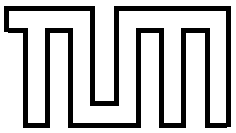
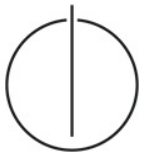


Technische Universität München
Fakultät für Informatik

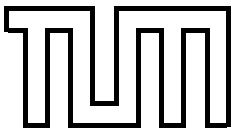


Systementwicklungsprojekt

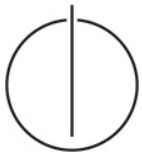
Virtual Engineering

Design of Optical See-through Displays

Stefan Machleidt



Technische Universität München
Fakultät für Informatik



Systementwicklungsprojekt

Virtual Engineering

Design of Optical See-through Displays

Stefan Machleidt

Aufgabenstellerin:	Prof. Gudrun Klinker, Ph.D.
Betreuer:	Marcus Tönnis
Abgabedatum:	15.09.2008

Ich versichere, daß ich dieses Systementwicklungsprojekt selbständig verfaßt und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 15.09.2008 Stefan Machleidt

Zusammenfassung

Die Entwicklung von optischen See-Through Displays stellt besondere Anforderungen an rechnergestütztes optisches Konstruieren. Position und Lage der fokalen Ebene sowie Größe und Form des Bildes sind dabei von besonderer Bedeutung. Diese Arbeit stellt einen Ansatz für die Entwicklung einer Konstruktionssoftware für solche optische System vor und diskutiert einige wichtige Aspekte einer konkreten Implementierung einiger optischer Elemente (z.B. Linsen, Spiegel, etc.). Dafür wurde eine einheitliche Struktur für die optischen Elemente eingeführt, die alle geforderten Objekte abbilden kann. Es wird ein Algorithmus für die Berechnung des optischen Weges und des Bildes beschrieben, der die sequentielle Anordnung der optischen Elemente innerhalb des optischen Systems vorsieht. Außerdem wird ein interaktives Konstruktionstool vorgestellt, das auf diesem Algorithmus basiert. Dieses Tool unterstützt eine intuitive Vorgehensweise auf der Suche nach neuen optischen Anordnungen und vereinfacht das Verständnis der zugrundeliegenden Auswirkungen und Beziehungen. Ergänzend werden einige Aspekte bei der Entwicklung von konformen Head-Up Displays diskutiert.

Abstract

The design of optical see-through displays implicates special requirements to computer-aided optical design. Position and orientation of the focal plane as well as the image size and shape are of particular importance. This document introduces an approach for the design of a construction software for such optical systems and discusses some important aspects of a concrete implementation of particular optical elements (e.g. lenses, mirrors, etc.). A uniform structure for the optical elements was established that is able to represent all desired objects. An algorithm for the computation of the optical path and the image is described, that provides the sequential arrangement of the optical elements within the optical system. Based on this algorithm, an interactive construction tool is presented. The tool features an intuitive proceeding in finding new optical arrangements and to facilitate the understanding of the underlying effects and relationships. In extension, some aspects of the design of a conformal head-up display are discussed.

Overview

Contents

List of Figures

1 Introduction

Design of optical see-through displays and approaches to the implementation of a construction software for optical systems

---

The concept of optical see-through displays appears in a wide range of applications, that varies from head-mounted displays (HMDs) to head-up displays (HUDs), e.g. the HUD in the BMW 5 series or the Lumus video eyeglasses. Indeed, all of them are based on the idea of augmenting the view on the environment with some additional, computer generated information. Thereby, the distances between the observer and the virtual elements play a decisive role in making the perception of the augmentation as pleasant as possible. In the case of a conformal display that combines virtual and real content accordingly, it is even necessary to be able to place single objects on specific positions on the environment. Here attention should be paid to the aspects of human depth perception, to provide a consistent fusion of the virtual objects with the real world. This inevitably leads to a complex arrangement of a large number of optical elements in a restricted space, with the objective of placing the focal plane at a desired position and distance and at the same time providing the target image size. Depending on the particular application, the requirements for construction volume and optical properties may be completely different and may demand new approaches and solutions.

With increasing complexity of the optical arrangement, undesired side effects like aberration of the virtual image may appear. In the majority of cases, for example, the deployment of lenses or curved mirrors produces a distorted image and/or a curved focal plane. Thus, even the effect of a minimal change in alignment of a single object is not evident. Using a thick lens with a short focal length, a difference of a few millimeters in the position of the display often corresponds to many meters with regard to the position of the virtual image. At the same time the level of the distortion varies dramatically. To face these problems in the design of optical see-through displays, it makes sense to simulate the optical elements in a virtual environment.

This document introduces an approach for the implementation of a construction software for optical systems and discusses some important aspects of a concrete implementation of particular optical elements. Its main focus is to provide the possibility to find new optical arrangements by intuitively moving objects and to understand the underlying effects and relationships. Therefore it is necessary to accomplish real-time computation. The expense of ray tracing and therefore the total number of rays to be computed must be minimized. To satisfy this demand, an algorithm was developed that provides the sequential arrangement of the optical elements. In contrast to many other ray tracing procedures, only the relevant part of the rays is computed that actually builds the optical path and reaches the human eye. Depending on the complexity of the optical arrangement and the desired accuracy, the

number of rays can be reduced as well, as different rays often contain redundant information with regard to the computation of the image and the optical path. In many cases a restriction on the four limitative rays that start from the corners of the display and reach the center of the eye after passing each element in the setup provides a completely satisfying result.

In order to realize a high potential for extension, a uniform structure for the optical elements was established that is able to represent all desired objects. The range of optical elements can be extended by deriving them from existing objects or implementing completely new elements without the need for changing the algorithm. At runtime, changes to the setup can easily be made by adding/removing objects to/from the list of elements.

For the purpose of an accurate representation of the optical properties of each element, several approximation algorithms were developed. With increasing accuracy normally the expense of computation increases as well. Therefore it turns out to be reasonable to reduce the resolution of the representation in the early stages of the engineering process, to provide intuitive real-time manipulation of the setup. For a specific construction the accuracy then can be increased to achieve an exacting representation and to identify aberration. Also a dynamic adaptation of the resolution to the processor load was implemented.

As an approach to meet the outlined theoretical requirements and to solve some of the inherent problems, this document presents an interactive construction tool. This tool provides the possibility to grab single objects in the arrangement with the help of so-called manipulators that surround the objects' geometry to change their position and orientation. Thus the effect of the manipulation appears promptly, giving feedback of the prevailing dependencies. In contrast, all values representing the position, orientation and size as well as parameters for the optical properties of the object in the optical path can be shown and changed via the graphical user interface (GUI). This offers the options to do virtual reverse engineering as well as exporting a specific construction.

The possibilities of the tool are shown on the example of the development of a conformal, automotive HUD. With given optical components an arrangement was virtually designed to fit the requirements for position and orientation of the focal plane and image size. Based on this construction plan a prototype was built. In extend, some general aspects of the design of such displays are discussed, ranging from basic principles and requirements to the point of image quality and distortion.

2 Related Work

Computer-aided optical design and optical system optimization

The use of computers had an important impact on optical system design since it started in the mid-1950s. It allows the design of increasingly complex and sophisticated optical systems. With the objective of finding the most suitable system that satisfies the requirements of the particular application and at the same time shows the least amount of aberration, the primary tasks are numerical calculations for ray tracing and analysis, automatic system optimization and rendering graphics for system visualisation. These computations are mainly based on ray tracing. Spencer and Murty [17] describe a general procedure for tracing skew rays through optical system. While this procedure is limited to cylindrical and toric surfaces, the automatic ray-surface method of Howell and Wilson [10] is applicable to unusual surface configurations as well.

