measuring and minimizing end-to-end head tracking latency in an immersive VE. Our mechanism builds on previous mechanisms by using an oscilloscope to compare two signals, assembled by low-cost, custom-made and portable equipment. One signal is generated by the head-tracker movement and reported by a shaft encoder attached on a servo motor moving the tracker. The other signal is generated by the visual consequences of this movement in the VE and reported by a photodiode attached to the computer monitor. The end-to-end head tracking latency of the VE is the measured time-shift between these two signals. The presented system calculates this time-shift by off-line processing the tracker position and display brightness measurements stored in a computer t derived from the oscilloscope using a USB connection. Thus, an accurate measuring mechanism is provided, utilizing equipment commonly found in an academic facility. Subsequent software reorganizations to the VE system result in the reduction of the overall system latency resulting to a VE with minimal end-to-end head-tracking latency.

The utility of simulation environments for training, such as flight simulators, or collaborative 3D design, as well as remote tele-operation manipulations is predicated upon the accuracy of the 3D spatial representation formed mentally. Spatial awareness is essential for human performance efficiency of tasks requiring spatial knowledge of an environment (Mania, Badariah and Coxon 2010). A central research issue therefore for such simulations is how participants cognitively represent 3D spatial elements and how their memory and recognition of such worlds corresponds to real world conditions.

The system presented in this thesis was used to investigate the effect of latency levels, ranging from the minimum system latency to added latency, on spatial awareness states. The main premise of future work is that accuracy of memory performance per se is an imperfect reflection of the cognitive activity that underlies

user performance. Memory, in the sense of 'information' for subsequent analysis, plays an important role in perceptual systems such as the visual, auditory, haptic and kinesthetic systems (Mania, Troscianko, et al. 2003), (Mania, Wooldridge, et al. 2006). Memory research has established that accurate memory recollections can be linked with the subjective awareness states "Remember", which is a recollection based on a mental image or a prior experience, and "Know", which is a general sense of knowing with no or little recollection of this sense (Mania, Badariah and Coxon 2010). It has been shown that latency does not affect memory recognition and memory awareness states. This could mean that there are strong adaptation processes occurring and related 'perceptual compensation' results in similar spatial awareness while being immersed in an environment of minimum latency as well as in one of somehow higher latency. Therefore, latency could be crucial for tele-operation tasks to be successful; however, generic spatial awareness can be achieved in latency-rich environments. Future work should confirm or contradict such findings also involving immersive environments of varied rendering quality.

References/Bibliography

Ashdown, Ian. "http://tralvex.com/pub/rover/abs-ian0.htm." *Radiosity Bibliography.* March 2004.

Baddeley, A. "Human Memory, Theory and Practice." Psychology Press, 1977.

Björk, Staffan , and Jussi Holopainen. *Patterns in Game Design.* 2004.

Bouknight, J. "A procedure for generation of three dimensional half-toned computer graphics presentations." *Communications of the ACM.* Vol. 13 (9). 1970. 527-536.

Brandt, K. R., J. M. Gardiner, and C. N. MacRae. "The distinctiveness effect in forenames: The role of subjective experiences and recognition memory." *British Journal of Psychology*, 2006: 269-280.

Brewer, W. F., and J. Treyens. "Role of Schemata in Memory for Places." *Cognitive Psychology* 13 (1981): 207-230.

Cakmakci, O., and J. P. Rolland. "Head-Worn Displays: A Review." *IEEE Journal of Display Technology, Vol. 2, No. 3*, 2006.

Cobb, S., S. Nichols, A. Ramsey, and J. Wilson. "Virtual reality incuced symptoms and effects (VRISE)." *Presence: Teleoperators & Virtual Environments*, 1999: 169-186.

Cohen, M. F., and D. P. Greenberg. "The Hemi-Cube: A radiosity solution for complex environments." *Computer Graphics (SIGGRAPH '85 Proceedings).* 1985. 31-40.

Cohen, M. F., S. E. Chen, J. R. Wallace, and D. P. Greenberg. "A progressive refinement approach to fast radiosity image generation." Edited by Computer Graphics. *SIGGRAPH 1988 Proceedings.* 2004. 75-84.

Conway, M. A., J. M. Gardiner, T. J. Perfect, S. J. Anderson, and G. M. Cohen. "Changes in Memory Awareness During Learning: The acquisition of knowledge by psychology undergraduates." *Journal of Experimental Psychology: General* 126, no. 4 (1997): 393 – 413.

Coolican, H. "Research Methods and Statistics in Psychology." Hodder & Stoughton Educational, 1999.

Cunningham, D. W., A. Chatziastros, M. von der Heyde, and H. H. Bulthoff. "Driving in the future: Temporal visuomotor adaptation and generalization." *Journal of Vision* 1 (2001): 88-98.

Cunningham, D. W., V. A. Billock, and B. H. Tsou. "Sensorimotor adaptation to violations in temporal contiguity." *Psychological Science.* Vol. 12. no. 6. 2001. 532-535.

Current-to-voltage converter. n.d. http://en.wikipedia.org/w/index.php?title=Current-to-voltage_converter&oldid=542629706 (accessed 2013).

Davenport, J. L., and M. C. Potter. "Scene Consistency in Object and Background Perception." *Psychological Science* 15, no. 8 (2004): 559-564.

Di Luca, M. "New Method to measure end-to-end delay of virtual reality." *Presence* 19, no. 6 (2010): 569-584.

dictionary, Merriam-Webster online. *Timer.* n.d. http://www.merriam-webster.com/dictionary/timer (accessed 2012).

Dingliana, J. "Image Synthesis Group,." *Trinity College, Dublin.* n.d. http://isg.cs.tcd.ie/dingliaj/3d4ii/Light1.ppt (accessed March 2004).

DiZio, Paul, J. R. Lackner , and L. Matin. "Combined influences of gravitoinertial force level and visual field pitch on visually perceived eye level." *Journal of Vestibular Research*, 1997: 1-12.

Draper, M. H., E. S. Viirre, T. A. Furness, and V. J. Gawon. "Effects of image scale and system time delay on simulator sickness within head-coupled virtual encironments." *Humnan Factors 43(1)*, 20001: 129-146.

Ellis, S. R., A. Wolfram, and B. D. Adelstein. "Large amplitude three-dimensional tracking in augmented environments: a human performance trade-off between system latency and update rate." *46th Annual Meeting Human Factors and Ergonomics Society.* 2002. 2149-2154.

Ellis, S. R., B. D. Adelstein, S. Baumler, G. J. Jense, and R. H. Jacoby. "Sensor spatial distortion, visual latency, and update rate effects on 3D tracking in virtual environments." *VR '99.* 1999. 218-221.

Ellis, S. R., K. Bréant, B. M. Menges, and B. D. Adelstein. "Operator interaction with virtual objects: effects of system latency." *HCI International,.* 1997. 973-976.

Ellis, S. R., K. Mania, B. D. Adelstein, and M Hill. "Generaliza-bility of latency detection in a variety of virtual environments." *48th Annual Meeting Human Factors and Ergonomics Society.* 2004.

Encoders. n.d. http://irtfweb.ifa.hawaii.edu/~tcs3/tcs3/0306_conceptual_design/Docs/05_Encoders/encoder_primer.pdf (accessed 2013).

Fink, Donald G., and H. Wayne, Beaty. *Standard Handbook for Electrical Engineers.* McGraw-Hill, 1999.

Flavios. "Light and optic theory and principles." n.d. http://homepages.tig.com.au/~flavios/diffrac.htm (accessed March 2004).

Foley, J., A. Van Dam, S. Feiner, and J. Hughes. *Computer Graphics: Principles and Practice.* Addison-Wesley Publishing Co., 1990.

Gardiner, J. M., and A. Richardson-Klavenhn. "Remembering and Knowing." In *Handbook of Memory*, by E. Tulving and F. I. M. Craik. Oxford: Oxford University Press, 1992.

Garrett, A., M. Aguilar, and Y. Barniv. "A recurrent neural network approach to virtual environment latency reduction." *IJCNN.* Honolulu, HI, 2002.

Gibson, S., and R.J. Hubbold. "Perceptually Driven Radiosity." *Computer Graphics Forum 16.* 1997. 129-140.

Glassner. "An Introduction to Ray tracing." Morgan kaufmann, 1989.

Glassner, A. S. "Principles of Digital Image Synthesis." San Fransisco, California: Morgan Kaufmann Publishers, 1995.

Glassner, A. S. "Space Subdivision for Fast Ray Tracing, Vol.4, No. 10." *IEEE Computer Graphics & Applications*, 1984: 15-22.

Goral, C. M., K. E. Torrance, D. P. Greenberg, and B. Battaile. "Modeling the interaction of light between diffuse surfaces." *Computer Graphics* 18, no. 3 (July 1984).

Goraud, H. "Continuous shading of curved surfaces." *IEEE Transactions on Computers* 20, no. 6 (1971): 623-628.

Heim, Michael. *The Metaphysics of Virtual Reality.* Oxford University Press, 1993.

Held, R., A. Efstathiou, and M. Greene. "Adaptation to displaced and delayed visual feedback from the hand." *Journal of Experi-mental Psychology* 72, no. 6 (1966): 887-891.

Held, R., and N. Durlach. "Telepresence." *Presence: Teleoperators and Virtual Environments.* 1992. 109–112.

Hill, Michael I., Bernard D. Adelstein, and Stephen R. Ellis. "Achieving Minimum Latency in Virtual Environment Applications." *IMAGE Society Annual Conference.* Scottsdale AZ, 2004.

Holingworth, A., and J. M. Henderson. "Does consistent scene context facilitate object perception?" *Journal of Experimental Psychology: General* 127, no. 4 (1998): 398-415.

Integrated, Maxim. *ADC and DAC Glossary.* n.d. http://www.maximintegrated.com/app-notes/index.mvp/id/641 (accessed April 2013).

Jacoby, R. H., B. D. Adelstein, and S. R. Ellis. "Improved temporal response in virtual environments through system hardware and software reorganization." *SPIE Conference on Stereoscopic Dis-plays and Applications.* 1996. 271-284.

Kajiya, J. T. "The Rendering Equation." *ACM SIGGRAPH.* 1986. 143-150.

Katedros. "Illumination: simulation and perception." n.d. http://www.maf.vu.lt/katedros/cs2/lietuva/courses/spalvos/ illumination5.pdf (accessed February 2004).

Kennedy, R. S., N. E. Lane, M. G. Lilienthal, K. S. Berbaum , and L. J. Hettinger. "Profile analysis of simulator sickness symptoms: application to virtual environment systems." *Presence* 1, no. 3 (1992): 295-301.

Kularatna, Nihal. *Fundamentals of Oscilloscopes", Digital and Analogue Instrumentation: Testing and Measurement, Institution of Engineering and Technology.* Institution of Engineering and Technology, 2003.

Langbein, F. C. "Advanced Rendering - Radiosity." *Cardiff University, (course notes).* n.d. http://cyl.cs.cf.ac.uk/teaching/graphics/G-20-V_2.pdf (accessed February 2004).

Languénou , E., K. Bouatouch , and P. Tellier . "An adaptive Discretization Method for Radiosity." *Computer Graphics Forum 11.* 1992. 205-216.

Levine, B., et al. "Episodic memory and the self in a case of isolated retrograde amnesia." *Brain* 121 (1998): 1951-1973.

Liang, J., C. Shaw, and M. Green. "On temporal-spatial realism in the virtual reality environment." *4th ACM Symposium on User In-terface Software and Technology.* ACM Press, 1991. 19-25.

Lischinski, D. "Radiosity." n.d. http://www.cs.huji.ac.il/~danix/advanced/notes3.pdf (accessed December 2003).

Liu, A, G. Tharp, S. Lai, L. French, and L. W. Stark. "Some of what one needs to know about using head-mounted displays to improve teleoperator performance." *IEEE Transactions on Robotics and Automation* 9, no. 5 (1993): 638-648.

Mania, K., B. D. Adelstein, S. R. Ellis, and M. Hill. "Perceptual sensitivity to head tracking latency in virtual environments with varying degrees of scene complexity." *ACM Siggraph Symposium on Applied Perception in Graphics and Visualization.* ACM Press, 2004. 39-47.

Mania, K., D. Wooldridge, M. Coxon, and A. Robinson. "The effect of visual and interaction fidelity on spatial cognition in immersive virtual environments." *IEEE Transactions on Visualization and Computer Graphics* 12, no. 3 (2006): 396-404.

Mania, K., S. Badariah, and M. Coxon. "Cognitive transfer of training from immersive virtual environments to reality." *ACM Transactions on Applied Perception* 7, no. 2 (2010): 9:1-9:14.

Mania, K., T. Troscianko, R. Hawkes, and A. Chalmers. "Fidelity metrics for virtual environment simulations based on spatial memory awareness states." *Presence, Teleoperators and Virtual Environments* (MIT Press) 12, no. 3 (2003): 296-310.

Mania, Katerina, A. Robinson, and K. Brandt. "The Effect of Memory Schemas on Object Recognition in Virtual Environments." *Presence Teleoperators and Virtual Environments* (MIT Press) 6, no. 1 (2005): 73-86.

McCabe, D. P., and L. D. Geraci. "The influence of instructions and terminology on the accuracy of remember-know judgements." *Consciousness and Cognition* 18 (2009): 401-413.

McNamara, A. "Comparing Real and Synthetic Scenes using Human Judgments of Lightness." *Ph.D thesis.* University of Bristol, 2000.

Meehan, M., S. Razzaque, M. C. Whitton, and F. P. Jr. Brooks. "Effect of latency on presence in stressful virtual environments." *IEEE Virtual Reality Conference.* Los Angeles, 2003. 141-148.

Meijering, Erik. "A chronology of interpolation: from ancient astronomy to modern signal and image processing." *Proceedings of the IEEE 90 (3).* 2002. 319–342,.

Microsoft. *DirectX Software Development Kit.* 2013. http://www.microsoft.com/en-us/download/details.aspx?id=9977 (accessed 2013).

Miné, M. R. "Characterization of end-to-end delays in head-mounted display systems." Chapel Hill, NC, University of North Carolina, 1993.

Minsky, M. "A framework for representing knowledge." *The Psychology of Computer Vision.* Edited by P. H. Winston. McGraw-Hill, 1975.

Mitchell Electronics, Inc. *TI-5000EX Serial/Incremental Encoder Test System User Manual.* n.d. http://www.mitchell-electronics.com/downloads/Catalog_PriceList/TI5000EXManual.pdf (accessed 2011).

Mortensen, J., et al. "Real-Time Global Illumination for VR Applications." *IEEE Computer Graphics and Applications - ISI Impact Factor: 1.87*, 2008: 56-64.

Mourkoussis, N., F. Rivera, T. Troscianko, T. Dixon, R. Hawkes, and Katerina Mania. "Quantifying Fidelity for Virtual Environment simulations employing memory schema assumptions." *ACM Transactions on Applied Perception* (ACM Press) 8, no. 1 (2010): 2:2-2:21.

Nic, M., J. Jirat, and B. Kosata. *Compendium of Chemical Terminology.* IUPAC, 2006.

Nusselt, W. "Grapische Bestimmung des Winkelverhaltnisses bei der Warmestrahlung." *Zeitschrift des Vereines Deutscher Ingenieure.* 1928. 72(20):673.

Ogle, K. N. "Researchers in binocular vision." New York: Hafner Publishing Company, 1950.

Palmer, S. E. " Vision Science – Photons to Phenomenology." Massachusetts Institute of Technology Press, 1999.

Phong, B. T. "Illumination for Computer-Generated Images." *Communications of the ACM* 18, no. 6 (1975): 449-455.

"Physics of light And colour." *Molecular Expressions.* n.d. http://micro.magnet.fsu.edu/primer/java/reflection/specular/ (accessed February 2004).

Pichert, J. W., and R. C. Anderson. "Taking a different perspectives on a story." *Journal of Educational Psychology* 69 (1966): 309-315.

Radoff, Jon. *Anatomy of MMORPG.* 2008. http://radoff.com/blog/2008/08/22/anatomy-of-an-mmorpg.

Repas, Robert. *Multiturn absolute encoders.* n.d. http://machinedesign.com (accessed 2013).