For evaluation of the image quality, numerous different methods exist. Kirkham [11] basically describes the computation of transverse error curves that refer to intersection points of a fan of rays with the image plane. The visualisation of these intersection points leads to a spot diagram that provides image evaluation as well (Miyamoto [15]). Lucy [14] introduces further image quality criteria also based on tracing skew rays through the optical system.

In order to facilitate and expedite the process of seeking and optimizing an optical system, it makes sense to provide some automatic correction methods. A complete review over the existing optimization techniques is beyond the scope of this document. Rosen and Eldert [16], for example, present a least-squares method for correction of an optical system. Feder [5] describes further procedures for automatic lens design. Gray introduces an aberration theory suitable for computer programs to provide automatic optimization [6] and describes a specific computer program for the intermediate and final stages of a lens design problem [7].

There exist a large number of commercial and free software products that apply one or more of these approaches and provide optical design. CODE V ([1]) and LightTools ([2]) from the Optical Research Associates or Lighttrans' VirtualLab ([13]) are examples for advanced commercial optical design software. The Modern Optical Design and Analysis Software ([12]) is available as freeware, limited up to four optical surfaces.

3 Basic Approach

Structure of the generic algorithm and the representation of the optical elements

The basic approach of the generic algorithm uses the properties of sequential alignment of the optical components with the human eye on the one side and the display on the other. Therefore only the particular successor of each object in the optical path can absorb or transmit the emitted light. On the one hand, this approach allows a fast computation of the optical path and the focal plane, because there is no need for expensive ray tracing. On the other hand it offers an intuitive proceeding, because side effects like opposite occlusion or unintentional multiple occurrence of an object in the optical path are excluded. For this reason the optical path is always completely visible in the maximum size that is possible depending on the display size. This facilitates the adaption of the position, orientation and size of the particular optical elements to achieve the desired image properties.

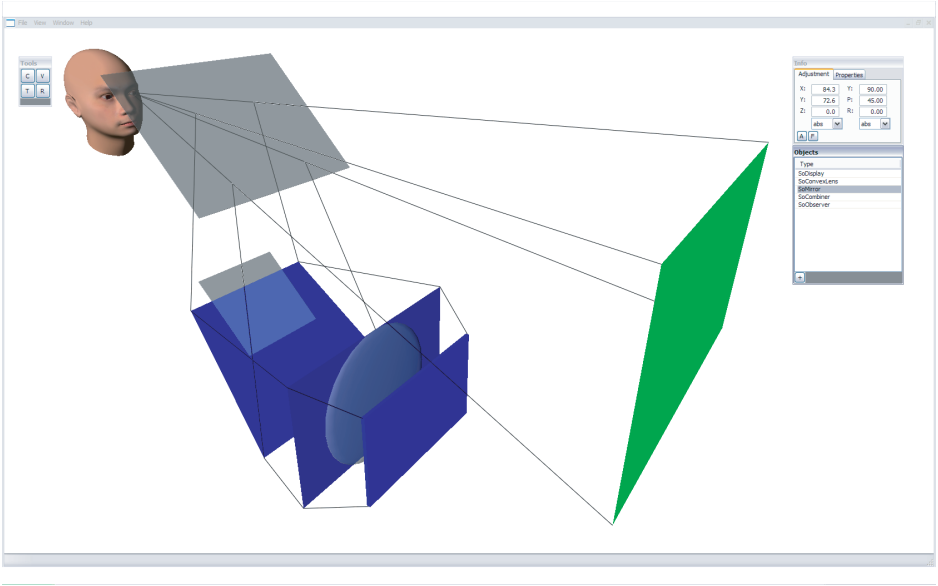


Figure 3.1: Parts of the optical path are outside the geometry of some objects

To realize the sequential arrangement using a fast algorithm based on a linked list of the optical elements, a uniform structure for the optical elements was established that is able to represent all desired objects. This structure allows a high potential for extension of the

range of optical elements. In principle, the internal representation of the optical elements is restricted to the six degrees of freedom that define the position and the orientation of the object plane. For some objects like lenses or curved mirrors, it may be necessary to take the object size into account, depending on the particular algorithm that implements the optical properties and the desired accuracy. But in general, each optical element is of infinite size to facilitate the intuitive manipulation of the scene. Figure 3.1 shows an example of a complete image (green rectangle) being computed, although parts of the optical path are outside the geometry of some of the objects. The blue rectangles symbolize the area where the image hits the particular object plane. In the case of the spherical lens, the corners of the image are outside the object's geometry. At the succeeding object only a small part of the image is located inside the rectangular mirror. This allows a quick proceeding whereas in a second step the size of the object can be easily adopted to the optical path. According to the definition of a linked list, each object only needs to know its immediate successor and predecessor in the optical path. The successor provides the *eye pose*, which is not necessarily the absolute position of the human eye, as it is transformed by reflecting objects (e.g. mirrors), but the position where the eye appears after mirroring on the current object's plane (figure 3.2). By means of the transformed *eye pose*, it is possible to restrict the computation to the rays that actually hit the human eye. From its predecessor each object receives the *image points* that result from looking through the preceding part of the optical system. Depending on the eye pose, the image points and the optical properties, the image points for the current object are computed. To achieve the optical path, also the points where the rays hit the object's surface, in the following termed *object points*, are computed.