Riesberg, D. "Cognition: Exploring the Science of the Mind." London, UK: Norton and Company, 1997.

Siegel, R., and J. R. Howell. *Thermal Radiation Heat Transfer.* 3rd Edition. New York, NY: Hemisphere Publishing Corporation, 1992.

Spitzer, Frank, and Barry Howarth. *Principles of modern Instrumentation.* New York: Holt, Rinehart and Winston, 1972.

Stanney, K. M., R. R. Mourant, and R. S. Kennedy. "Human factors issues in virtual environments: A review of the literature." *Presence* 7, no. 4 (1998): 327-351.

Steed, A. "A simple method for estimating the latency of interac-tive, real-time graphics simulations." *ACM VRST '08.* 2008.

Steuer, J. "Defining Virtual Reality: Dimensions Determining Telepresence." *Journal of Communication 42*, 1992: 506–508.

Sutherland, I. "The Ultimate Display." *IFIP Congress.* 1965. 506–508.

Tulving, E. *Elements of Episodic Memory.* Oxford: Oxford Science Publications, 1992.

Uno, M., and M. Slater. "The sensitivity of presence to collision response." *IEEE VRAIS.* 1997. 95-101.

Watson, B., N. Walker, P. Woytiuk, and W. Ribarsky. "Maintaining usability during 3D placement despite delay." *IEEE Virtual Reality 2003.* 2003. 133-140.

Welch, R. B., T. T. Blackmon, A. Liu, B. A. Mellers, and L. W. Stark. "The effects of pictorial realism, delay of visual feedback and observer interactivity on the subjective sense of presence." *Presence: Teleoperators and Virtual Environments* 5, no. 3 (1996): 263-273.

Wheeler, M. A., and D. T. Stuss. "Remembering and knowing in patients with frontal lobe injuries." *Cortex* 39 (2003): 827-846.

Wloka, M. "Lag in Multiprocessor Virtual Reality." *Presence Vir-tual Environments and Teleoperators* (MIT Press) 4, no. 1 (1995): 50-63.

WorldTechPub. n.d. http://www.WorldTechPub.com.

Yee, H., S. Pattanaik, and D. P. Greenberg. "Spatiotemporal sensitivity and Visual Attention for efficient rendering of dynamic Environments." *ACM Transactions on Computer Graphics* 20 (2001): 39-65.

Zotos, Alexandros, K. Mania, and N. Mourkoussis. "A Schema-based Selective Rendering Framework." *ACM Siggraph Symposium on Applied Perception in Graphics and Visualization.* Chania, Crete, Greece, 2009. 85-92.

APPENDIX A

Σενάριο πειράματος

Σας ευχαριστούμε που συμμετέχετε στη διαδικασία των πειραμάτων μας. Το πείραμα αποτελείται από **3 στάδια**. Ακολουθούν οι λεπτομέρειες για κάθε στάδιο ξεχωριστά. Αφού διαβάσετε προσεκτικά τις λεπτομέρειες για κάθε στάδιο, είστε έτοιμοι να ξεκινήσετε.

Συνοπτικά, το πρώτο στάδιο αποσκοπεί στην εξοικείωση σας με τον εξοπλισμό και τα τρισδιάστατα γραφικά. Το δεύτερο στάδιο είναι και το κύριο πείραμα. Το τελευταίο στάδιο αποτελείται από μία ενότητα ανασκόπησης.

Στάδιο 1. ***Εξοικείωση με τον εξοπλισμό και τα τρισδιάστατα γραφικά***

Εξάσκηση σε ένα απλό περιβάλλον

Είστε έτοιμοι να συμμετάσχετε στο πρώτο στάδιο. Σκοπός είναι να εξοικειωθείτε με τον εξοπλισμό (Head Mounted Display ή HMD) και τα τρισδιάστατα γραφικά. Έχετε όσο χρόνο επιθυμείτε για να περιηγηθείτε σε έναν ανοιχτό χώρο. Χρησιμοποιήστε τον εξοπλισμό για να στρέψετε την καρέκλα σας 360 μοίρες και να κοιτάξετε προς όλες τις γωνίες. Ρωτήστε ελεύθερα για οποιαδήποτε απορία ή διακόψτε την διαδικασία εάν αισθανθείτε αδιαθεσία.

Εξάσκηση σε ένα πιο ρεαλιστικό περιβάλλον

Αφού αισθανθείτε άνετα με τον εξοπλισμό και την περιήγηση στο πρώτο τρισδιάστατο χώρο ζητήστε από τον υπεύθυνο του πειράματος να "φορτώσει" μια καινούργια πιο ρεαλιστική σκηνή. Σκοπός είναι να εγκλιματιστείτε σε πιο ρεαλιστικές συνθήκες από τις προηγούμενες ώστε να είστε έτοιμοι για το επόμενο στάδιο που είναι και το βασικό. Η καινούρια σκηνή περιγράφει ένα **δωμάτιο** με αντικείμενα και περιοχή κουζίνας, γραφείου και σαλονιού και το απαιτούμενο είναι να παρατηρήσετε όλους τους χώρους και τα αντικείμενα των χώρων αυτών.

Στάδιο 2. ***Το κύριο πείραμα. Περιήγηση σε μια τρισδιάστατη σκηνή.***

Αφού έχετε εξοικειωθεί πλήρως με τον εξοπλισμό, τη περιήγηση σε τρισδιάστατους ρεαλιστικούς χώρους αλλά και εξωτερικά ερεθίσματα όπως ήχοι, καλείστε να περιηγηθείτε σε μια πολυπλοκότερη σκηνή. Λεπτομέρειες για την πλοήγηση και το σενάριο του πειράματος θα δοθούν αμέσως πριν την πραγματοποίηση του.

Στάδιο 3. ***Ανασκόπηση***

Σύμφωνα με την εμπειρία που αποκομίσατε από τη τελευταία σκηνή που είδατε (στάδιο 2), απαντήστε στη φόρμα ανασκόπησης που ακολουθεί. Έχετε όσο χρόνο επιθυμείτε για να διαβάσετε τις οδηγίες και να απαντήσετε σε όλες τις ερωτήσεις.

Παρακαλώ μη συζητήσετε με τους υπόλοιπους συμμετέχοντες οτιδήποτε έχει σχέση με τις εντυπώσεις σας από την πλοήγηση ή το στάδιο ανασκόπησης. Παρόλα αυτά, μπορείτε να ρωτήσετε τους υπεύθυνους του πειράματος για οποιαδήποτε απορία έχετε.

Αυτό είναι και το τελικό στάδιο. Ευχαριστούμε για τη συμμετοχή σας και ελπίζουμε να διασκεδάσατε τη διαδρομή!! ☺

APPENDIX B

ΕΡΩΤΗΜΑΤΟΛΟΓΙΟ ΜΝΗΜΗΣ

Για κάθε αριθμημένη θέση αντικειμένου στη λίστα, ανατρέξτε στην κάτοψη του περιβάλλοντος και διαλέξτε από τη λίστα αντικειμένων ανά περιοχή το αντικείμενο που νομίζετε ότι βρίσκεται σε αυτή τη θέση. Στη συνέχεια, σημειώστε πόσο βέβαιοι είσαστε για κάθε επιλογή σας. Τέλος, επιλέξτε τον τύπο που ακριβέστερα περιγράφει πώς σκεφτήκατε πριν κάνετε την επιλογή σας. Εάν οποιαδήποτε από αυτούς τους όρους είναι παράξενος για σας, ανατρέξτε στον οδηγό επιλογών:

- **ΤΥΠΟΣ Α** σημαίνει ότι μπορείτε να θυμηθείτε με σαφήνεια κάθε αντικείμενο, σε αυτή τη συγκεκριμένη τοποθεσία μέσα στο μυαλό σας. Μπορείτε ουσιαστικά να «δείτε» και πάλι τα αντικείμενα του δωματίου στο μυαλό σας ή θυμάστε συγκεκριμένη άλλη πληροφορία σε σχέση με όταν τα βλέπατε στο χώρο.
- **ΤΥΠΟΣ Β** σημαίνει ότι απλά «ξέρετε» τη σωστή απάντηση που έχετε επιλέξει η οποία απλώς «ξεχώρισε» από τις διαθέσιμες επιλογές. Σε αυτή την περίπτωση δεν μπορείτε να απεικονίσετε την συγκεκριμένη εικόνα ή πληροφορία στο μυαλό σας.

Για κάθε θέση αντικειμένου στη λίστα, ανατρέξτε στην κάτοψη του περιβάλλοντος και διαλέξτε από τη λίστα αντικειμένων το αντικείμενο που νομίζετε ότι βρίσκεται σε αυτή τη θέση. Στη συνέχεια, σημειώστε πόσο βέβαιοι είσαστε για κάθε επιλογή σας. Τέλος, επιλέξτε τον τύπο που ακριβέστερα περιγράφει πώς σκεφτήκατε πριν κάνετε την επιλογή σας. Εάν οποιαδήποτε από αυτούς τους όρους είναι παράξενος να σας, ανατρέξτε στον οδηγό επιλογών παραπάνω:

Αντικείμενο Θέση 1: .....
Βεβαιότητα: ○ Καθόλου βέβαιος/η ○ Χαμηλή βεβαιότητα ○ Μεσαία Βεβαιότητα ○ Βέβαιος/η ○ Απόλυτα Βέβαιος/η
Επίγνωση: ○ ΤΥΠΟΣ Α ○ ΤΥΠΟΣ Β

Αντικείμενο Θέση 2: .....
Βεβαιότητα: ○ Καθόλου βέβαιος/η ○ Χαμηλή βεβαιότητα ○ Μεσαία Βεβαιότητα ○ Βέβαιος/η ○ Απόλυτα Βέβαιος/η
Επίγνωση: ○ ΤΥΠΟΣ Α ○ ΤΥΠΟΣ Β

Αντικείμενο Θέση 3: .....
Βεβαιότητα: ○ Καθόλου βέβαιος/η ○ Χαμηλή βεβαιότητα ○ Μεσαία Βεβαιότητα ○ Βέβαιος/η ○ Απόλυτα Βέβαιος/η
Επίγνωση: ○ ΤΥΠΟΣ Α ○ ΤΥΠΟΣ Β

Αντικείμενο Θέση 4: .....
Βεβαιότητα: ○ Καθόλου βέβαιος/η ○ Χαμηλή βεβαιότητα ○ Μεσαία Βεβαιότητα ○ Βέβαιος/η ○ Απόλυτα Βέβαιος/η
Επίγνωση: ○ ΤΥΠΟΣ Α ○ ΤΥΠΟΣ Β

Αντικείμενο Θέση 5: .....
Βεβαιότητα: ○ Καθόλου βέβαιος/η ○ Χαμηλή βεβαιότητα ○ Μεσαία Βεβαιότητα ○ Βέβαιος/η ○ Απόλυτα Βέβαιος/η
Επίγνωση: ○ ΤΥΠΟΣ Α ○ ΤΥΠΟΣ Β

Αντικείμενο Θέση 6: .....
Βεβαιότητα: ○ Καθόλου βέβαιος/η ○ Χαμηλή βεβαιότητα ○ Μεσαία Βεβαιότητα ○ Βέβαιος/η ○ Απόλυτα Βέβαιος/η
Επίγνωση: ○ ΤΥΠΟΣ Α ○ ΤΥΠΟΣ Β

Αντικείμενο Θέση 7: .....
Βεβαιότητα: ○ Καθόλου βέβαιος/η ○ Χαμηλή βεβαιότητα ○ Μεσαία Βεβαιότητα ○ Βέβαιος/η ○ Απόλυτα Βέβαιος/η
Επίγνωση: ○ ΤΥΠΟΣ Α ○ ΤΥΠΟΣ Β

Αντικείμενο Θέση 8: .....

Βεβαιότητα: ○ Καθόλου βέβαιος/η ○ Χαμηλή βεβαιότητα ○ Μεσαία Βεβαιότητα ○ Βέβαιος/η ○ Απόλυτα Βέβαιος/η

Επίγνωση: ○ ΤΥΠΟΣ Α ○ ΤΥΠΟΣ Β

Αντικείμενο Θέση 9: .....

Βεβαιότητα: ○ Καθόλου βέβαιος/η ○ Χαμηλή βεβαιότητα ○ Μεσαία Βεβαιότητα ○ Βέβαιος/η ○ Απόλυτα Βέβαιος/η

Επίγνωση: ○ ΤΥΠΟΣ Α ○ ΤΥΠΟΣ Β

Αντικείμενο Θέση 10: .....

Βεβαιότητα: ○ Καθόλου βέβαιος/η ○ Χαμηλή βεβαιότητα ○ Μεσαία Βεβαιότητα ○ Βέβαιος/η ○ Απόλυτα Βέβαιος/η

Επίγνωση: ○ ΤΥΠΟΣ Α ○ ΤΥΠΟΣ Β

Αντικείμενο Θέση 11: .....

Βεβαιότητα: ○ Καθόλου βέβαιος/η ○ Χαμηλή βεβαιότητα ○ Μεσαία Βεβαιότητα ○ Βέβαιος/η ○ Απόλυτα Βέβαιος/η

Επίγνωση: ○ ΤΥΠΟΣ Α ○ ΤΥΠΟΣ Β

Αντικείμενο Θέση 12: .....

Βεβαιότητα: ○ Καθόλου βέβαιος/η ○ Χαμηλή βεβαιότητα ○ Μεσαία Βεβαιότητα ○ Βέβαιος/η ○ Απόλυτα Βέβαιος/η

Επίγνωση: ○ ΤΥΠΟΣ Α ○ ΤΥΠΟΣ Β

Αντικείμενο Θέση 13: .....

Βεβαιότητα: ○ Καθόλου βέβαιος/η ○ Χαμηλή βεβαιότητα ○ Μεσαία Βεβαιότητα ○ Βέβαιος/η ○ Απόλυτα Βέβαιος/η

Επίγνωση: ○ ΤΥΠΟΣ Α ○ ΤΥΠΟΣ Β

Αντικείμενο Θέση 14: .....

Βεβαιότητα: ○ Καθόλου βέβαιος/η ○ Χαμηλή βεβαιότητα ○ Μεσαία Βεβαιότητα ○ Βέβαιος/η ○ Απόλυτα Βέβαιος/η

Επίγνωση: ○ ΤΥΠΟΣ Α ○ ΤΥΠΟΣ Β

Αντικείμενο Θέση 15: .....

Βεβαιότητα: ○ Καθόλου βέβαιος/η ○ Χαμηλή βεβαιότητα ○ Μεσαία Βεβαιότητα ○ Βέβαιος/η ○ Απόλυτα Βέβαιος/η

Επίγνωση: ○ ΤΥΠΟΣ Α ○ ΤΥΠΟΣ Β

Αντικείμενο Θέση 16: .....

Βεβαιότητα: ○ Καθόλου βέβαιος/η ○ Χαμηλή βεβαιότητα ○ Μεσαία Βεβαιότητα ○ Βέβαιος/η ○ Απόλυτα Βέβαιος/η

Επίγνωση: ○ ΤΥΠΟΣ Α ○ ΤΥΠΟΣ Β

Αντικείμενο Θέση 17: .....

Βεβαιότητα: ○ Καθόλου βέβαιος/η ○ Χαμηλή βεβαιότητα ○ Μεσαία Βεβαιότητα ○ Βέβαιος/η ○ Απόλυτα Βέβαιος/η

Επίγνωση: ○ ΤΥΠΟΣ Α ○ ΤΥΠΟΣ Β

Αντικείμενο Θέση 18: .....

Βεβαιότητα: ○ Καθόλου βέβαιος/η ○ Χαμηλή βεβαιότητα ○ Μεσαία Βεβαιότητα ○ Βέβαιος/η ○ Απόλυτα Βέβαιος/η

Επίγνωση: ○ ΤΥΠΟΣ Α ○ ΤΥΠΟΣ Β

Αντικείμενο Θέση 19: .....