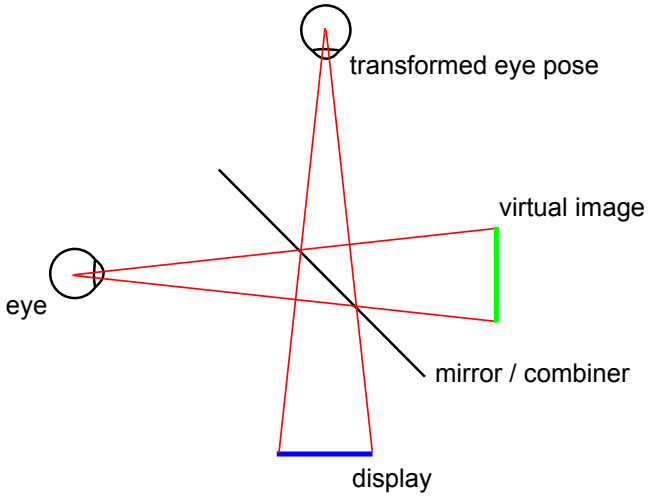


Figure 3.2: Sketch illustrating the transformation of the preceding eye pose

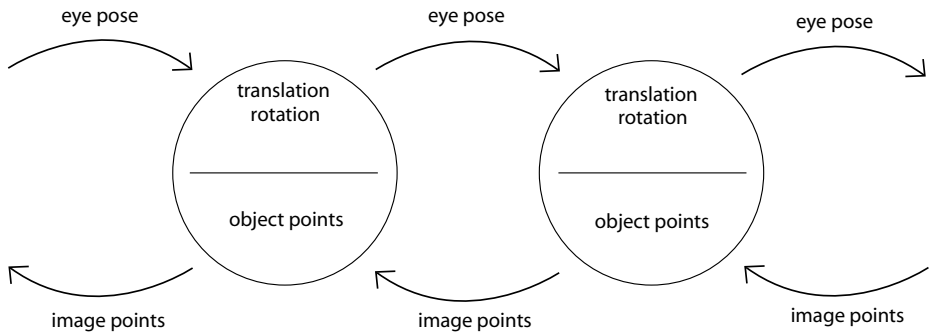


Figure 3.3: Structure of the algorithm and the optical elements

The basic algorithm (figure 5.2) computes the focal plane and the final image points as well as the optical path of the particular rays by passing though the linked list three times. As each pass depends on the results of the previous pass and needs to step through the list into the opposite direction, this is the minimum number of passes. The first pass starts at the human eye, updating the eye pose of each object. In a second pass into the opposite direction, the image points are computed. Thereby, the initial image points for the display result from the object points on the display as they are equal. Depending on the different kinds of optical elements that are used in the arrangement, the accuracy of the representation increases with increasing number of object points. During the last pass, the object points of each element are computed. Based on a simplified representation they result from the intersection points of the rays through the particular image point and the eye pose with the current object surface. A more accurate representation provides an individual implementation of the object points for each optical element. The final image and the focal plane results from the image points of the last instance to the human eye in the optical path. In the majority of cases, this is the combiner, a half-transparent part that effects the fusion of the virtual object with the environment.

4 Optical Elements

Properties and implementation of the plane mirror, the convex lens and the concave mirror

---

In the preceding chapter the basic structure of the generic algorithm and the representation of the optical elements was introduced. In the following, the concrete implementation of some optical elements is described. The optical properties of the particular object are respectively explained in general before the aspects of its implementation are discussed. The chapter is restricted to the plane mirror, the convex lens and the concave mirror as these are the most important elements in respect to the construction of an optical see-though display. Furthermore a comprehensive discussion of every optical element would go beyond the scope of this document.

4.1 Plane Mirror

A flat surface with specular reflection is called a *plane mirror*. Incoming light from a single incoming direction is reflected to a single outgoing direction, with the incoming and outgoing rays drawing the same angle with respect to the surface normal (law of reflection: figure 4.1). In optical systems plane mirrors are used to bend the optical path and to lengthen the distance to the focal plane. The optical properties of the *combiner*, which is a half-transparent part that mostly builds up the last instance in the optical path to the human eye and is used to project the virtual objects to the view on the environment, are very similar to the properties of a mirror.

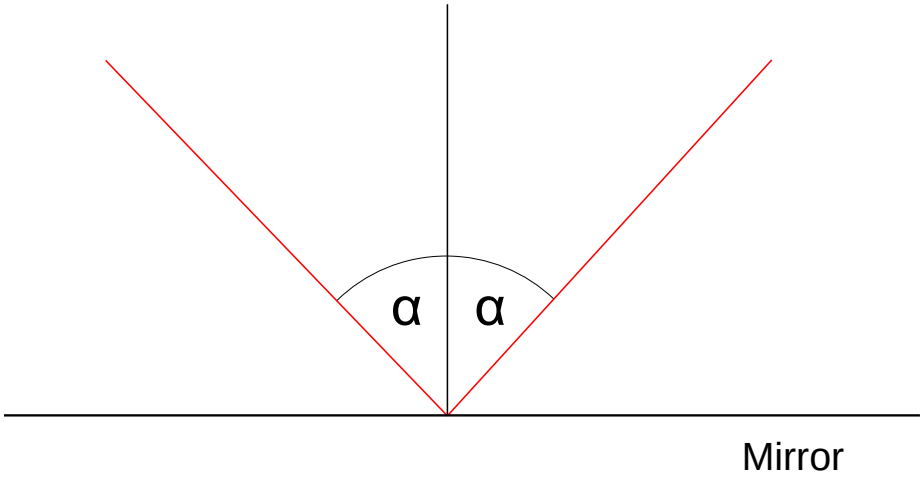


Figure 4.1: Law of reflection

The implementation of the plane mirror in consideration of the outlined approach and structure provides the transformation of the eye pose. This allows the restriction of any further computations to the rays that actually reach the human eye. The eye pose results from the eye pose of the succeeding object mirrored on the object plane of the mirror and therefore represents the position where the human eye appears when looking from the preceding object into the mirror (figure 3.2). Now the rays that hit the mirror surface can be determined by intersecting the line through the transformed eye pose and each image point of the preceding object with the mirror plane. The emerging intersection points make up the object points of the mirror. If an object point lies outside the boundary of the mirror surface, the appropriate image point can be taken out of computation, as the appropriate ray does not arrive at the human eye in compliance with the order of the sequential arrangement of the optical elements. However, it may be reasonable to keep and illustrate these points anyway to simplify the understanding of the relationship between the position, orientation and size of the mirror and the final image. The image points of the mirror result from the image points of the preceding object mirrored on the object plane of the mirror.

Figure 3.2 shows a simple setup with a display and a combiner. The object points on the corners of the display are used to determine the image points of the combiner with the result of the virtual image.

4.2 Convex Lens

Due to its optical properties the *convex lens* is the most important lens for a lot of applications. It can be convexly shaped on either one or on both sides, which means that it is thicker at the center than at the edges. Convex lenses are also called *converging lenses* or *positive lenses*, because incoming rays converge after passing the lens. The *principal axis* is a construction line drawn perpendicular to the lens through its optical center, which is located at the geometric

center of the lens. If the incoming rays are parallel to the principal axis, they intersect in one single point, the *focal point*. Every convex lens has two focal points, one on each side. The distances between the lens and the focal points are called *focal lengths* and are parameters that describe the strength of a lens. The thicker a convex lens is, the stronger is the deflection of the rays and therefore the smaller are the focal lengths of the lens.

The diverging rays of a point light source located farther away than the focal point on the corresponding side of the lens converge and intersect in one single point that can be seen as projection of the source point and is termed *focal point* as well. This point is farther away than the focal point of the lens on that side. As the human eye is not able to deal with converging rays, the observer would not see a sharp image in the range between the lens and the focal point. Behind the focal point, the light source appears to be on the same position as the focal point. Thus an object behind the lens appears nearer to the observer and mirrored, because of the intersection of the rays. With increasing distance of the light source, the intersection point moves nearer to the focal point of the lens. If the point light source is at infinity, the incoming rays are approximately parallel and intersect in the focal point of the lens according to its definition.

In the case of the light source being located in the space between the focal point and the lens near to the focal point of the lens, the rays leave the lens almost parallel on the opposite side. Thus an object behind a convex lens near to its focal point appears from any arbitrary point of view on the principal axis in the same size as from the position of the lens itself. This ”extension” of the rays is the principle of the classic magnifier and plays a decisive role for the construction of optical see-through displays as well. As the rays are approximately parallel, an object on this position appears on infinite distance. With decreasing distance to the lens, the deflection of the rays and for that reason the distance to the focal plane and the image size decrease as well. Therefore the convex lens provides the possibility to increase the image size and the distance to the focal plane.

Depending on the particular design of the lens, various types of differently pronounced aberration may appear. For the construction of an optical see-through display, the *distortion* of the image plays an important role. Distortion is due to the different deflection of parallel rays with varying distance to the optical axis and therefore to the center of the lens. This effect is called *spherical aberration* (figure 4.2). The image can be either cushion-shaped, barrel-shaped or show a combination of both types of image distortion (figure 4.3), depending on the design and alignment of the lens. Especially in the case of a spherical shaped lens, the distortion increases heavily with increasing distance to the center of the lens.