Βεβαιότητα: ○ Καθόλου βέβαιος/η ○ Χαμηλή βεβαιότητα ○ Μεσαία Βεβαιότητα ○ Βέβαιος/η ○ Απόλυτα Βέβαιος/η

Επίγνωση: ○ ΤΥΠΟΣ Α ○ ΤΥΠΟΣ Β

Αντικείμενο Θέση 20: .....

Βεβαιότητα: ○ Καθόλου βέβαιος/η ○ Χαμηλή βεβαιότητα ○ Μεσαία Βεβαιότητα ○ Βέβαιος/η ○ Απόλυτα Βέβαιος/η

Επίγνωση: ○ ΤΥΠΟΣ Α ○ ΤΥΠΟΣ Β

Αντικείμενο Θέση 21: .....

Βεβαιότητα: ◯ Καθόλου βέβαιος/η ◯ Χαμηλή βεβαιότητα ◯ Μεσαία Βεβαιότητα ◯ Βέβαιος/η ◯ Απόλυτα Βέβαιος/η

Επίγνωση: ◯ ΤΥΠΟΣ Α ◯ ΤΥΠΟΣ Β

Αντικείμενο Θέση 22: .....

Βεβαιότητα: ◯ Καθόλου βέβαιος/η ◯ Χαμηλή βεβαιότητα ◯ Μεσαία Βεβαιότητα ◯ Βέβαιος/η ◯ Απόλυτα Βέβαιος/η

Επίγνωση: ◯ ΤΥΠΟΣ Α ◯ ΤΥΠΟΣ Β

Αντικείμενο Θέση 23: .....

Βεβαιότητα: ◯ Καθόλου βέβαιος/η ◯ Χαμηλή βεβαιότητα ◯ Μεσαία Βεβαιότητα ◯ Βέβαιος/η ◯ Απόλυτα Βέβαιος/η

Επίγνωση: ◯ ΤΥΠΟΣ Α ◯ ΤΥΠΟΣ Β

Αντικείμενο Θέση 24: .....

Βεβαιότητα: ◯ Καθόλου βέβαιος/η ◯ Χαμηλή βεβαιότητα ◯ Μεσαία Βεβαιότητα ◯ Βέβαιος/η ◯ Απόλυτα Βέβαιος/η

Επίγνωση: ◯ ΤΥΠΟΣ Α ◯ ΤΥΠΟΣ Β

APPENDIX C

Σαλόνι

- Αναπτήρας
- Βάζο με λουλούδια
- Βαρέλι
- Βιβλίο
- Καπέλο cowboy
- Κηροπήγιο
- Κολοκύθα Halloween
- Κράνος πολεμιστή
- Μάσκα αερίων
- Μπάλα μπάσκετ
- Μπουκέτο Λουλούδια
- Περιοδικό
- Ρακέτα
- Σφυρί
- Τασάκι
- Τηλεκοντρόλ

Κουζίνα

- Γάλα
- Κατσαρόλα
- Κομπιουτεράκι τσέπης
- Μίξερ
- Μπαστούνι baseball
- Μπρίκι
- Ρόδα ποδηλάτου
- Ρόπαλο με καρφιά
- Σουρωτήρι
- Σύριγγα
- Φλιτζάνι καφέ
- Φρούτα
- Φρυγανιέρα
- Φτυάρι
- Φωτογραφική μηχανή
- Χαρταετός

Γραφείο

- Βιβλία
- Γλάστρα
- Εκτυπωτής
- Ηλεκτρική κιθάρα
- Κούπα καφέ
- Μολυβοθήκη
- Μπαλάκι τένις
- Πιστωτική κάρτα
- Σελοτέιπ
- Σπαθί
- Στυλό
- Συρραπτικό
- Ταμιακή μηχανή
- Τηγάνι
- Τρομπέτα
- Φορητός υπολογιστής

Standard Industrial types AC 58

Absolute Parallel

Synchro flange

- Compact design
- Aids for start up and operation: diagnostic LED, preset key with optical response (only with MT), status information
- Output Tristate short circuit-proof
- Gray or Binary code
- Encoder monitoring

TECHNICAL DATA
mechanical

Housing diameter	58 mm
Shaft diameter	6 mm / 10 mm (Solid shaft) 10 mm / 12 mm (Hub shaft)
Flange (Mounting of housing)	Synchro flange, Clamping flange, Tether, Square flange
Protection class shaft input (EN 60529)	IP64 or IP67
Protection class housing (EN 60529)	IP64 or IP67
Shaft load axial / radial	40 N / 60 N
Axial endplay of mounting shaft (hubshaft)	± 1.5 mm
Radial runout of mating shaft (hubshaft)	± 0.2 mm
Max. speed	max. 10 000 rpm (continuous), max. 12 000 rpm (short term)
Starting torque typ. [3]	≤ 0.01 Nm
Moment of inertia	ca. 3.8 x 10^{-6} kgm²
Vibration resistance (DIN EN 60068-2-6)	100 m/s² (10 ... 2000 Hz)
Shock resistance (DIN EN 60068-2-27)	1000 m/s² (6 ms)
Operating temperature	-40 °C ... +100 °C
Storage temperature	-40 °C ... +85 °C
Weight	approx. 350 g (ST) / 400 g (MT)
Connection [2]	Cable, axial or radial M23 connector (Conin), 17 pole, axial or radial Sub-D connector, 37 pole

TECHNICAL DATA
electrical

Supply voltage	DC 10-30 V On request: DC 5 V
Current w/o load typ.	5 V: 150 mA (ST), 300 mA (MT) 10 - 30 V: 200 mA (ST), 300 mA (MT)
Allowable load	max. 30 mA
Resolution singleturn	10 - 14 Bit Gray Excess: 360, 720 increments
Resolution multiturn	12 Bit
Output code	Binary, Gray, Gray Excess
Linearity	± ½ LSB
Output current	30 mA per Bit, short-circuit-proof

Absolute Parallel

TECHNICAL DATA
electrical (continued)

Control inputs	Latch, Direction, Tristate with ST, Tristate with MT
Alarm output	NPN-O.C., max. 5 mA
Status LED	Green = ok, red = alarm

Data output level

Supply voltage U_B	DC 5 V - 5 % +10 % [1]	DC 10 - 30 V
Output level High	≥ 3.5 V (30 mA) ≥ 3.9 V (10 mA)	≥ U_B -2.2 V (30 mA) ≥ U_B -1.8 V (10 mA)
Output level Low	≤ 1.6 V (30 mA) ≤ 1.2 V (10 mA)	≤ 1.6 V (30 mA) ≤ 1.2 V (10 mA)
Rise time (1.5 m Cable)	≤ 0.1 µs	≤ 0.2 µs
Drop time (1.5 m Cable)	≤ 0.05 µs	≤ 0.1 µs

[1] on request

Control inputs

Input	Level logical (physical)	Function
Direction	1 (+ U_B or open) 0 (0 V)	ascending code values when turning clockwise (cw) descending code values when turning clockwise (cw)
Latch	1 (+ U_B or open) 0 (0 V)	encoder data continuously changing at output encoder data stored and constant at output
Tristate (with singleturn)	1 (+ U_B or open) 0 (0 V)	outputs active outputs at high impedance (Tristate mode)
Tristate (with multiturn)	1 (+ U_B) 0 (0 V or open)	outputs at high impedance (Tristate mode) outputs active

Typical actuating delay time 10 µs with push-pull selection; when selected via O.C., an external pull-down resistor (1 KΩ) is required

ELECTRICAL CONNECTIONS
Singleturn, cable

Colour (PVC)	9 Bit / 360 incr.	10 Bit / 720 incr.	12 Bit	13 Bit	14 Bit
grey/pink	N.C.	N.C.	N.C.	N.C.	S0 (LSB)
brown/yellow	N.C.	N.C.	N.C.	S0 (LSB)	S1
brown/grey	N.C.	N.C.	S0 (LSB)	S1	S2
red/blue	N.C.	N.C.	S1	S2	S3
violet	N.C.	S0 (LSB)	S2	S3	S4
white/brown	S0 (LSB)	S1	S3	S4	S5
white/green	S1	S2	S4	S5	S6
white/yellow	S2	S3	S5	S6	S7
white/grey	S3	S4	S6	S7	S8
white/pink	S4	S5	S7	S8	S9
white/blue	S5	S6	S8	S9	S10
white/red	S6	S7	S9	S10	S11
white/black	S7	S8	S10	S11	S12
brown/green	S8 (MSB)	S9 (MSB)	S11 (MSB)	S12 (MSB)	S13 (MSB)
yellow	Tristate S0...S8	Tristate S0...S9	Tristate S0...S11	Tristate S0...S12	Tristate S0...S13
pink	Latch	Latch	Latch	Latch	Latch
green	Direction	Direction	Direction	Direction	Direction
black	0 V	0 V	0 V	0 V	0 V
red	DC 5 V/ 10-30 V	DC 5 V/ 10-30 V	DC 5 V/ 10-30 V	DC 5 V/ 10-30 V	DC 5 V/ 10-30 V
brown	Alarm	Alarm	Alarm	Alarm	Alarm

ELECTRICAL CONNECTIONS
Singleturn, M23 connector (Conin), 17 pole

Pin	9 Bit / 360 incr.	10 Bit / 720 incr.	12 Bit	13 Bit	14 Bit
1	S0 (LSB)	S0 (LSB)	S0 (LSB)	S12 (MSB)	S13 (MSB)
2	S1	S1	S1	S11	S12
3	S2	S2	S2	S10	S11
4	S3	S3	S3	S9	S10
5	S4	S4	S4	S8	S9
6	S5	S5	S5	S7	S8
7	S6	S6	S6	S6	S7
8	S7	S7	S7	S5	S6
9	S8 (MSB)	S8	S8	S4	S5
10	N.C.	S9 (MSB)	S9	S3	S4
11	N.C.	N.C.	S10	S2	S3
12	$\overline{\text{Tristate}}$ S0...S8	$\overline{\text{Tristate}}$ S0...S9	S11 (MSB)	S1	S2
13	$\overline{\text{Latch}}$	$\overline{\text{Latch}}$	$\overline{\text{Latch}}$	S0 (LSB)	S1
14	$\overline{\text{Direction}}$	$\overline{\text{Direction}}$	$\overline{\text{Direction}}$	$\overline{\text{Direction}}$	S0 (LSB)
15	0 V	0 V	0 V	0 V	0 V
16	DC 5 V/ 10-30 V	DC 5 V/ 10-30 V	DC 5 V/ 10-30 V	DC 5 V/ 10-30 V	DC 5 V/ 10-30 V
17	$\overline{\text{Alarm}}$	$\overline{\text{Alarm}}$	$\overline{\text{Alarm}}$	$\overline{\text{Latch}}$/$\overline{\text{Alarm}}$	$\overline{\text{Latch}}$/$\overline{\text{Alarm}}$

ELECTRICAL CONNECTIONS
Multiturn, cable

Cable (TPE)	10 cm cable with Sub-D connector, 37 pole		Cable (TPE)	10 cm cable with Sub-D connector, 37 pole	
Colour	Pin	Connection	Colour	Pin	Connection
brown	2	S0	white/blue	14	M4 [1]
green	21	S1	brown/blue	33	M5 [1]
yellow	3	S2	white/red	15	M6 [1]
grey	22	S3	brown/red	34	M7 [1]
pink	4	S4	white/black	16	M8 [2]
violet	23	S5	brown/black	35	M9 [2]
grey/pink	5	S6	grey/green	17	M10 [2]
red/blue	24	S7	yellow/grey	36	M11 [2]
white/green	6	S8	pink/green	18	$\overline{\text{Alarm}}$
brown/green	25	S9	yellow/pink	10	$\overline{\text{Direction}}$
white/yellow	7	S10	green/blue	30	$\overline{\text{Latch}}$
yellow/brown	26	S11	yellow/blue	12	Tristate
white/grey	8	M0	red (0.5mm²)	13	DC 10-30 V
grey/brown	27	M1	white (0.5mm²)	31	DC 10-30 V
white/pink	9	M2	blue (0.5mm²)	1	0 V
pink/brown	28	M3	black (0.5mm²)	20	0 V

[1] N. C. with resolution 16 Bit (4 Bit MT)
[2] N. C. with resolution 16 Bit or 20 Bit (4 or 8 Bit MT)

DIMENSIONED DRAWINGS

see chapter "Dimensioned drawings AC 58, starting page 178

Absolute Parallel

ORDERING INFORMATION

Type	Resolution [1, 2]	Supply voltage	Flange, Protection, Shaft [3 7]	Interface	Connection [4, 5, 6]
AC58	**0010** 10 Bit ST **0012** 12 Bit ST **0013** 13 Bit ST **0014** 14 Bit ST **0360** 360 increments ST **0720** 720 increments ST **0412** 4 Bit MT + 12 Bit ST **0812** 8 Bit MT + 12 Bit ST **1212** 12 Bit MT + 12 Bit ST	**E** DC 10 - 30 V	**S.41 Synchro, IP64, 6 mm** **S.71 Synchro, IP67, 6 mm** **K.42 Clamping, IP64, 10 mm** **K.46** Clamping, IP64, 9.52 mm **K.72 Clamping, IP67, 10 mm** **K.76** Clamping, IP67, 9.52 mm **F.46** Spring tether, IP64, hubshaft 9.52 mm, mounting with clamping ring front **F.42 Spring tether, IP64, hubshaft 10 mm, mounting with clamping ring front** **F.47 Spring tether, IP64, hubshaft 12 mm, mounting with clamping ring front** **Q.46** Square, IP64, 9.52 mm **Q.42** Square, IP64, 10 mm **Q.76** Square, IP67, 9.52 mm **Q.72** Square, IP67, 10 mm	**PB** Parallel binary **PG** Parallel Gray	**A** Cable, axial **B** Cable, radial **U** M23 connector (Conin), 17 pole, axial, ccw **V** M23 connector (Conin), 17 pole, radial, ccw **W** M23 connector (Conin), 17 pole, axial, cw **Y** M23 connector (Conin), 17 pole, radial, cw **A-A1-F** 0,1 m cable with Sub-D connector, 37 pole, axial **B-A1-F** 0,1 m cable with Sub-D connector, 37 pole, radial

[1] Resolution 360 increments ST with Offset 76 (value range 76...435)
[2] Resolution 720 increments ST with Offset 152 (value range 152...871)
[3] Protection class IP67 not available in combination with preset key and LED display
[4] Connection code "A", "B" (cable): ST and MT
[5] Connection code "U", "V", "W", "Y" (M23 connector): only ST
[6] Connection code "A-A1-F" and "B-A1-F" (Sub-D connector): only MT
[7] IP67 on cover with connector only if IP67 mating connector mounted properly.

Preferably available versions are printed in bold type.

ORDERING INFORMATION

Selection of cable length

Versions with cable outlet (connection A, B, E or F) are available with various lengths of cable. To order your desired cable length, please add the respective code to the end of your ordering code. For variants with connector on cable end please add cable length code in between. Further cable lengths on request.

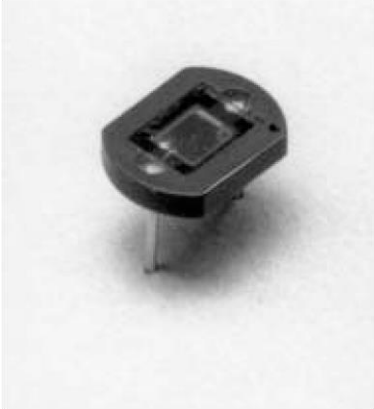
Code	Cable length
without code	1.5 m
-D0	3 m
-F0	5 m
-K0	10 m
-P0	15 m
-U0	20 m
-V0	25 m

Example:
Cable 3 m length: ... B - D0
Cable mit 3 m length and M23 connectorr, cw: ... B - D0 - I

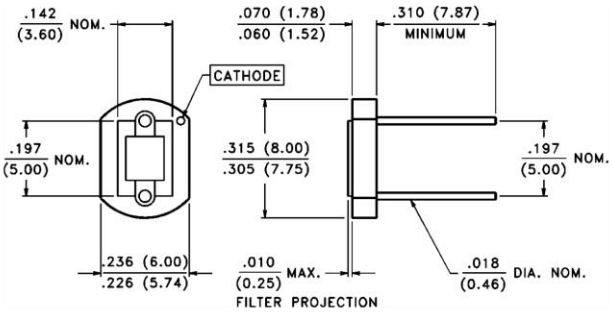
ACCESSORIES

see chapter "Accessories"

VTB Process Photodiode — VTB8440BH, 8441BH



PACKAGE DIMENSIONS inch (mm)



CASE 21F 8 mm CERAMIC
CHIP ACTIVE AREA: .008 in^2 (5.16 mm^2)

PRODUCT DESCRIPTION

Planar silicon photodiode in recessed ceramic package. The package incorporates an infrared rejection filter. These diodes have very high shunt resistance and have good blue response.