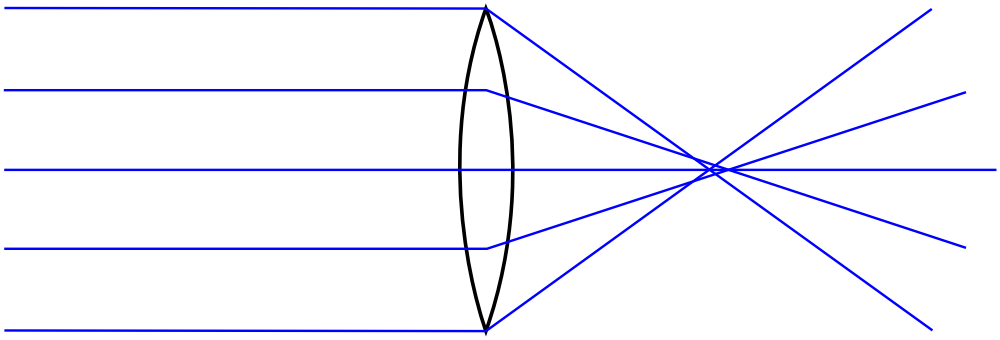


Figure 4.2: Spherical Aberration

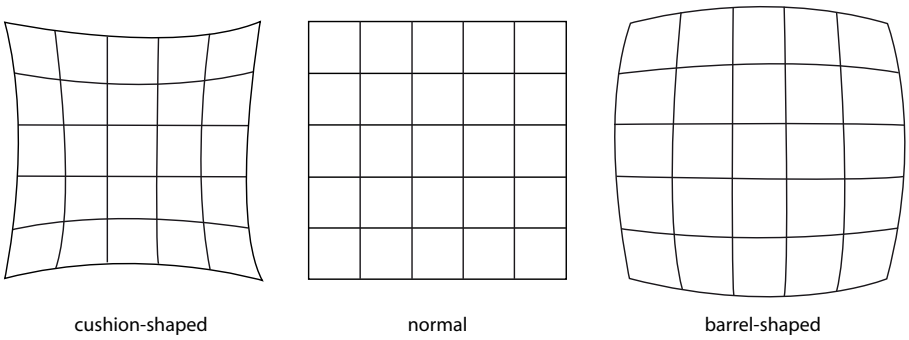


Figure 4.3: Distortion of the image

In the following, three approaches for the implementation of a convex lens are introduced. All of them presume the knowledge of the eye pose. In this case this is not the position the human eye that appears from the view of the preceding objects position but the eye pose of the succeeding object.

4.2.1 Approximation - Thin Lens Formula

The thin lens formula (figure 4.4) provides an approximation of the distance to the image point depending on the distance to the object point and the focal length of the lens. It is assumed that the image point lies on a line through the object point and the center of the lens. Based on the theorem of intersecting lines at the same time the distance to the optical axis and therefore the image size is approximated.

The approximation of the image points of the lens using the thin lens formula provides a fast computation of the final image and the optical path. As it is assumed that object points

on the same plane are again projected to a single plane, the computation can be restricted to the four limiting rays that start at the corners of the display. Furthermore, this approach is independent of the lens size. Thus it benefits an intuitive manipulation of the scene, as the computation of the optical path can be proceeded even if some of the rays do not hit the lens surface. Due to this simplified representation the approximation becomes inaccurate in many cases, and no aberration like distortion is considered. To resolve this disadvantage and to increase the accuracy of the representation, two further approaches were developed that are discussed in the following sections.

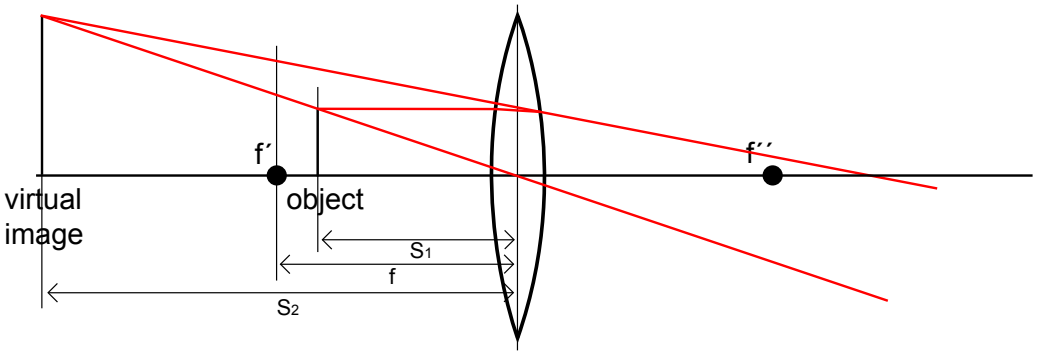


Figure 4.4: Thin lens formula

$$\frac{1}{f} = \frac{1}{S_1} + \frac{1}{S_2}$$

4.2.2 Accurate Computation

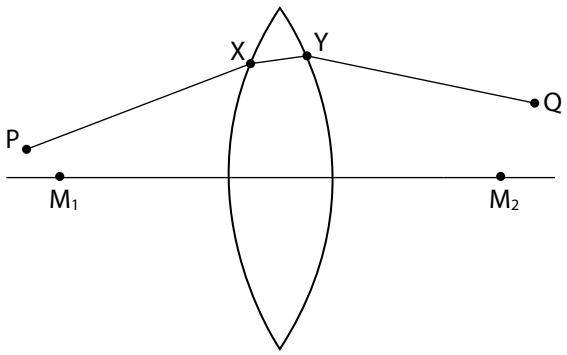


Figure 4.5: Optical Path

The approach of an accurate computation of the image and the optical path is based on the exact determination of the two intersection points X and Y of the particular ray and the lens

surface (figure 4.5). The path taken between two points by a ray of light is the path that can be traversed in the least time (Fermat's principle, figure 4.6). Thus the optical path length (OPL) that results from the sum of the distances d of the specific parts through different materials multiplied with the refractive index n of the particular material must be minimal.

$$OPL = \sum(n_i \cdot d_i)$$

$$OPL(x_1, x_2) = n_1 \cdot |PX(x_1)| + n_2 \cdot |X(x_1)Y(x_2)| + n_1 \cdot |Y(x_2)Q|$$

The refractive index is a measure for how much the speed of light is reduced inside the medium. The expression of the optical path lengths in relation to one or more unknowns allows the exact determination of the intersection points on the lens surface by building the derivatives in respect to the particular unknown. As in the case of a non-planar lens surface at least two unknowns are necessary to express the optical path length, this inevitably leads to a complex non-linear system of equations. The enormous complexity of the equations results in a very expensive computation and often no unique solution exists. Thus this approach is not very practicable.

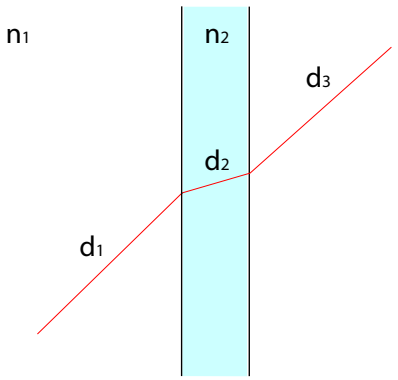


Figure 4.6: Fermat's principle

4.2.3 Approximation - Ray Tracing

This section describes an approach for the approximation of the optical properties of the convex lens and therefore for the deflection of the rays, that was developed within the scope of this project. It constitutes a combination of the approximation via the thin lens formula and concrete ray tracing.

Initially a first approximation for the image point Q' (figure 4.7) is computed applying the thin lens formula to the particular image point Q of the preceding object in the optical path. Thereby the eye pose is split into two points P1 and P2 on the lens of the human eye. All further computations are accomplished twice, once for each of them. Based on the new image point, the optical path of the ray is determined backwards. The result is an intersection

point on the plane through the original image point of the preceding object that is perpendicular to the optical axis. Therefore the point on the lens surface is determined where a ray would have to leave the lens in order to produce the approximated image point. This is the intersection point of a line through the appropriate eye pose and the approximated image point and the lens surface. The optical path is computed using the law of refraction (figure 4.8) depending on the appropriate normal vector at the intersection point and the refractive indices.

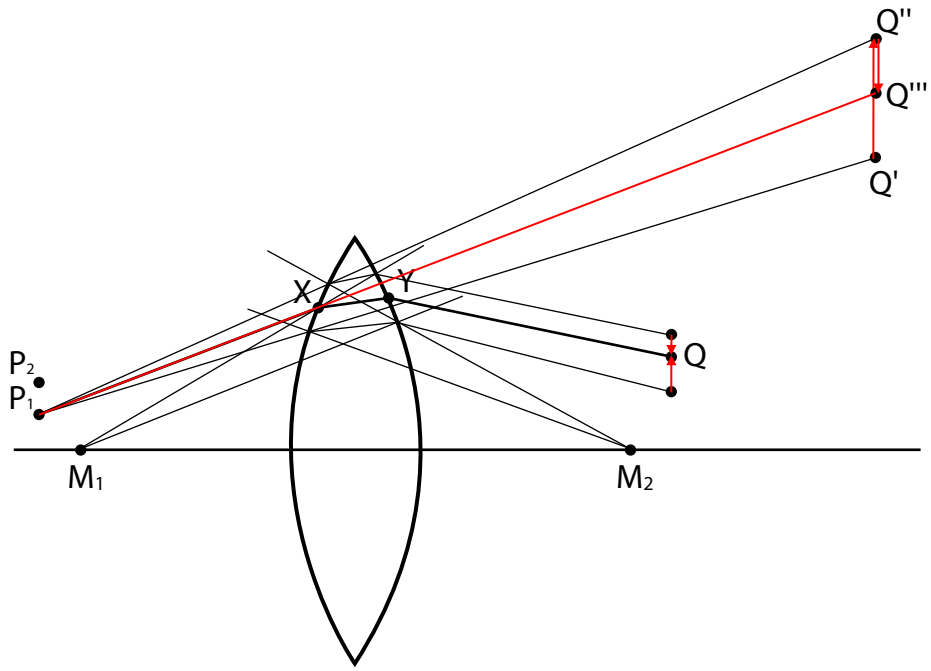


Figure 4.7: Combined approximation algorithm

The vector that results from the difference between the original image point and the computed intersection point on the same plane represents the failure of the first approximation. The inverse vector to the failure vector is multiplied by the factor known from the thin lens formula and added to the image point with the result of a new approximation. This calculation is repeated until the length of the failure vector is smaller than a desired accuracy or a maximum of cycles is reached. The final image point B results from the intersection point of the two lines through the particular eye pose and the appropriate lastly approximated image point Q‴.

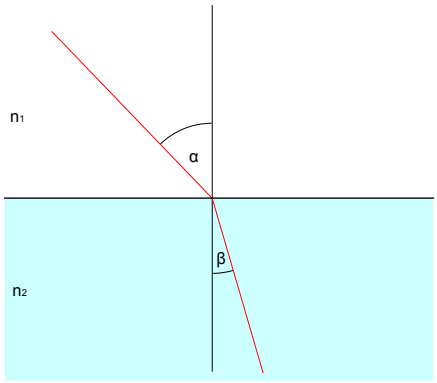


Figure 4.8: Law of refraction

$$\frac{sin(\alpha)}{sin(\beta)} = \frac{n_2}{n_1}$$

4.3 Concave Mirror

With regard to its optical properties, the concave mirror is quite similar to the convex lens, but there exists only one focal point on the same side of the mirror from where the light comes and is reflected to. This focal point is again the intersection point of incoming rays that are parallel to the principal axis. Diverging rays from a point light source farther away than the focal point converge after being reflected and intersect, analogously to the convex lens, in a focal point located farther away than the focal point of the mirror. For the light source being located at the focal point of the mirror, the rays are reflected parallel to the principal axis. Within the space between the focal point and the mirror, the rays still diverge after being reflected, with the deflection decreasing proportionally to the distance to the mirror.

The implementation of the concave mirror provides the translation of the eye pose. The eye pose of the succeeding object in the optical path is mirrored on the object plane that is perpendicular to the optical axis and includes the center of the concave mirror. The image points and the object points are either approximated using the thin lens formula or the combined approach that was introduced in the previous section.

In the first case (figure 4.9), the image points Q of the preceding object are mirrored on the object plane (Q') and the thin lens formula is used to determine the image points Q'' of the mirror. In the second case, analogously to the convex lens these approximated image points are used as initial points for the first cycle of the algorithm. Thereby the eye pose is again divided into two points on the lens of the human eye. The failure is determined by tracing the ray though the eye pose of the succeeding object and the intersection point on the mirror that results from the line through the eye pose and the particular approximated image point. The deflection of the ray at this intersection point is determined using the law of reflection

(figure 4.1). The failure vector results from the difference between the particular image point of the preceding object and the intersection point that results from the reflected ray and the plane through this image point that is perpendicular to the line through the image point and the center of the mirror. The inverted failure vector is mirrored on the object plane of the mirror and added to the first approximated image point. The result is used as starting point in the next cycle of the approximation. When the desired accuracy or a maximum number of cycles is reached, the final image point again results from the intersection point of the two lines through the particular eye pose and the appropriate lastly approximated image point.

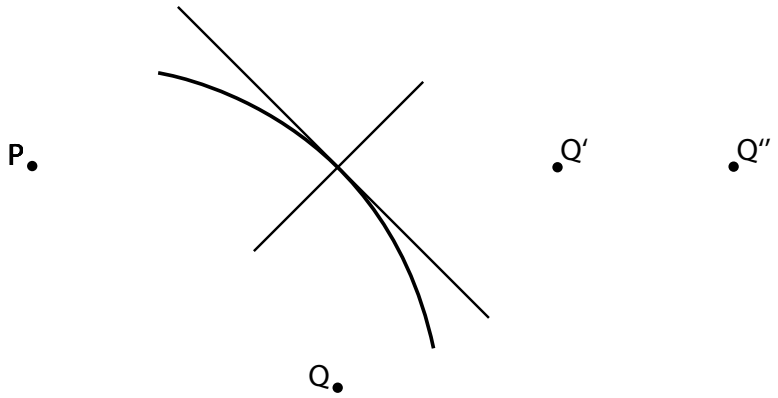


Figure 4.9: Concave Mirror

5 Implementation

Important implementation issues and class hierarchy

The implementation of the construction tool is realized as an extension of the OpenGL based Coin3D graphic library. This library uses the SGI Open Inventor specification ([19], [20]) and provides a 3D scene database, a set of node kits and a set of draggers and manipulators. The 3D scene database stores one or more scene graphs that are implemented as ordered collections of nodes. Each node holds a specific information such as materials, shapes, transformations, lights or cameras. Node kit nodes are used to structure nodes in the scene graph and to define application-specific objects and semantics. Draggers and Manipulators are special nodes that provide the manipulation of the scene graph by letting the user interact with elements in 3D. In addition, the Coin3D graphic library contains a component library that offers viewers to represent and manipulate the scene graph in a system window.