ABSOLUTE MAXIMUM RATINGS

Storage Temperature:	-20°C to 75°C
Operating Temperature:	-20°C to 75°C

RoHS Compliant

ELECTRO-OPTICAL CHARACTERISTICS @ 25°C (See also VTB curves, pages 21-22)

SYMBOL	CHARACTERISTIC	TEST CONDITIONS	VTB8440BH			VTB8441BH			UNITS
			Min.	Typ.	Max.	Min.	Typ.	Max.	
I_{SC}	Short Circuit Current	H = 100 fc, 2850 K	4	5		4	5		µA
TC I_{SC}	I_{SC} Temperature Coefficient	2850 K		.02	.08		.02	.08	%/°C
V_{OC}	Open Circuit Voltage	H = 100 fc, 2850 K		420			420		mV
TC V_{OC}	V_{OC} Temperature Coefficient	2850 K		-2.0			-2.0		mV/°C
I_D	Dark Current	H = 0, VR = 2.0 V			2000			100	pA
R_{SH}	Shunt Resistance	H = 0, V = 10 mV		.07			1.4		GΩ
TC R_{SH}	R_{SH} Temperature Coefficient	H = 0, V = 10 mV		-8.0			-8.0		%/°C
C_J	Junction Capacitance	H = 0, V = 0		1.0			1.0		nF
λ_{range}	Spectral Application Range		330		720	330		720	nm
λ_p	Spectral Response - Peak			580			580		nm
V_{BR}	Breakdown Voltage		2	40		2	40		V
$\theta_{1/2}$	Angular Resp. - 50% Resp. Pt.			±50			±50		Degrees
NEP	Noise Equivalent Power		1.1 x 10^{-13} (Typ.)			2.4 x 10^{-14} (Typ.)			$W/\sqrt{Hz}$
D*	Specific Detectivity		2.2 x 10^{12} (Typ.)			9.7 x 10^{12} (Typ.)			$cm\sqrt{Hz}/W$

PerkinElmer Optoelectronics, 22001 Dumberry, Vaudreuil, Canada J7V 8P7 Phone: 877-734-6786 Fax: 450-424-3413 www.perkinelmer.com/opto

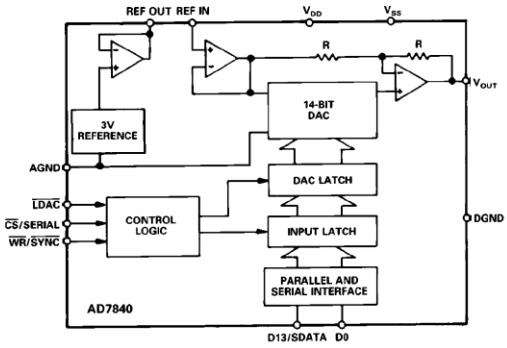
ANALOG DEVICES

LC²MOS Complete 14-Bit DAC

AD7840

FEATURES
Complete 14-Bit Voltage Output DAC
Parallel and Serial Interface Capability
80 dB Signal-to-Noise Ratio
Interfaces to High Speed DSP Processors
e.g., ADSP-2100, TMS32010, TMS32020
45 ns min $\overline{WR}$ Pulse Width
Low Power – 70 mW typ.
Operates from ±5 V Supplies

FUNCTIONAL BLOCK DIAGRAM



GENERAL DESCRIPTION

The AD7840 is a fast, complete 14-bit voltage output D/A converter. It consists of a 14-bit DAC, 3 V buried Zener reference, DAC output amplifier and high speed control logic.

The part features double-buffered interface logic with a 14-bit input latch and 14-bit DAC latch. Data is loaded to the input latch in either of two modes, parallel or serial. This data is then transferred to the DAC latch under control of an asynchronous $\overline{LDAC}$ signal. A fast data setup time of 21 ns allows direct parallel interfacing to digital signal processors and high speed 16-bit microprocessors. In the serial mode, the maximum serial data clock rate can be as high as 6 MHz.

The analog output from the AD7840 provides a bipolar output range of ±3 V. The AD7840 is fully specified for dynamic performance parameters such as signal-to-noise ratio and harmonic distortion as well as for traditional dc specifications. Full power output signals up to 20 kHz can be created.

The AD7840 is fabricated in linear compatible CMOS (LC^2MOS), an advanced, mixed technology process that combines precision bipolar circuits with low power CMOS logic. The part is available in a 24-pin plastic and hermetic dual-in-line package (DIP) and is also packaged in a 28-terminal plastic leaded chip carrier (PLCC).

PRODUCT HIGHLIGHTS

1. Complete 14-Bit D/A Function
   The AD7840 provides the complete function for creating ac signals and dc voltages to 14-bit accuracy. The part features an on-chip reference, an output buffer amplifier and 14-bit D/A converter.
2. Dynamic Specifications for DSP Users
   In addition to traditional dc specifications, the AD7840 is specified for ac parameters including signal-to-noise ratio and harmonic distortion. These parameters along with important timing parameters are tested on every device.
3. Fast, Versatile Microprocessor Interface
   The AD7840 is capable of 14-bit parallel and serial interfacing. In the parallel mode, data setup times of 21 ns and write pulse widths of 45 ns make the AD7840 compatible with modern 16-bit microprocessors and digital signal processors. In the serial mode, the part features a high data transfer rate of 6 MHz.

REV. B

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 617/329-4700 **Fax: 617/326-8703**

AD7840–SPECIFICATIONS (V_{DD} = +5 V ± 5%, V_{SS} = –5 V ± 5%, AGND = DGND = 0 V, REF IN = +3 V, R_L = 2 kΩ, C_L = 100 pF. All specifications T_{MIN} to T_{MAX} unless otherwise noted.)

Parameter	J, A[1]	K, B[1]	S[1]	Units	Test Conditions/Comments
DYNAMIC PERFORMANCE[2]					
Signal to Noise Ratio[3] (SNR)	76	78	76	dB min	V_{OUT} = 1 kHz Sine Wave, f_{SAMPLE} = 100 kHz
					Typically 82 dB at +25°C for 0 < V_{OUT} < 20 kHz[4]
Total Harmonic Distortion (THD)	–78	–80	–78	dB max	V_{OUT} = 1 kHz Sine Wave, f_{SAMPLE} = 100 kHz
					Typically –84 dB at +25°C for 0 < V_{OUT} < 20 kHz[4]
Peak Harmonic or Spurious Noise	–78	–80	–78	dB max	V_{OUT} = 1 kHz Sine Wave, f_{SAMPLE} = 100 kHz
					Typically –84 dB at +25°C for 0 < V_{OUT} < 20 kHz[4]
DC ACCURACY					
Resolution	14	14	14	Bits	
Integral Nonlinearity	±2	±1	±2	LSB max	
Differential Nonlinearity	±0.9	±0.9	±0.9	LSB max	Guaranteed Monotonic
Bipolar Zero Error	±10	±10	±10	LSB max	
Positive Full Scale Error[5]	±10	±10	±10	LSB max	
Negative Full Scale Error[5]	±10	±10	±10	LSB max	
REFERENCE OUTPUT[6]					
REF OUT @ +25°C	2.99	2.99	2.99	V min	
	3.01	3.01	3.01	V max	
REF OUT TC	±60	±60	±60	ppm/°C max	
Reference Load Change					
(ΔREF OUT vs. ΔI)	–1	–1	–1	mV max	Reference Load Current Change (0–500 μA)
REFERENCE INPUT					
Reference Input Range	2.85	2.85	2.85	V min	3 V ± 5%
	3.15	3.15	3.15	V max	
Input Current	50	50	50	μA max	
LOGIC INPUTS					
Input High Voltage, V_{INH}	2.4	2.4	2.4	V min	V_{DD} = 5 V ± 5%
Input Low Voltage, V_{INL}	0.8	0.8	0.8	V max	V_{DD} = 5 V ± 5%
Input Current, I_{IN}	±10	±10	±10	μA max	V_{IN} = 0 V to V_{DD}
Input Current ($\overline{CS}$ Input Only)	±10	±10	±10	μA max	V_{IN} = V_{SS} to V_{DD}
Input Capacitance, C_{IN}[7]	10	10	10	pF max	
ANALOG OUTPUT					
Output Voltage Range	±3	±3	±3	V nom	
DC Output Impedance	0.1	0.1	0.1	Ω typ	
Short-Circuit Current	20	20	20	mA typ	
AC CHARACTERISTICS[7]					
Voltage Output Settling Time					Settling Time to within ±1/2 LSB of Final Value
Positive Full-Scale Change	4	4	4	μs max	Typically 2 μs
Negative Full-Scale Change	4	4	4	μs max	Typically 2.5 μs
Digital-to-Analog Glitch Impulse	10	10	10	nV secs typ	
Digital Feedthrough	2	2	2	nV secs typ	
POWER REQUIREMENTS					
V_{DD}	+5	+5	+5	V nom	±5% for Specified Performance
V_{SS}	–5	–5	–5	V nom	±5% for Specified Performance
I_{DD}	14	14	15	mA max	Output Unloaded, SCLK = +5 V. Typically 10 mA
I_{SS}	6	6	7	mA max	Output Unloaded, SCLK = +5 V. Typically 4 mA
Power Dissipation	100	100	110	mW max	Typically 70 mW

NOTES
[1]Temperature ranges are as follows: J, K Versions, 0°C to +70°C; A, B Versions, –25°C to +85°C; S Version, –55°C to +125°C.
[2]V_{OUT} (pk-pk) = ±3 V
[3]SNR calculation includes distortion and noise components.
[4]Using external sample-and-hold (see Testing the AD7840).
[5]Measured with respect to REF IN and includes bipolar offset error.
[6]For capacitive loads greater than 50 pF, a series resistor is required (see Internal Reference section).
[7]Sample tested @ +25°C to ensure compliance.

Specifications subject to change without notice.

TIMING CHARACTERISTICS[1, 2] (V_{DD} = +5 V ± 5%, V_{SS} = –5 V ± 5%, AGND = DGND = 0 V.)

Parameter	Limit at T_{MIN}, T_{MAX} (J, K, A, B Versions)	Limit at T_{MIN}, T_{MAX} (S Version)	Units	Conditions/Comments
t_1	0	0	ns min	$\overline{CS}$ to $\overline{WR}$ Setup Time
t_2	0	0	ns min	$\overline{CS}$ to $\overline{WR}$ Hold Time
t_3	**45**	**50**	ns min	$\overline{WR}$ Pulse Width
t_4	**21**	**28**	ns min	Data Valid to $\overline{WR}$ Setup Time
t_5	**10**	**15**	ns min	Data Valid to $\overline{WR}$ Hold Time
t_6	40	40	ns min	$\overline{LDAC}$ Pulse Width
t_7	50	50	ns min	$\overline{SYNC}$ to SCLK Falling Edge
t_8[3]	150	200	ns min	SCLK Cycle Time
t_9	30	40	ns min	Data Valid to SCLK Setup Time
t_{10}	75	100	ns min	Data Valid to SCLK Hold Time
t_{11}	75	100	ns min	$\overline{SYNC}$ to SCLK Hold Time

NOTES
[1]Timing specifications in **bold print** are 100% production tested. All other times are sample tested at +25°C to ensure compliance. All input signals are specified with tr = tf = 5 ns (10% to 90% of 5 V) and timed from a voltage level of 1.6 V.
[2]See Figures 6 and 8.
[3]SCLK mark/space ratio is 40/60 to 60/40.

Specifications subject to change without notice.

ABSOLUTE MAXIMUM RATINGS*

V_{DD} to AGND . . . . . –0.3 V to +7 V
V_{SS} to AGND . . . . . +0.3 V to –7 V
AGND to DGND . . . . . –0.3 V to V_{DD} + 0.3 V
V_{OUT} to AGND . . . . . V_{SS} to V_{DD}
REF OUT to AGND . . . . . 0 V to V_{DD}
REF IN to AGND . . . . . –0.3 V to V_{DD} + 0.3 V
Digital Inputs to DGND . . . . . –0.3 V to V_{DD} + 0.3 V
Operating Temperature Range
  Commercial (J, K Versions) . . . . . 0°C to +70°C
  Industrial (A, B Versions) . . . . . –25°C to +85°C
  Extended (S Version) . . . . . –55°C to +125°C
Storage Temperature Range . . . . . –65°C to +150°C
Lead Temperature (Soldering, 10 sec) . . . . . +300°C
Power Dissipation (Any Package) to +75°C . . . . . 450 mW
Derates above +75°C by . . . . . 10 mW/°C

*Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those listed in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ORDERING GUIDE

Model[1]	Temperature Range	SNR (dB)	Integral Nonlinearity (LSB)	Package Option[2]
AD7840JN	0°C to +70°C	78 min	±2 max	N-24
AD7840KN	0°C to +70°C	80 min	±1 max	N-24
AD7840JP	0°C to +70°C	78 min	±2 max	P-28A
AD7840KP	0°C to +70°C	80 min	±1 max	P-28A
AD7840AQ	–25°C to +85°C	78 min	±2 max	Q-24
AD7840ARS	–25°C to +85°C	78 min	±2 max	RS-24
AD7840BQ	–25°C to +85°C	80 min	±1 max	Q-24
AD7840SQ[3]	–55°C to +125°C	78 min	±2 max	Q-24

NOTES
[1]To order MIL-STD-883, Class B processed parts, add /883B to part number. Contact your local sales office for military data sheet and availability.
[2]N = Plastic DIP; P = Plastic Leaded Chip Carrier; Q = Cerdip.
[3]This grade will be available to /883B processing only.

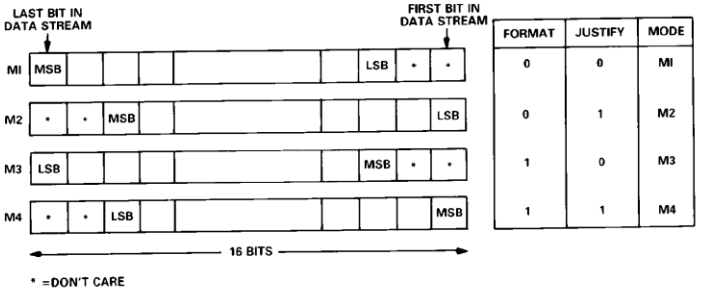
CAUTION

ESD (electrostatic discharge) sensitive device. Electrostatic charges as high as 4000 V readily accumulate on the human body and test equipment and can discharge without detection. Although the AD7840 features proprietary ESD protection circuitry, permanent damage may occur on devices subjected to high energy electrostatic discharges. Therefore, proper ESD precautions are recommended to avoid performance degradation or loss of functionality.