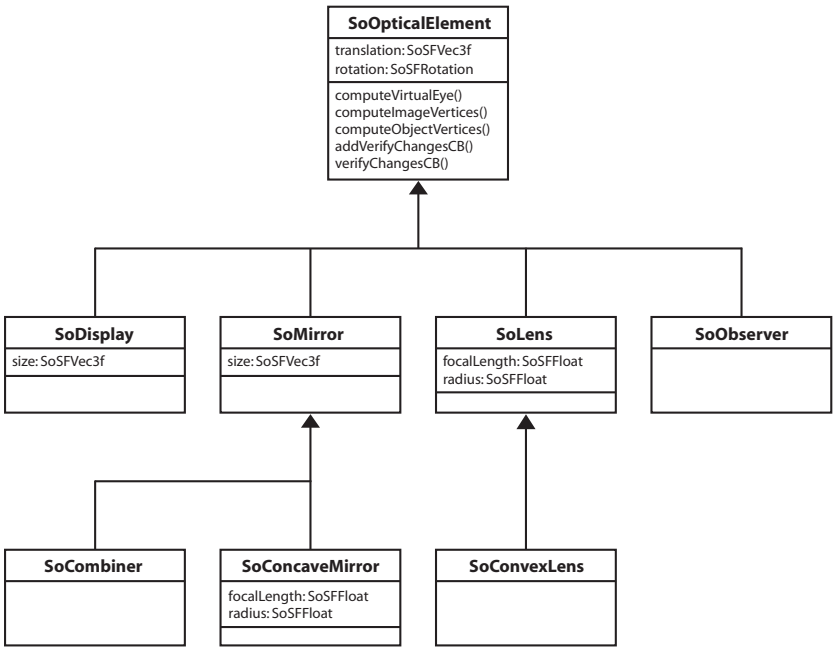


Figure 5.1: Class hierarchy of the optical element classes

Figure 5.1 shows the class hierarchy of the optical element classes in the construction tool implementation. The *SoOpticalElement* class is the superclass from which all classes that implement a specific optical element are derived. It extends the SoBaseKit class of the Coin3D library that provides a structured arrangement of preassigned nodes. The basic structure of the nodes that build the optical element is illustrated in figure 5.2. The *top separator* is the parent node of the optical element and is attached to the scene graph. The *dragger switch* allows to show either the translation dragger, the rotation dragger or none of both. Although the SoOpticalElement class contains two standard draggers that provide the translation, rotation and scaling of the optical element in all three dimensions, it may be desirable to override these with ones that implement a specific behavior depending on the particular optical element. The *object* node which holds the object's geometry as well as the *image* node, which is the part where the optical path hits the object plane, are respectively grouped with a translation, a rotation and a scale node that determine their position, orientation and size in the scene graph.

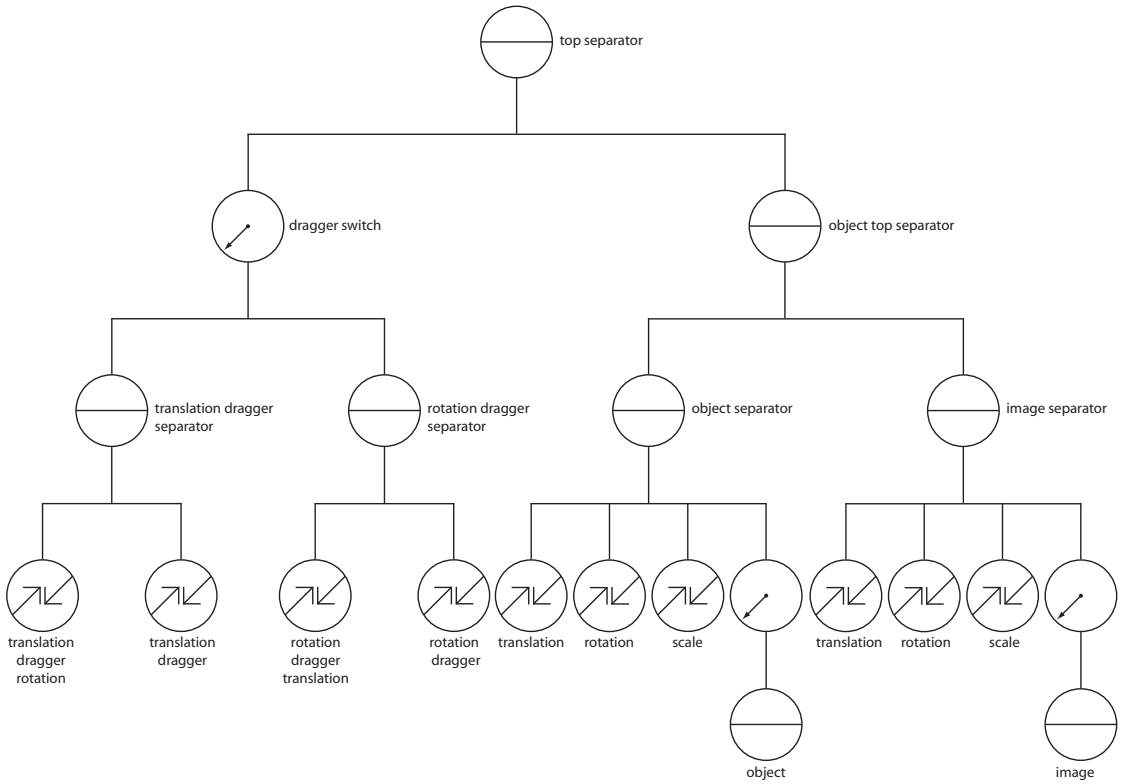


Figure 5.2: Node structure of optical element

The SoOpticalElement class provides three virtual methods that are used during the three passes through the linked list of the optical elements. In the first pass the eye pose for each element is computed by the *computeVirtualEye()* function. If the particular element has a re-

fractive surface, the eye pose results from the eye pose of the successor mirrored on the object plane. In all other cases the eye pose is equal to the eye pose of the succeeding object. In the second pass the *computeImageVertices()* function computes the image points of the particular object. Finally, the object points are determined by a call to the *computeObjectVertices()* function during the third pass. The *verifyChangesCB()* function can be attached to a dragger to check for each manipulation whether it is valid and can be applied or leads to an undesired arrangement and needs to be discarded. If a verifyChangesCB()-function is attached to the optical element itself by the use of the *addVerifyChangesCB()* function, it is called after the verifyChangesCB() function is finished. The values for the *translation* and the *rotation* of each optical element as well as the *size* of all scalable objects and the *focal length* and the *radius* of the SoConcaveMirror and the SoLens are stored to child classes of the SoSFField class, that are part of the Coin3D library. Thus, the values can easily be saved/loaded to/from an text file using the appropriate mechanisms provided by the Coin3D library.