PIN FUNCTION DESCRIPTION

DIP Pin No.	Pin Mnemonic	Function
1	$\overline{CS}$/SERIAL	Chip Select/Serial Input. When driven with normal logic levels, it is an active low logic input which is used in conjunction with $\overline{WR}$ to load parallel data to the input latch. For applications where $\overline{CS}$ is permanently low, an R, C is required for correct power-up (see $\overline{LDAC}$ input). If this input is tied to V_{SS}, it defines the AD7840 for serial mode operation.
2	$\overline{WR}$/$\overline{SYNC}$	Write/Frame Synchronization Input. In the parallel data mode, it is used in conjunction with $\overline{CS}$ to load parallel data. In the serial mode of operation, this pin functions as a Frame Synchronization pulse with serial data expected after the falling edge of this signal.
3	D13/SDATA	Data Bit 13(MSB)/Serial Data. When parallel data is selected, this pin is the D13 input. In serial mode, SDATA is the serial data input which is used in conjunction with $\overline{SYNC}$ and SCLK to transfer serial data to the AD7840 input latch.
4	D12/SCLK	Data Bit 12/Serial Clock. When parallel data is selected, this pin is the D12 input. In the serial mode, it is the serial clock input. Serial data bits are latched on the falling edge of SCLK when $\overline{SYNC}$ is low.
5	D11/FORMAT	Data Bit 11/Data Format. When parallel data is selected, this pin is the D11 input. In serial mode, a Logic 1 on this input indicates that the MSB is the first valid bit in the serial data stream. A Logic 0 indicates that the LSB is the first valid bit (see Table I).
6	D10/JUSTIFY	Data Bit 10/Data Justification. When parallel data is selected, this pin is the D10 input. In serial mode, this input controls the serial data justification (see Table I).
7–11	D9–D5	Data Bit 9 to Data Bit 5. Parallel data inputs.
12	DGND	Digital Ground. Ground reference for digital circuitry.
13–16	D4–D1	Data Bit 4 to Data Bit 1. Parallel data inputs.
17	D0	Data Bit 0 (LSB). Parallel data input.
18	V_{DD}	Positive Supply, +5 V ± 5%.
19	AGND	Analog Ground. Ground reference for DAC, reference and output buffer amplifier.
20	V_{OUT}	Analog Output Voltage. This is the buffer amplifier output voltage. Bipolar output range (±3 V with REF IN = +3 V).
21	V_{SS}	Negative Supply Voltage, –5 V ± 5%.
22	REF OUT	Voltage Reference Output. The internal 3 V analog reference is provided at this pin. To operate the AD7840 with internal reference, REF OUT should be connected to REF IN. The external load capability of the reference is 500 μA.
23	REF IN	Voltage Reference Input. The reference voltage for the DAC is applied to this pin. It is internally buffered before being applied to the DAC. The nominal reference voltage for correct operation of the AD7840 is 3 V.
24	$\overline{LDAC}$	Load DAC. Logic Input. A new word is loaded into the DAC latch from the input latch on the falling edge of this signal (see Interface Logic Information section). The AD7840 should be powered-up with $\overline{LDAC}$ high. For applications where $\overline{LDAC}$ is permanently low, an R, C is required for correct power-up (see Figure 19).

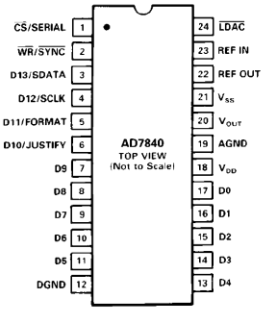
Table I. Serial Data Modes



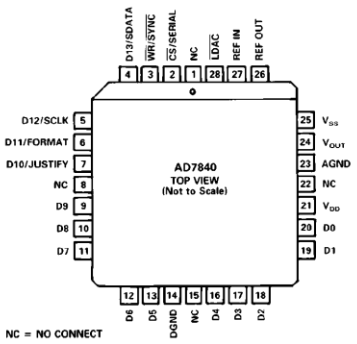
FORMAT	JUSTIFY	MODE
0	0	M1
0	1	M2
1	0	M3
1	1	M4

PIN CONFIGURATIONS

DIP/SSOP



PLCC



D/A SECTION

The AD7840 contains a 14-bit voltage mode D/A converter consisting of highly stable thin film resistors and high speed NMOS single-pole, double-throw switches. The simplified circuit diagram for the DAC section is shown in Figure 1. The three MSBs of the data word are decoded to drive the seven switches A–G. The 11 LSBs switch an 11-bit R-2R ladder structure. The output voltage from this converter has the same polarity as the reference voltage, REF IN.

The REF IN voltage is internally buffered by a unity gain amplifier before being applied to the D/A converter and the bipolar bias circuitry. The D/A converter is configured and sealed for a 3 V reference and the device is tested with 3 V applied to REF IN. Operating the AD7840 at reference voltages outside the ±5% tolerance range may result in degraded performance from the part.

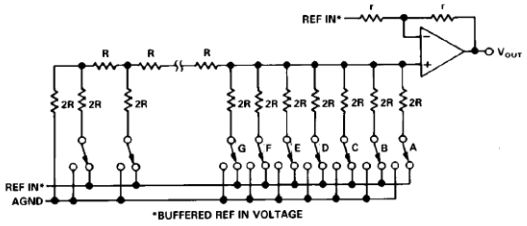


Figure 1. DAC Ladder Structure

INTERNAL REFERENCE

The AD7840 has an on-chip temperature compensated buried Zener reference (see Figure 2) which is factory trimmed to 3 V ±10 mV. The reference voltage is provided at the REF OUT pin. This reference can be used to provide both the reference voltage for the D/A converter and the bipolar bias circuitry. This is achieved by connecting the REF OUT pin to the REF IN pin of the device.

The reference voltage can also be used as a reference for other components and is capable of providing up to 500 µA to an external load. The maximum recommended capacitance on REF OUT for normal operation is 50 pF. If the reference is required for external use, it should he decoupled to AGND with a 200 Ω resistor in series with a parallel combination of a 10 µF tantalum capacitor and a 0.1 µF ceramic capacitor.

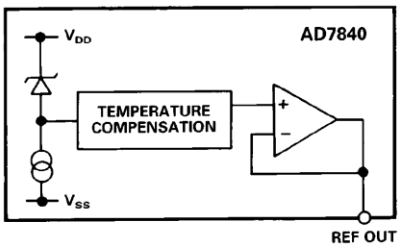


Figure 2. Internal Reference

EXTERNAL REFERENCE

In some applications, the user may require a system reference or some other external reference to drive the AD7840 reference input. Figure 3 shows how the AD586 5 V reference can be conditioned to provide the 3 V reference required by the AD7840 REF IN. An alternate source of reference voltage for the AD7840 in systems which use both a DAC and an ADC is to use the REF OUT voltage of ADCs such as the AD7870 and AD7871. A circuit showing this arrangement is shown in Figure 20.

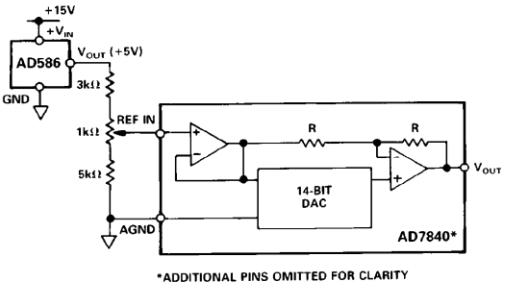


Figure 3. AD586 Driving AD7840 REF IN

OP AMP SECTION

The output from the voltage mode DAC is buffered by a noninverting amplifier. Internal scaling resistors on the AD7840 configure an output voltage range of ±3 V for an input reference voltage of +3 V. The arrangement of these resistors around the output op amp is as shown in Figure 1. The buffer amplifier is capable of developing ±3 V across a 2 kΩ and 100 pF load to ground and can produce 6 V peak-to-peak sine wave signals to a frequency of 20 kHz. The output is updated on the falling edge of the $\overline{\text{LDAC}}$ input. The amplifier settles to within 1/2 LSB of its final value in typically less than 2.5 µs.

The small signal (200 mV p-p) bandwidth of the output buffer amplifier is typically 1 MHz. The output noise from the amplifier is low with a figure of 30 nV/√Hz at a frequency of 1 kHz. The broadband noise from the amplifier exhibits a typical peak-to-peak figure of 150 µV for a 1 MHz output bandwidth. Figure 4 shows a typical plot of noise spectral density versus frequency for the output buffer amplifier and for the on-chip reference.

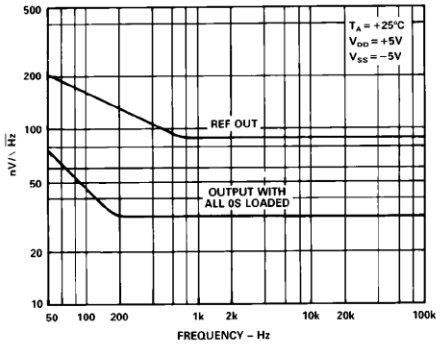


Figure 4. Noise Spectral Density vs. Frequency

TRANSFER FUNCTION

The basic circuit configuration for the AD7840 is shown in Figure 5. Table II shows the ideal input code to output voltage relationship for this configuration. Input coding to the DAC is 2s complement with 1 LSB = FS/16,384 = 6 V/16,384 = 366 µV.

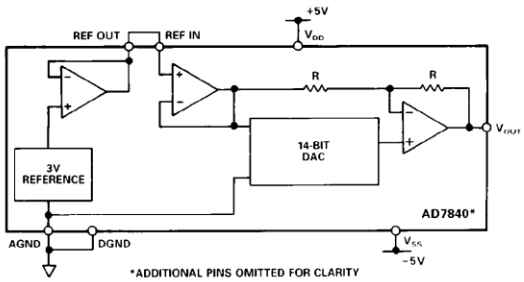


Figure 5. AD7840 Basic Connection Diagram

Table II. Ideal Input/Output Code Table

DAC Latch Contents MSB LSB	Analog Output, V_{OUT}*
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	+2.999634 V
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0	+2.999268 V
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	+0.000366 V
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 V
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	–0.000366 V
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	–2.999634 V
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	–3 V

*Assuming REF IN = +3 V.

The output voltage can be expressed in terms of the input code, N, using the following expression:

$$V_{OUT} = \frac{2 \times N \times REFIN}{16384} \quad -8192 \le N \le +8191$$

INTERFACE LOGIC INFORMATION

The AD7840 contains two 14-bit latches, an input latch and a DAC latch. Data can be loaded to the input latch in one of two basic interface formats. The first is a parallel 14-bit wide data word; the second is a serial interface where 16 bits of data are serially clocked into the input latch. In the parallel mode, $\overline{\text{CS}}$ and $\overline{\text{WR}}$ control the loading of data. When the serial data format is selected, data is loaded using the SCLK, $\overline{\text{SYNC}}$ and SDATA serial inputs. Data is transferred from the input latch to the DAC latch under control of the $\overline{\text{LDAC}}$ signal. Only the data in the DAC latch determines the analog output of the AD7840.

Parallel Data Format

Table III shows the truth table for AD7840 parallel mode operation. The AD7840 normally operates with a parallel input data format. In this case, all 14 bits of data (appearing on data inputs D13 (MSB) through D0 (LSB)) are loaded to the AD7840 input latch at the same time. $\overline{\text{CS}}$ and $\overline{\text{WR}}$ control the loading of this data. These control signals are level-triggered; therefore, the input latch can be made transparent by holding both signals at a logic low level. Input data is latched into the input latch on the rising edge of $\overline{\text{CS}}$ or $\overline{\text{WR}}$.

The DAC latch is also level triggered. The DAC output is normally updated on the falling edge of the $\overline{\text{LDAC}}$ signal. However, both latches cannot become transparent at the same time. Therefore, if $\overline{\text{LDAC}}$ is hardwired low, the part operates as follows; with $\overline{\text{LDAC}}$ low and $\overline{\text{CS}}$ and $\overline{\text{WR}}$ high, the DAC latch is transparent. When $\overline{\text{CS}}$ and $\overline{\text{WR}}$ go low (with $\overline{\text{LDAC}}$ still low), the input latch becomes transparent but the DAC latch is disabled. When $\overline{\text{CS}}$ or $\overline{\text{WR}}$ return high, the input latch is locked out and the DAC latch becomes transparent again and the DAC output is updated. The write cycle timing diagram for parallel data is shown in Figure 6. Figure 7 shows the simplified parallel input control logic for the AD7840.

Table III. Parallel Mode Truth Table

$\overline{CS}$	$\overline{WR}$	$\overline{LDAC}$	Function
H	X	H	} Both Latches Latched
X	H	H	
L	L	H	Input Latch Transparent
H	H	L	} Input Latch Latched
H	X	L	} DAC Latch Transparent
X	H	L	} Analog Output Updated
↑	↑	L	Input Latch Transparent DAC Latch Data Transfer Inhibited
L	↓	L	} Input Latch Is Latched
↓	L		} DAC Latch Data Transfer Occurs

X = Don't Care

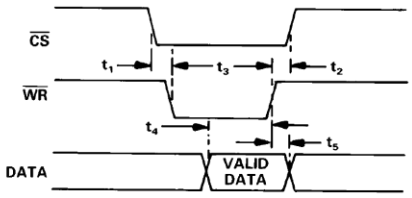


Figure 6. Parallel Mode Timing Diagram

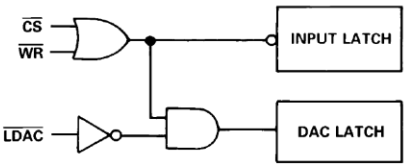


Figure 7. AD7840 Simplified Parallel Input Control Logic

Serial Data Format

The serial data format is selected for the AD7840 by connecting the $\overline{CS}$/SERIAL line to –5 V. In this case, the $\overline{WR}$/$\overline{SYNC}$, D13/SDATA, D12/SCLK, D11/FORMAT and D10/JUSTIFY pins all assume their serial functions. The unused parallel inputs should not be left unconnected to avoid noise pickup. Serial data is loaded to the input latch under control of SCLK, $\overline{SYNC}$ and SDATA. The AD7840 expects a 16-bit stream of serial data on its SDATA input. Serial data must be valid on the falling edge of SCLK. The $\overline{SYNC}$ input provides the frame synchronization signal which tells the AD7840 that valid serial data will be available for the next 16 falling edges of SCLK. Figure 8 shows the timing diagram for serial data format.

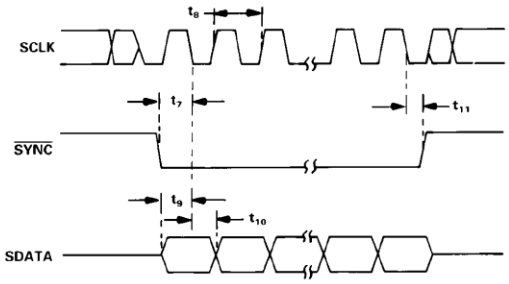


Figure 8. Serial Mode Timing Diagram

Although 16 bits of data are clocked into the AD7840, only 14 bits go into the input latch. Therefore, two bits in the stream are don't cares since their value does not affect the input latch data. The order and position in which the AD7840 accepts the 14 bits of input data depends upon the FORMAT and JUSTIFY inputs. There are four different input data modes which can be chosen (see Table I in the Pin Function Description section).

The first mode (M1) assumes that the first two bits of the input data stream are don't cares, the third bit is the LSB and the last (or 16th bit) is the MSB. This mode is chosen by tying both the FORMAT and JUSTIFY pins to a logic 0. The second mode (M2; FORMAT = 0, JUSTIFY = 1) assumes that the first bit in the data stream is the LSB, the fourteenth bit is the MSB and the last two bits are don't cares. The third mode (M3; FORMAT= 1, JUSTIFY 0) assumes that the first two bits in the stream are again don't cares, the third bit is now the MSB and the sixteenth bit is the LSB. The final mode (M4; FORMAT = 1, JUSTIFY= 1) assumes that the first bit is the MSB, the fourteenth bit is the LSB and the last two bits of the stream are don't cares.

As in the parallel mode, the $\overline{LDAC}$ signal controls the loading of data to the DAC latch. Normally, data is loaded to the DAC latch on the falling edge of $\overline{LDAC}$. However, if $\overline{LDAC}$ is held low, then serial data is loaded to the DAC latch on the sixteenth falling edge of SCLK. If $\overline{LDAC}$ goes low during the transfer of serial data to the input latch, no DAC latch update takes place on the falling edge of $\overline{LDAC}$. If $\overline{LDAC}$ stays low until the serial transfer is completed, then the update takes place on the sixteenth falling edge of SCLK. If $\overline{LDAC}$ returns high before the serial data transfer is completed, no DAC latch update takes place. Figure 9 shows the simplified serial input control logic for the AD7840.