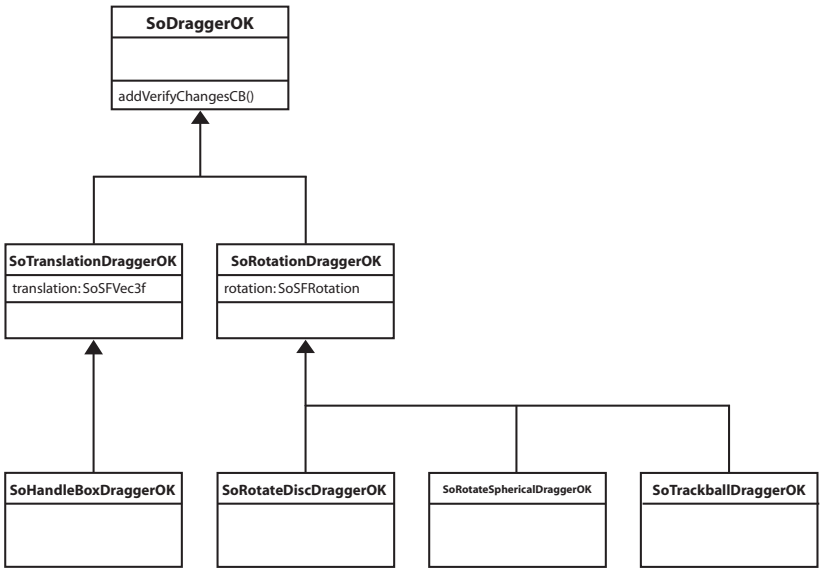


Figure 5.3: Class hierarchy of the dragger classes

Figure 5.3 shows the class hierarchy of the dragger classes. The *SoDraggerOK* class is the superclass of all further dragger classes and extends the SoDragger class, which is the base class of all draggers in the Coin3D library. To avoid manipulation that leads to a misconfiguration of the optical arrangement, the *addVerifyChangesCB()* function can be used to add a callback function to the dragger. This function is called at each sub-step of a manipulation. Depending on the return value, the transformation is applied to the dragger or is discarded. That way you make sure that the draggers geometry stays correctly aligned to the optical element it is attached to.

The basic algorithm is implemented in the *optical system* class. It is derived from the Coin3D SoSeparator class, that is used to group an arbitrary amount of child nodes. If

an optical element is added to the optical system with the *addChild()* function, the verifyChangesCB() function of the optical system is attached to the element. Thus, the validation of a manipulation can be checked for each child of the optical system before it is finally applied. Furthermore, the optical system class provides several functions that respectively return a pointer to a SoSeparator object that holds additional nodes to the optical elements' geometries and their images. These objects are updated with each manipulation to the scene graph. The *getRaySeparator()* function returns the four limitative rays that start at the corners of the display and reach the center of the human eye after passing all objects in the arrangement. The ray through the optical center of each optical element can be accessed using the *computeOpticalAxisSeparator()* function. Finally, the image that is produced by the combiner and the four rays to its corners are provided by the *computeImageSeparator()* and the *computeImageRaySeparator()* function.

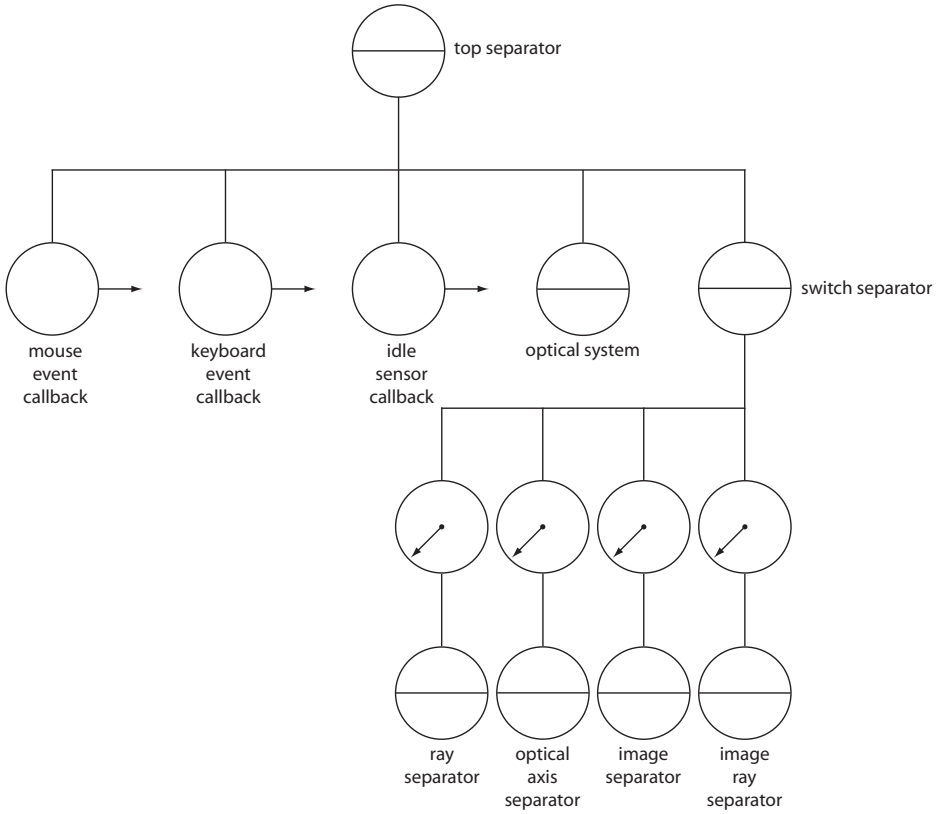


Figure 5.4: Node structure of the optical kit

The *SoOpticalKit* class, that again is derived from the SoBaseKit class of the Coin3D library, implements the interactive functionality and the adaption of the resolution to the processor load. Figure 5.4 shows the node structure of the optical kit. Beside the optical system node, it contains several other nodes. The *mouse event callback* node and the *keyboard event callback* node handle the mouse and keyboard events to provide selection of objects and their drag-

gers and the manipulation of the arrangement. While the user is interactively manipulating the arrangement, the number of rays to be traced through the optical system is reduced to provide real-time computation. When the manipulation is finished, the *idle sensor callback* node causes an increase of the number of rays when no other actions are applied to the scene graph. Thus an adaption of the resolution to the processor load is achieved. The *switch separator* holds several switches to respectively show or hide the rays, the optical axis and the final image.

For the implementation of the GUI the wxWidgets library ([21]) is used. wxWidgets is a cross-platform GUI toolkit that allows writing GUI applications on multiple platforms (e.g. Windows NT/2K/XP, Linux with X11, MacOS). For this purpose the examiner viewer of the Coin3D component library is integrated into a wxWidgets child window. Mouse or keyboard events are passed through to the examiner viewer if the mouse is inside the area of the appropriate window and the windows is activated.

6 The Construction Tool

Layout and basic functionality of the construction tool

This chapter gives an overview of the layout and the basic functionality of the construction tool (figure 6.1). The main window contains three child windows: the tool window, the information window and the object list window. Furthermore, there is one construction window for each opened construction that shows the 3-D model of the appropriate optical setup.