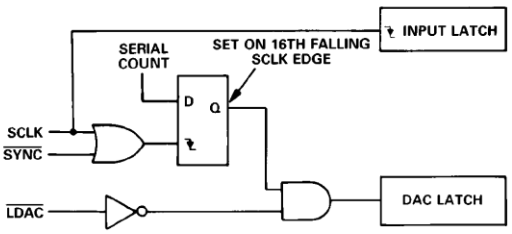


Figure 9. AD7840 Simplified Serial Input Control Logic

AD7840 DYNAMIC SPECIFICATIONS

The AD7840 is specified and 100% tested for dynamic performance specifications as well as traditional dc specifications such as integral and differential nonlinearity. These ac specifications are required for the signal processing applications such as speech synthesis, servo control and high speed modems. These applications require information on the DAC's effect on the spectral content of the signal it is creating. Hence, the parameters for which the AD7840 is specified include signal-to-noise ratio, harmonic distortion and peak harmonics. These terms are discussed in more detail in the following sections.

Signal-to-Noise Ratio (SNR)

SNR is the measured signal-to-noise ratio at the output of the DAC. The signal is the rms magnitude of the fundamental. Noise is the rms sum of all the nonfundamental signals up to half the sampling frequency (fs/2) excluding dc. SNR is dependent upon the number of quantization levels used in the digitization process; the more levels, the smaller the quantization noise. The theoretical signal to noise ratio for a sine wave output is given by

$$SNR = (6.02N + 1.76)\ dB \qquad (1)$$

where N is the number of bits. Thus for an ideal 14-bit converter, SNR = 86 dB.

Figure 10 shows a typical 2048 point Fast Fourier Transform (FFT) plot of the AD7840KN with an output frequency of 1 kHz and an update rate of 100 kHz. The SNR obtained from this graph is 81.8 dB. It should be noted that the harmonics are taken into account when calculating the SNR.

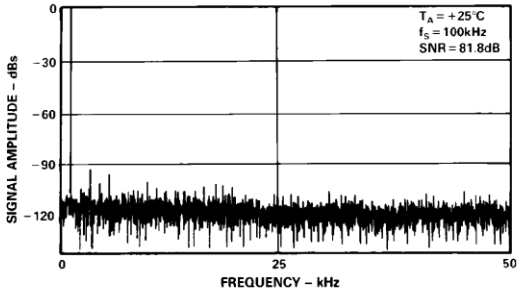


Figure 10. AD7840 FFT Plot

Effective Number of Bits

The formula given in (1) relates the SNR to the number of bits. Rewriting the formula, as in (2) it is possible to get a measure of performance expressed in effective number of bits (N).

$$N = \frac{SNR - 1.76}{6.02} \qquad (2)$$

The effective number of bits for a device can be calculated directly from its measured SNR.

Harmonic Distortion

Harmonic distortion is the ratio of the rms sum of harmonics to the fundamental. For the AD7840, total harmonic distortion (THD) is defined as

$$THD = 20\log\frac{\sqrt{V_2^2 + V_3^2 + V_4^2 + V_5^2 + V_6^2}}{V_1}$$

where V_1 is the rms amplitude of the fundamental and V_2, V_3, V_4, V_5 and V_6 are the rms amplitudes of the second through the sixth harmonic. The THD is also derived from the 2048-point FFT plot.

Peak Harmonic or Spurious Noise

Peak harmonic or spurious noise is defined as the ratio of the rms value of the next largest component in the DAC output spectrum (up to fs/2 and excluding dc) to the rms value of the fundamental. Normally, the value of this specification will be determined by the largest harmonic in the spectrum, but for parts where the harmonics are buried in the noise floor the peak will be a noise peak.

Testing the AD7840

A simplified diagram of the method used to test the dynamic performance specifications is outlined in Figure 11. Data is loaded to the AD7840 under control of the microcontroller and associated logic at a 100 kHz update rate. The output of the AD7840 is applied to a ninth order, 50 kHz, low-pass filter. The output of the filter is in turn applied to a 16-bit accurate digitizer. This digitizes the signal and the microcontroller generates an FFT plot from which the dynamic performance of the AD7840 can be evaluated.

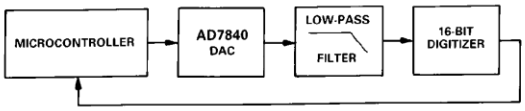


Figure 11. AD7840 Dynamic Performance Test Circuit

The digitizer sampling is synchronized with the AD7840 update rate to ease FFT calculations. The digitizer samples the AD7840 after the output has settled to its new value. Therefore, if the digitizer was to sample the output directly it would effectively be sampling a dc value each time. As a result, the dynamic performance of the AD7840 would not be measured correctly. Using the digitizer directly on the AD7840 output would give better results than the actual performance of the AD7840. Using a filter between the DAC and the digitizer means that the digitizer samples a continuously moving signal and the true dynamic performance of the AD7840 is measured.

Some applications will require improved performance versus frequency from the AD7840. In these applications, a simple sample-and-hold circuit such as that outlined in Figure 12 will extend the very good performance of the AD7840 to 20 kHz.

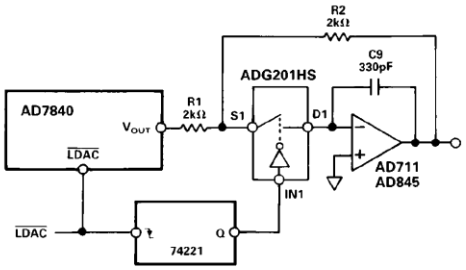


Figure 12. Sample-and-Hold Circuit

Other applications will already have an inherent sample-and-hold function following the AD7840. An example of this type of application is driving a switched-capacitor filter where the updating of the DAC is synchronized with the switched-capacitor filter. This inherent sample-and-hold function also extends the frequency range performance of the AD7840.

Performance versus Frequency

The typical performance plots of Figures 13 and 14 show the AD7840's performance over a wide range of input frequencies at an update rate of 100 kHz. The plot of Figure 13 is without a sample-and-hold on the AD7840 output while the plot of Figure 14 is generated with the sample-and-hold circuit of Figure 12 on the output.

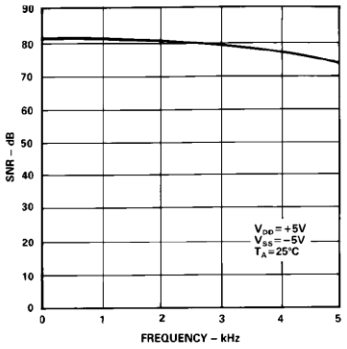


Figure 13. Performance vs. Frequency (No Sample-and-Hold)

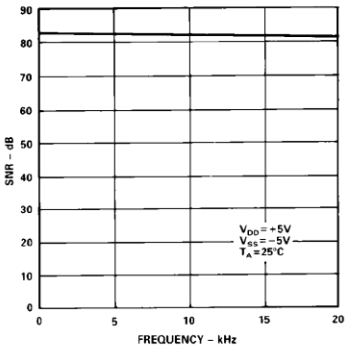


Figure 14. Performance vs. Frequency (with Sample-and-Hold)

MICROPROCESSOR INTERFACING

The AD7840 logic architecture allows two interfacing options for interfacing the part to microprocessor systems. It offers a 14-bit wide parallel format and a serial format. Fast pulse widths and data setup times allow the AD7840 to interface directly to most microprocessors including the DSP processors. Suitable interfaces to various microprocessors are shown in Figures 15 to 23.

Parallel Interfacing

Figures 15 to 17 show interfaces to the DSP processors, the ADSP-2100, the TMS32010 and TMS32020. An external timer controls the updating of the AD7840. Data is loaded to the AD7840 input latch using the following instructions:

ADSP-2100: DM(DAC) = MR0
TMS32010: OUT DAC,D
TMS32020: OUT DAC,D
MR0 = ADSP-2100 MR0 Register
D = Data Memory Address
DAC = AD7840 Address

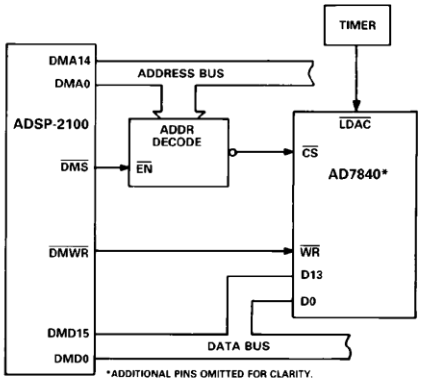


Figure 15. AD7840–ADSP-2100 Parallel Interface

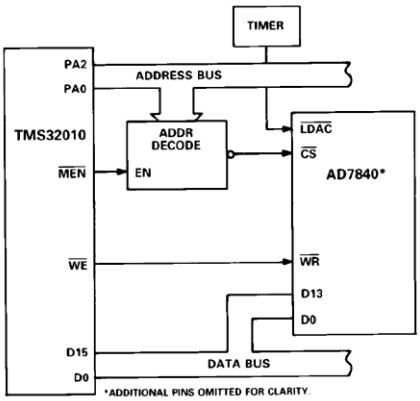


Figure 16. AD7840–TMS32010 Parallel Interface

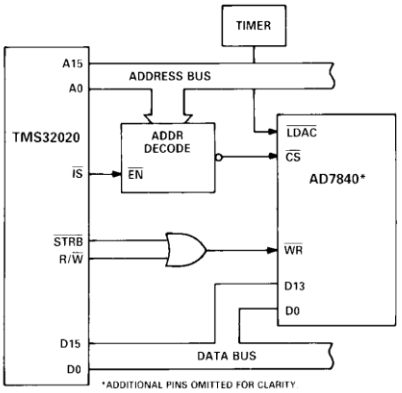


Figure 17. AD7840–TMS32020 Parallel Interface

Some applications may require that the updating of the AD7840 DAC latch be controlled by the microprocessor rather than the external timer. One option (for double-buffered interfacing) is to decode the AD7840 $\overline{\text{LDAC}}$ from the address bus so that a write operation to the DAC latch (at a separate address than the input latch) updates the output. An example of this is shown in the 8086 interface of Figure 18. Note that connecting the $\overline{\text{LDAC}}$ input to the $\overline{\text{CS}}$ input will not load the DAC latch correctly since both latches cannot he transparent at the same time.

AD7840–8086 Interface

Figure 18 shows an interface between the AD7840 and the 8086 microprocessor. For this interface, the $\overline{\text{LDAC}}$ input is derived from a decoded address. If the least significant address line, A0, is decoded then the input latch and the DAC latch can reside at consecutive addresses. A move instruction loads the input latch while a second move instruction updates the DAC latch and the AD7840 output. The move instruction to load a data word WXYZ to the input latch is as follows:

MOV DAC,#YZWX
DAC = AD7840 Address

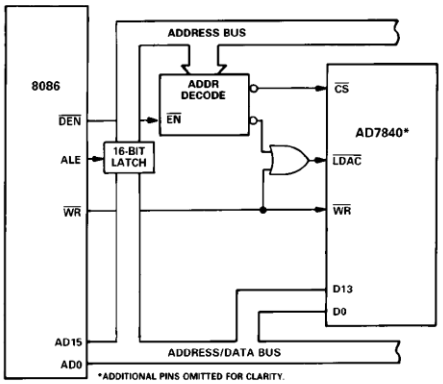


Figure 18. AD7840–8086 Parallel Interface

AD7840–68000 Interface
An interface between the AD7840 and the 68000 microprocessor is shown in Figure 19. In this interface example, the $\overline{LDAC}$ input is hardwired low. As a result the DAC latch and analog output are updated on the rising edge of $\overline{WR}$. A single move instruction, therefore, loads the input latch and updates the output.

MOVE.W D0,$DAC
D0 = 68000 D0 Register
DAC = AD7840 Address

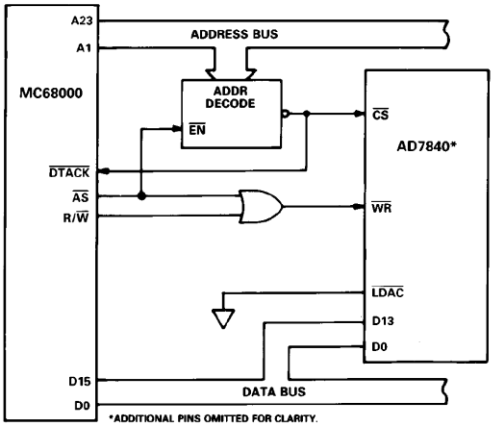


Figure 19. AD7840–MC68000 Parallel Interface

Serial Interfacing
Figures 20 to 23 show the AD7840 configured for serial interfacing with the $\overline{CS}$ input hardwired to –5 V. The parallel bus is not activated during serial communication with the AD7840.

AD7840–ADSP-2101/ADSP-2102 Serial Interface
Figure 20 shows a serial interface between the AD7840 and the ADSP-2101/ADSP-2102 DSP processor. Also included in the interface is the AD7870, a 12-bit A/D converter. An interface such as this is suitable for modem and other applications which have a DAC and an ADC in serial communication with a microprocessor.

The interface uses just one of the two serial ports of the ADSP-2101/ADSP-2102. Conversion is initiated on the AD7870 at a fixed sample rate (e.g., 9.6 kHz) which is provided by a timer or clock recovery circuitry. While communication takes place between the ADC and the ADSP-2101/ ADSP-2102, the AD7870 $\overline{SSTRB}$ line is low. This $\overline{SSTRB}$ line is used to provide a frame synchronization pulse for the AD7840 $\overline{SYNC}$ and ADSP-2101/ADSP-2102 TFS lines. This means that communication between the processor and the AD7840 can only take place while the AD7870 is communicating with the processor. This arrangement is desirable in systems such as modems where the DAC and ADC communication should be synchronous.

The use of the AD7870 SCLK for the AD7840 SCLK and ADSP-2101/ADSP-2102 SCLK means that only one serial port of the processor is used. The serial clock for the AD7870 must be set for continuous clock for correct operation of this interface.

Data from the ADSP-2101/ADSP-2102 is valid on the falling edge of SCLK. The $\overline{LDAC}$ input of the AD7840 is permanently low so the update of the DAC latch and analog output takes place on the sixteenth falling edge of SCLK (with $\overline{SYNC}$ low). The FORMAT pin of the AD7840 must be tied to +5 V and the JUSTIFY pin tied to DGND for this interface to operate correctly.

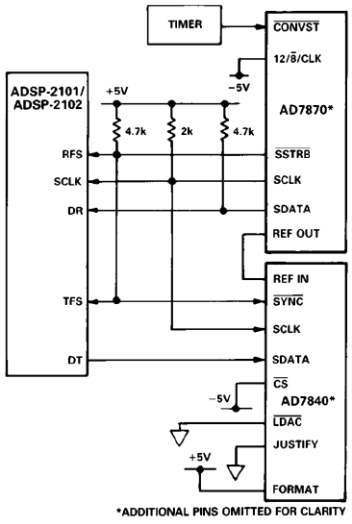


Figure 20. Complete DAC/ADC Serial Interface

AD7840–DSP56000 Serial Interface
A serial interface between the AD7840 and the DSP56000 is shown in Figure 21. The DSP56000 is configured for normal mode synchronous operation with gated clock. It is also set up for a 16-bit word with SCK and SC2 as outputs and the FSL control bit set to a 0. SCK is internally generated on the DSP56000 and applied to the AD7840 SCLK input. Data from the DSP56000 is valid on the falling edge of SCK. The SC2 output provides the framing pulse for valid data. This line must be inverted before being applied to the $\overline{SYNC}$ input of the AD7840.

The $\overline{LDAC}$ input of the AD7840 is connected to DGND so the update of the DAC latch takes place on the sixteenth falling edge of SCLK. As with the previous interface, the FORMAT pin of the AD7840 must be tied to +5 V and the JUSTIFY pin tied to DGND.

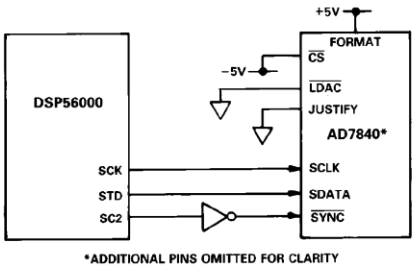


Figure 21. AD7840–DSP56000 Serial Interface

AD7840–TMS32020 Serial Interface

Figure 22 shows a serial interface between the AD7840 and the TMS32020 DSP processor. In this interface, the CLKX and FSX pin of the TMS32020 are generated from the clock/timer circuitry. The same clock/timer circuitry generates the $\overline{\text{LDAC}}$ signal of the AD7840 to synchronize the update of the output with the serial transmission. The FSX pin of the TMS32020 must be configured as an input.

Data from the TMS32020 is valid on the falling edge of CLKX. Once again, the FORMAT pin of the AD7840 must be tied to +5 V while the JUSTIFY pin must be tied to DGND.

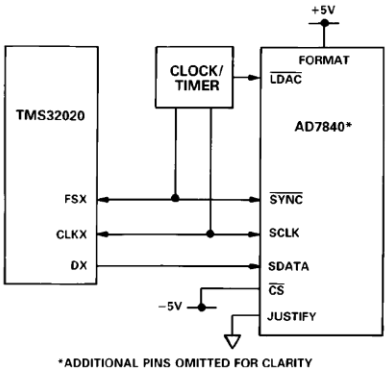


Figure 22. AD7840–TMS32020 Serial Interface

AD7840–NEC7720 Serial Interface

A serial interface between the AD7840 and the NEC7720 is shown in Figure 23. The serial clock must be inverted before being applied to the AD7840 SCLK input because data from the processor is valid on the rising edge of SCK.

The NEC7720 is programmed for the LSB to be the first bit in the serial data stream. Therefore, the AD7840 is set up with the FORMAT pin tied to DGND and the JUSTIFY pin tied to +5 V.

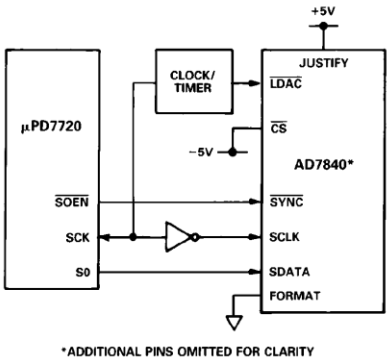


Figure 23. AD7840–NEC7720 Serial Interface

APPLYING THE AD7840

Good printed circuit board layout is as important as the overall circuit design itself in achieving high speed converter performance. The AD7840 works on an LSB size of 366 μV. Therefore, the designer must be conscious of minimizing noise in both the converter itself and in the surrounding circuitry. Switching mode power supplies are not recommended as the switching spikes can feed through to the on-chip amplifier. Other causes of concern are ground loops and digital feedthrough from microprocessors. These are factors which influence any high performance converter, and a proper PCB layout which minimizes these effects is essential for best performance.

LAYOUT HINTS

Ensure that the layout for the printed circuit board has the digital and analog lines separated as much as possible. Take care not to run any digital track alongside an analog signal track. Establish a single point analog ground (star ground) separate from the logic system ground. Place this star ground as close as possible to the AD7840 as shown in Figure 24. Connect all analog grounds to this star ground and also connect the AD7840 DGND pin to this ground. Do not connect any other digital grounds to this analog ground point.

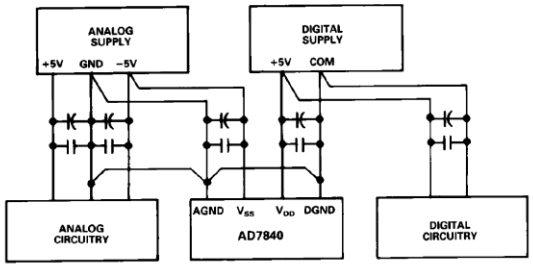


Figure 24. Power Supply Grounding Practice

Low impedance analog and digital power supply common returns are essential to low noise operation of high performance converters. Therefore, the foil width for these tracks should be kept as wide as possible. The use of ground planes minimizes impedance paths and also guards the analog circuitry from digital noise. The circuit layouts of Figures 27 and 28 have both analog and digital ground planes which are kept separated and only joined at the star ground close to the AD7840.

NOISE

Keep the signal leads on the V_{OUT} signal and the signal return leads to AGND as short as possible to minimize noise coupling. In applications where this is not possible, use a shielded cable between the DAC output and its destination. Reduce the ground circuit impedance as much as possible since any potential difference in grounds between the DAC and its destination device appears as an error voltage in series with the DAC output.

DATA ACQUISITION BOARD

Figure 25 shows the AD7840 in a data acquisition circuit. The corresponding printed circuit board (PCB) layout and silkscreen are shown in Figures 26 to 28. The board layout has three interface ports: one serial and two parallel. One of the parallel ports is directly compatible with the ADSP-2100 evaluation board expansion connector.

Some systems will require the addition of a re-construction filter on the output of the AD7840 to complete the data acquisition system. There is a component grid provided near the analog output on the PCB which may be used for such a filter or any other output conditioning circuitry. To facilitate this option, there is a shorting plug (labeled LK1 on the PCB) on the analog output track. If this shorting plug is used, the analog output connects to the output of the AD7840; otherwise this shorting plug can be omitted and a wire link used to connect the analog output to the PCB component grid.

The board also contains a simple sample-and-hold circuit which can be used on the output of the AD7840 to extend the very good performance of the AD7840 over a wider frequency range. A second wire link (labelled LK2 on the PCB) connects V_{OUT} (SKT1) to either the output of this sample-and-hold circuit or directly to the output of the AD7840.

INTERFACE CONNECTIONS

There are two parallel connectors, labeled SKT4 and SKT6, and one serial connector, labeled SKT5. A shorting plug option (LK8 in Figure 25) on the AD7840 $\overline{CS}$/SERIAL input configures the DAC for the appropriate interface (see Pin Function Description).

SKT6 is a 96-contact (3-row) Eurocard connector which is directly compatible with the ADSP-2100 Evaluation Board Prototype Expansion Connector. The expansion connector on the ADSP-2100 has eight decoded chip enable outputs labeled $\overline{ECE1}$ to $\overline{ECE8}$. $\overline{ECE6}$ is used to drive the AD7840 $\overline{CS}$ input on the data acquisition board. To avoid selecting on-board sockets at the same time, LK6 on the ADSP-2100 board must be removed. The AD7840 and ADSP-2100 data lines are aligned for left justified data transfer.

SKT4 is a 26-way (2-row) IDC connector. This connector contains the same signal contacts as SKT6 and in addition contains decoded R/$\overline{W}$ and $\overline{STRB}$ inputs which are necessary for TMS32020 interfacing. This decoded $\overline{WR}$ can be selected via LK4. The pinout for this connector is shown in Figure 29.

SKT5 is a nine-way D-type connector which is meant for serial interfacing only. The evaluation board has the facility to invert $\overline{SYNC}$ line via LK7. This is necessary for serial interfacing between the AD7840 and DSP processors such as the DSP56000. The SKT5 pinout is shown in Figure 30.

SKT1, SKT2 and SKT3 are three BNC connectors which provide connections for the analog output, the $\overline{LDAC}$ input and an external reference input. The use of an external reference is optional; the shorting plug (LK3) connects the REF IN pin to either this external reference or to the AD7840's own internal reference.

Wire links LK5 and LK6 connect the D11 and D10 inputs to the data lines for parallel operation. In the serial mode, these links allow the user to select the required format and justification for serial data (see Table I).

POWER SUPPLY CONNECTIONS

The PCB requires two analog power supplies and one 5 V digital supply. Connections to the analog supplies are made directly to the PCB as shown on the silkscreen in Figure 26. The connections are labelled V+ and V– and the range for both of these supplies is 12 V to 15 V. Connection to the 5 V digital supply is made through any of the connectors (SKT4 to SKT6). The –5 V analog supply required by the AD7840 is generated from a voltage regulator on the V– power supply input (IC5 in Figure 25).

SHORTING PLUG OPTIONS

There are eight shorting plug options which must be set before using the board. These are outlined below:

LK1	Connects the analog output to SKT1. The analog output may also be connected to a component grid for signal conditioning.
LK2	Selects either the AD7840 V_{OUT} or the sample-and-hold output.
LK3	Selects either the internal or external reference.
LK4	Selects the decoded R/$\overline{W}$ and $\overline{STRB}$ inputs for TMS32020 interfacing.
LK5	Configures the D11/FORMAT input.
LK6	Configures the D10/JUSTIFY input.
LK7	Selects either the inverted or noninverted $\overline{SYNC}$.
LK8	Selects either parallel or serial interfacing.

COMPONENT LIST

IC1	AD7840 Digital-to-Analog Converter
IC2	AD711 Op Amp
IC3	ADG201HS High Speed Switch
IC4	74HC221 Monostable
IC5	79L05 Voltage Regulator
IC6	74HC02
C1, C3, C5, C7, C11, C13, C15, C17	10 µF Capacitors
C2, C4, C6, C8, C12, C14, C16, C18	0.1 µF Capacitors
C9	330 pF Capacitor
C10	68 pF Capacitor
R1, R2	2.2 kΩ Resistors
R3	15 kΩ Resistor
RP1, RP2	100 kΩ Resistor Packs
LK1, LK2, LK3, LK4, LK5, LK6, LK7, LK8	Shorting Plugs
SKT1, SKT2, SKT3	BNC Sockets
SKT4	26-Contact (2-Row) IDC Connector
SKT5	9-Contact D-Type Connector
SKT6	96-Contact (3-Row) Eurocard Connector

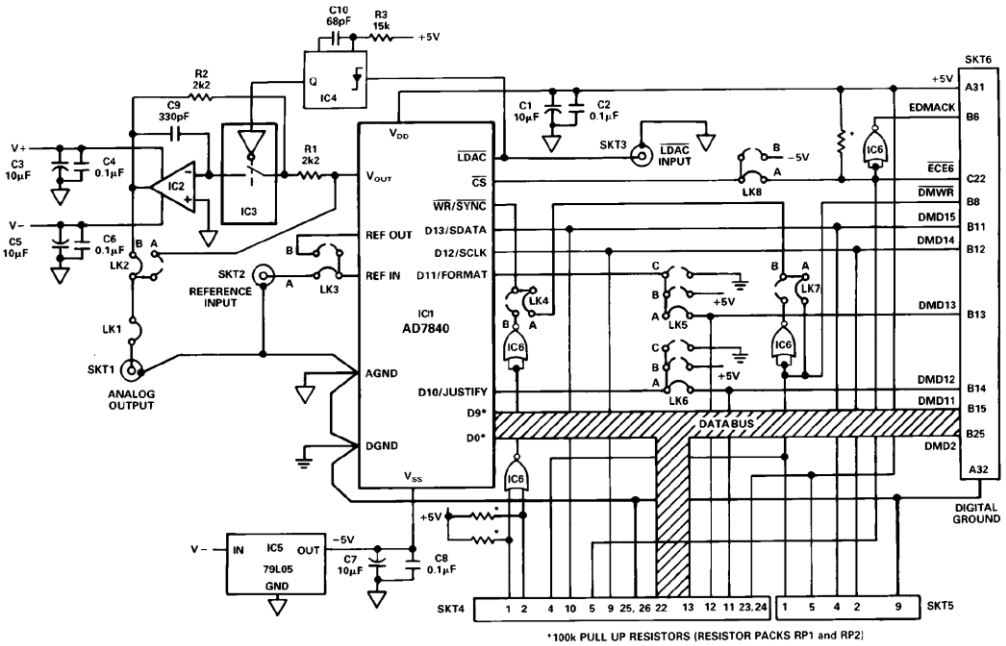


Figure 25. Data Acquisition Circuit Using the AD7840

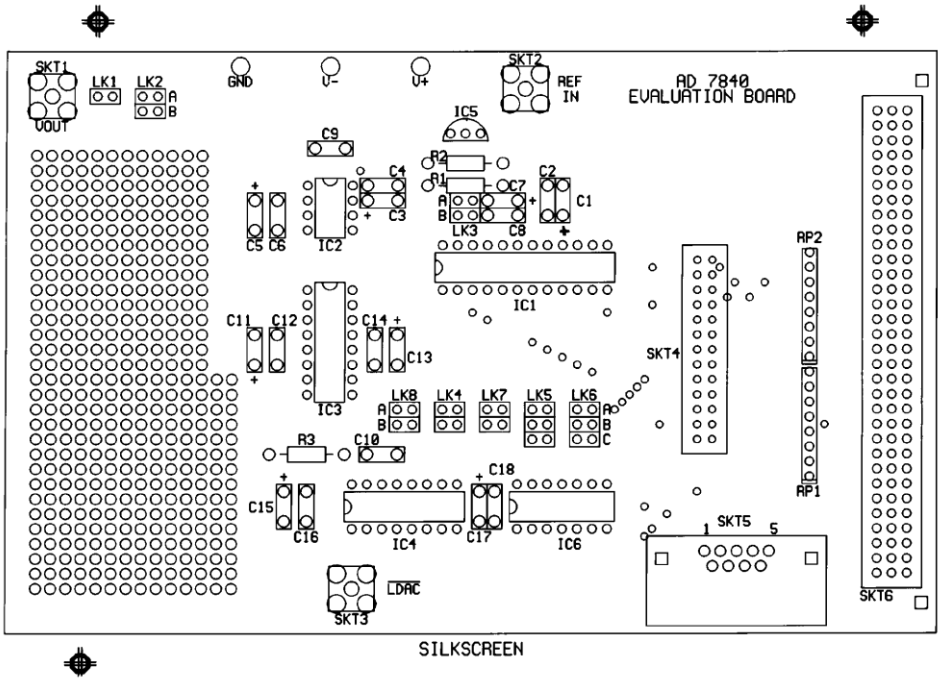


Figure 26. PCB Silkscreen for Figure 25

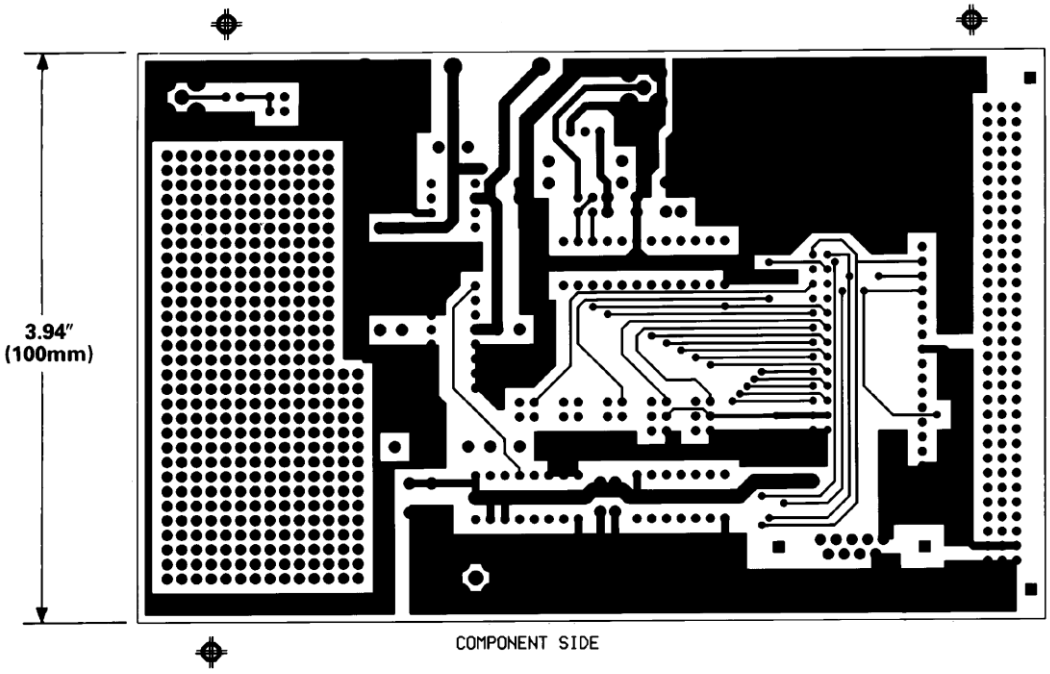


Figure 27. PCB Component Side Layout for Figure 25

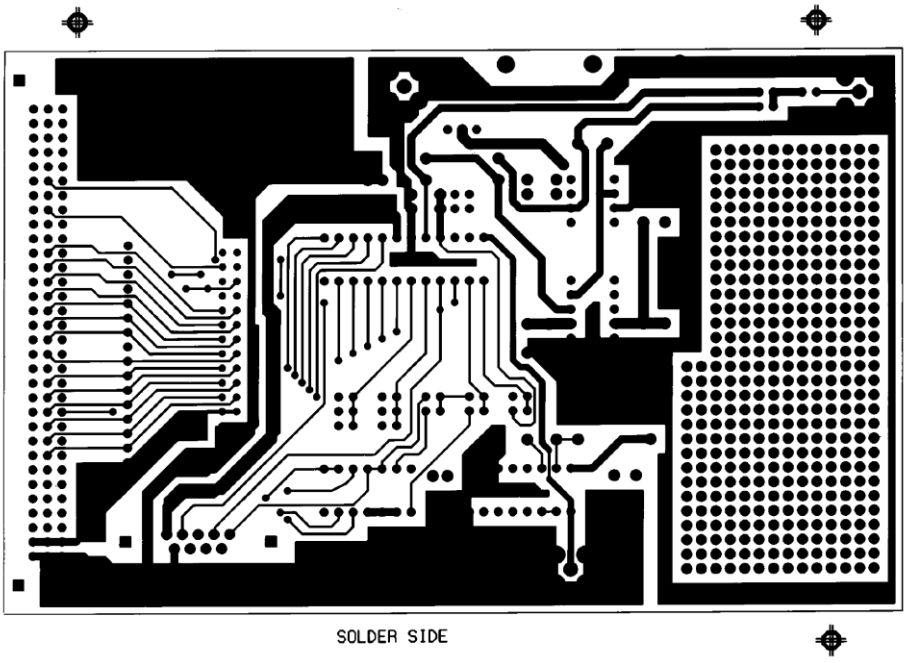


Figure 28. PCB Solder Side Layout for Figure 25

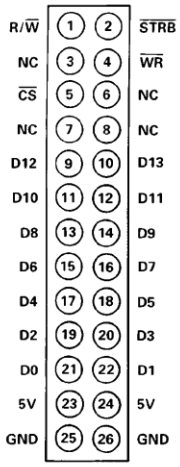


Figure 29. SKT4, IDC Connector Pinout

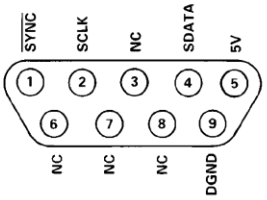
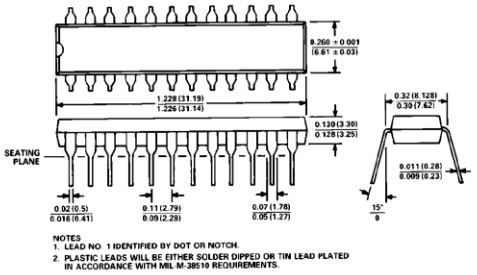


Figure 30. SKT5, D-Type Connector Pinout

OUTLINE DIMENSIONS

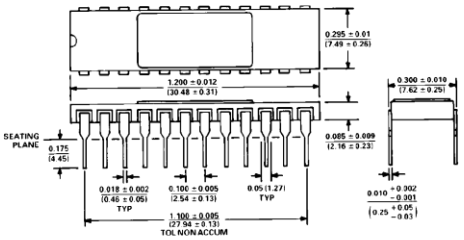
Dimensions shown in inches and (mm).

Plastic DIP (N-24)



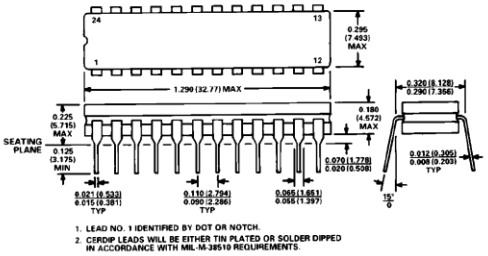
NOTES
1. LEAD NO. 1 IDENTIFIED BY DOT OR NOTCH.
2. PLASTIC LEADS WILL BE EITHER SOLDER DIPPED OR TIN LEAD PLATED IN ACCORDANCE WITH MIL-M-38510 REQUIREMENTS.

Ceramic DIP (D-24A)



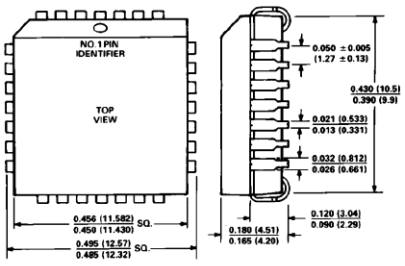
NOTES
1. LEAD NO. 1 IDENTIFIED BY DOT OR NOTCH.
2. CERAMIC DIP LEADS WILL BE EITHER GOLD OR TIN PLATED IN ACCORDANCE WITH MIL-M-385 TO REQUIREMENTS.
3. METAL LID IS CONNECTED TO DGND.

Cerdip (Q-24)



1. LEAD NO. 1 IDENTIFIED BY DOT OR NOTCH.
2. CERDIP LEADS WILL BE EITHER TIN PLATED OR SOLDER DIPPED IN ACCORDANCE WITH MIL-M-38510 REQUIREMENTS.

PLCC (P-28A)



C1259–10–2/89

PRINTED IN U.S.A.

APPENDIX G

Performance characteristics

Bandwidth (-3dB)[1, 2]	DSO1052B:	DC to 50 MHz
	DSO1002A, DSO1004A:	DC to 60 MHz
	DSO1072B:	DC to 70 MHz
	DSO1102B, DSO1012A, DSO1014A:	DC to 100 MHz
	DSO1152B:	DC to 150 MHz
	DSO1022A, DSO1024A:	DC to 200 MHz
Real-time sample rate	2 GSa/sec half channel interleaved, 1 GSa/sec all channels (A models)	
	1 GSa/sec half channel interleaved, 500 MSa/sec all channela (B models)	
Memory depth	20 kpts half channel interleaved, 10 kpts all channels (A models)	
	16 kpts half channel interleaved, 8 kpts all channels (B models)	
Channels	DSO1052B, DSO1002A, DSO1072B, DSO1102B, DSO1012A, DSO1152B, DSO1022A : 2 channels	
	DSO1004A, DSO1014A, DSO1024A : 4 channels	
Vertical resolution	8 bits	
Vertical sensitivity (range)	2 mV/div to 10 V/div	
DC gain accuracy[1]	2 mV/div to 5 mV/div: ± 4.0% full scale (A and B models)	
	10 mV/div to 5 V/div: ± 3.0% full scale (A models only)	
	10 mV/div to 10 V/div: ± 3.0% full scale (B models only)	
Vertical zoom	Vertical expand	
Maximum input voltage	CAT I 300 Vrms, 400 Vpk; transient overvoltage 1.6kVpk	
Dynamic range	+6 divisions from center screen	
Time-base range	DSO1022A, DSO1024A:	1 nsec/div to 50 sec/div
	DSO1012A, DSO1014A, DSO1102B:	2 nsec/div to 50 sec/div
	DSO1002A, DSO1004A, DSO1052B, DSO1072B:	5 nsec/div to 50 sec/div
Selectable BW limit	20 MHz	
Horizontal modes	Main (Y-T), XY, delayed zoom and roll	
Input coupling	DC, AC and ground	
Input impedance	1 MΩ ±1% in parallel with 18 pF ± 3 pF (A models)	
	1 MΩ ±2% in parallel with 15 pF ± 3 pF (B models)	
Time scale accuracy[1]	± 50 ppm from 0 °C to 30 °C, (A models)	
	± 50 ppm + 2 ppm per °C from 30 °C to 45 °C + 5 ppm × (years since manufacture) (A models)	
	± 50 ppm over 1 ms (B models only)	

[1] Denotes warranted specifications, all others are typical. Specifications are valid after a 30-minute warm-up period and ±10°C from firmware calibration temperature.
[2] 20 MHz (when vertical scale is set to < 5 mV)

Performance characteristics

Acquisition modes	
Normal	Displays sampled data directly to the screen in real time
Averaging	Selectable from 2, 4, 8,16, 32, 64, 128 or 256
Sequence	Selectable 1 to 1,000 acquisition frames can be recorded, played back and stored in the scope memory or external USB memory
Peak detect	Captures high-frequency glitches as narrow as 10 nsec (A models) and 20 nsec (B models) when viewing signals at slow sweep speeds
Roll	Waveform display rolls from right to left. minimum horizontal scale setting is 50 msec/div.
Interpolation	Sin (x)/x
Trigger coupling	AC, DC, LF reject, HF reject
Trigger modes	
Force	Triggers immediately when front panel button is pressed
Edge	Triggers on the positive and/or negative slope on any channel
Video	Triggers on NTSC, PAL or SECAM video signals
Pulse width	Triggers on pulse width greater than, equal to or less than a specific time limit, ranging from 20 nsec to 10 sec (A models) and 50ns to 10 sec (B models)
Alternate	Triggers on two non-synchronized active channels
Trigger source	Ch 1, 2, Ext, Ext/5, AC Line (edge only) (2 channel A models) Ch 1, 2, Ext, AC Line (edge only) (B Models) Ch 1, 2, 3, 4, Ext, Ext/5, AC Line (edge only) (4-channel A models)
Trigger sensitivity[1]	≥5 mV/div: 1 div from DC to 10 MHz, 1.5 div from 10 MHz to full bandwidth <5 mV/div: 1 div from DC to 10 MHz, 1.5 div from 10 MHz to 20 MHz
Cursor measurement	Manual, track waveform or automatic measurement selections. Manual and track waveform selections provide readout of Horizontal (X, ΔX) and Vertical (Y, ΔY)
Auto measurement	
Voltage	Maximum, minimum, peak-to-peak, top, base, amplitude, average, RMS, overshoot, preshoot
Time	Period, frequency, rise time, fall time, + width, - width, +duty cycle, -duty cycle, delay A-->B (rising edge), delay A-->B (falling edge), phase A-->B (rising edge) and phase A-->B (falling edge)
Counter	Integrated 6-digit frequency counter on any channel. Counts up to the scope's bandwidth
Display all measurements	Mode to display all single-channel automatic measurements simultaneously on the display
Math functions	A+B, A-B, AxB, FFT Source channel selection for A and B can be any combination of oscilloscope channels 1 and 2 (or 3 and 4 on 4 channel A models).
AutoScale	Finds and displays all active channels, sets edge trigger modes on highest numbered channels, sets vertical sensitivity on channels, time base to display ~2 periods. Requires minimum voltage >20 mVpp, 1% duty cycle and minimum frequency >50 Hz
Display	5.7 inch diagonal color QVGA TFT LCD display with 300 cd/m^2 backlight intensity
Display persistence	OFF, Infinite
Display types	Dots, Vectors
Waveform update rate	400 waveforms/sec (A models) 200 wfm/sec (B models)
Save/Recall internal	10 setups and 10 waveforms can be saved and recalled using internal non-volatile memory locations. 1 reference waveform can be saved and recalled using an internal volatile memory location for visual comparisons.
Save/Recall external	Setups: STP saved and recalled (Note: setups not transferable between A and B models) Waveforms: WFM saved and recalled, CSV saved Reference waveforms: REF saved and recalled for visual comparisons Images: 8-bit BMP, 24-bit BMP, PNG saved

1 Denotes warranted specifications, all others are typical. Specifications are valid after a 30-minute warm-up period and ±10°C from firmware calibration temperature.

Performance characteristics

I/O	
Standard ports	USB 2.0-compliant host port on front panel (A and B models) and rear panel (A models only) compatible with USB flash drives. USB 2.0 device port for PictBridge compatible printing (A and B models) and USBTMC remote PC control (A models only)
Max transfer rate	USB 2.0 full-speed up to 12 Mb/sec
USB flash drive compatibility	Most FAT formatted <2 GB or FAT32 formatted <32 GB flash drives
Printer compatibility	PictBridge-compliant printers via USB device port

General characteristics	
Physical size	12.78 inches W x 6.21 inches H x 5.08 inches D (32.46 cm W x 15.78 cm H x 12.92 cm D) (A models) 11.9 inches W x 6.06 inches H x 5.23 inches D (30.3 cm W X 15.4 cm H X 13.3 cm D) (B models)
Weight	Net: 3.03 kgs (6.68 lbs) Shipping: 4.87 kgs (10.74 lbs) (A models) Net: 2.4 kgs (5.3lbs) Shipping: 3.87 kgs (8.3lbs) (B models)
Probe comp output	Frequency ~1 kHz; Amplitude ~3 V
Scope lock	Secure with a Kensington lock or looped cable through notch built into chassis

Power requirements	
Line range	100-240 VAC, 50/60 Hz ± 10%
Power usage	~60 W max (A models) ~50 W max (B models)

Environmental characteristics (A models)	
Ambient temperature	Operating 0°C to +40°C; non-operating -20°C to +60°C
Humidity	Operating 90% RH at 40°C for 24 hr; non-operating 60% RH at 60°C for 24 hr
Altitude	Operating to 4,400 m (15,000 ft); non-operating to 15,000 m (49,213 ft)
Vibration	Agilent class GP and MIL-PRF-28800F; class 3 random
Shock	Agilent class GP and MIL-PRF-28800F
Pollution degree[2]	Normally only dry non-conductive pollution occurs. Occasionally a temporary conductivity caused by condensation must be expected
Indoor use	Rated for indoor use only

Environmental characteristics (B models)	
Ambient temperature	Operating 10°C to +40°C; non-operating -20°C to +60°C
Cooling Method	Fan force air flow
Humidity	Operating; +35°C or below ≤90 % relative humidity ; non-operating +40°C ≤60 % relative humidity
Altitude Operating	Operating to 3,000 m (9.842 ft); non-operating to 15,000 m (49,213 ft)
Vibration	Agilent class GP and M-PRF-28800F; class 3 random
Shock	Agilent class GP and M-PRF-28800F;
Pollution degree[2]	Normally only dry non-conductive pollution occurs. Occasionally a temporary conductivity caused by condensation must be expected
Regulatory	Safety - UL61010-1:2003, CSA22.2 No. 61010-1:2003, EN61010-1:2001, IEC61010-1:2001. EMI – Passes IEC 61236 -1:2004 / EN 61326-1:2006 Meets EU EMC Directive 2004/108/EC requirements
Indoor use	Rated for indoor use only

Ordering information

2 - Channel Models	Description
DSO1052B	50 MHz, 1 GSa/s, 16 kpts, 2-ch
DSO1072B	70 MHz, 1 GSa/s, 16 kpts, 2-ch
DSO1102B	100 MHz, 1 GSa/s, 16 kpts, 2-ch
DSO1152B	150 MHz, 1 GSa/s, 16 kpts, 2-ch
DSO1022A	200 MHz, 2 GSa/s, 20 kpts, 2-ch

4 - Channel Models	Description
DSO1004A	60 MHz, 2 GSa/s, 20 kpts, 4-ch
DSO1014A	100 MHz, 2 GSa/s, 20 kpts, 4-ch
DSO1024A	200 MHz, 2 GSa/s, 20 kpts, 4-ch

Soft carrying case for 1000 Series

Rackmount kit for 1000A Series only

Accessories included:

- Documentation CD
- Localized front panel overlay (if language option other than English is chosen)
- Power cord
- 10:1 passive probe for each input channel (2 or 4)
- Education student lab guide and professor slide set downloadable free from: www.agilent.com/find/1000edu

Optional accessories:

- N2738A – Soft carrying case for 1000A/B Series
- N2739A – Rackmount kit for 1000A Series (A models only)

Recommended probes

- N2862B – 150 MHz 10:1 passive probe (standard with 50, 60, 70, 100 MHz models)
- N2863B – 300 MHz 10:1 passive probe (standard with 150, 200 MHz models)
- 10070D – 20 MHz 1:1 passive probe
- 10076B – 250 MHz, 100:1, 4 kV passive probe
- N2771B – 50 MHz, 1000:1 30 kV passive probe
- N2791A – 25 MHz, 700V differential probe
- N2891A – 70 MHz, 7 kV differential probe
- 1146A – 100 kHz, 100A AC/DC current probe (requires 9V battery)

Software and Drivers

- IntuiLink toolbar connectivity software downloadable free from www.agilent.com/find/intuilink