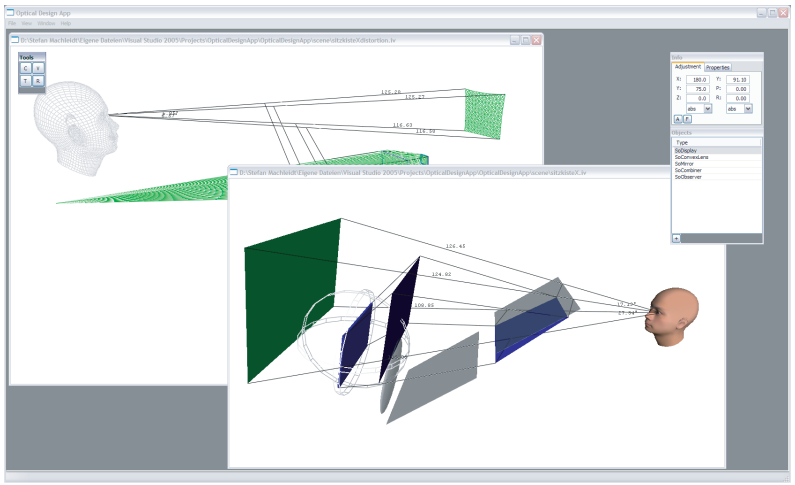
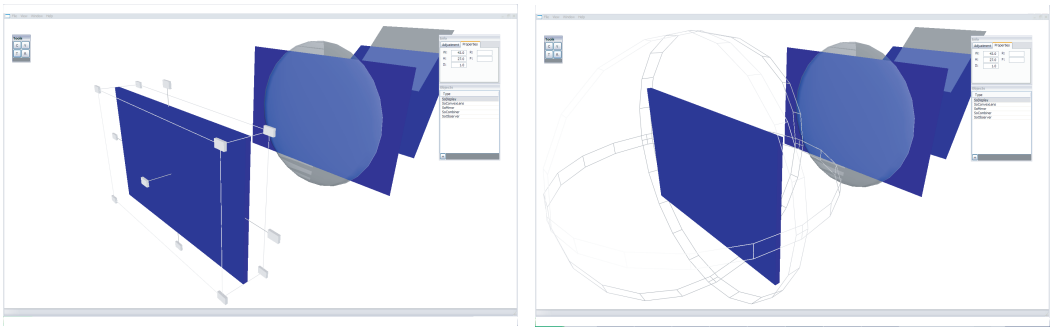


Figure 6.1: The construction tool

The *tool window* is composed of four buttons (C,T,R, and V) that control the functionality of the mouse. In the *cursor mode* (C), objects can be selected either by clicking directly into the 3-D model in the construction window or selecting them in the object list window. The selected object is highlighted in the object list window and the appropriate information about the position, orientation, size and the parameters that control the optical properties of the object are shown in the information window and can be edited. In the *translation mode* (T) and the *rotation mode* (R), the particular objects can be selected as well, but in addition they are surrounded by a so-called *manipulator* that provides the possibility to either move or rotate the selected object by clicking and moving the mouse over the manipulator. The *viewing mode* (V) allows to rotate the camera around the scene by clicking and moving somewhere into the



(a) Translator (b) Rotator

Figure 6.2: Manipulators

construction window. If the shift or the control key is pressed and held, the camera can be moved around and with both keys pressed at the same time, it is possible to zoom.

Figure 6.2 shows two examples of manipulators surrounding the display's geometry. The *translator* in figure 6.2(a) is composed of a wireframe box. Each of the six side surfaces allow to move the selected object along the corresponding plane. Using the massive boxes in the corners of the manipulator, the size of the selected object can be changed in all three dimensions. The boxes in the center of the side surfaces change the size along the corresponding local coordinate axis. Figure 6.2(b) shows the *rotator*. It is composed of three cylinders around the local coordinate axes that allow to rotate the selected object around the corresponding axis. Using the remaining surface of the imaginary sphere through the cylinders surrounding the object allows the rotation in all dimensions.

Each construction can be loaded and saved in the file menu. As the implementation of the optical elements is an extension of the Coin3D Graphic Library, the files are written as text files using the SGI Open Inventor specification. This allows to easily create new constructions by writing the particular objects and their properties into a file. Figure 6.3 shows an example of a file that describes a construction with a display, a convex lens, a mirror, a combiner and the observer. The optical elements of the particular construction are stored as nodes in the scene graph. Each optical element node has several attributes that describe the position, orientation and the optical properties. It is required to add all optical elements as children to the *optical system* node that implements the basic algorithm and therefore computes the optical path and the final image. This provides the possibility to illustrate the optical arrangement with an arbitrary inventor viewer. Furthermore, for the use in the construction tool, it is necessary to put an *optical kit* node on top of the optical system node. The optical kit provides all additional functionality, for example the different mouse modes, the controlling of the manipulators or the viewing modes.

The view menu of the construction tool provides several options to change the representation of the optical setup and to choose between the different approximation algorithms. In order to facilitate the identification of distortion, a wireframe representation of the arrangement is provided (figure 6.4). Therefore the intersection points of the rays and the image plane are connected each with its immediate neighbors.

```
#Inventor V2.1 ascii


SoOpticalKit {
     opticalSystem SoOpticalSystem {
          SoDisplay {
               rotation 0 1.0000001 0  1.5707963
               translation 1.8 0.60000002 0
               size 0.05 0.05 0.005
          }
          SoConvexLens {
               rotation 0 1.0000001 0  1.5707963
               translation 1.605 0.60000002 0
                focalLength 0.40000001
          }
          SoMirror {
               translation 1.114 0.77399999 0
               rotation -0.35740682 0.86285621 0.35740682  1.7177714
          }
          SoCombiner {
               rotation -0.35740682 0.86285621 0.35740682  1.7177714
               translation 1 1.12 0
          }
          SoObserver {
               translation 0.40000001 1.26 0
          }
     }
}
```

Figure 6.3: Construction Tool File

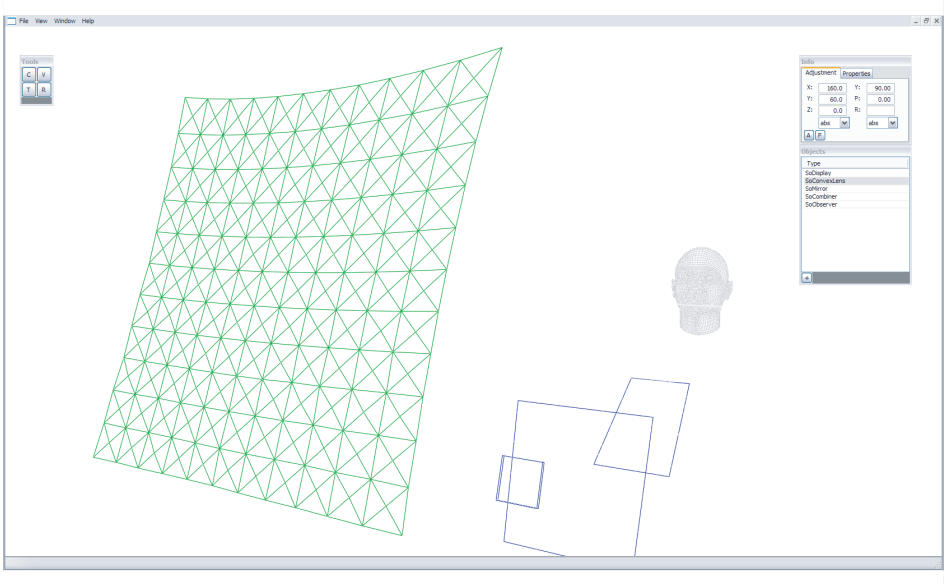


Figure 6.4: Representation of Distortion

7 Constructing a conformal head-up display

Aspects of design of a conformal head-up display

The development of the approaches described in this document and the implementation of the optical construction software was accomplished within the scope of a project with the objective of constructing a conformal, automotive head-up display. The basic principle of the conformal HUD provides the augmentation of the view on the environment with virtual objects that contain a specific position in the environment. The projection of the image is implemented using a half-transparent part, the combiner, in the space between the observer and the environment. Figure 7.1(a) shows the virtual design of the HUD consisting of the blue display, one mirror to bend the optical path, a spherical lens and the combiner. This construction plan was used to build a prototype of the HUD (figure 7.1(b)).

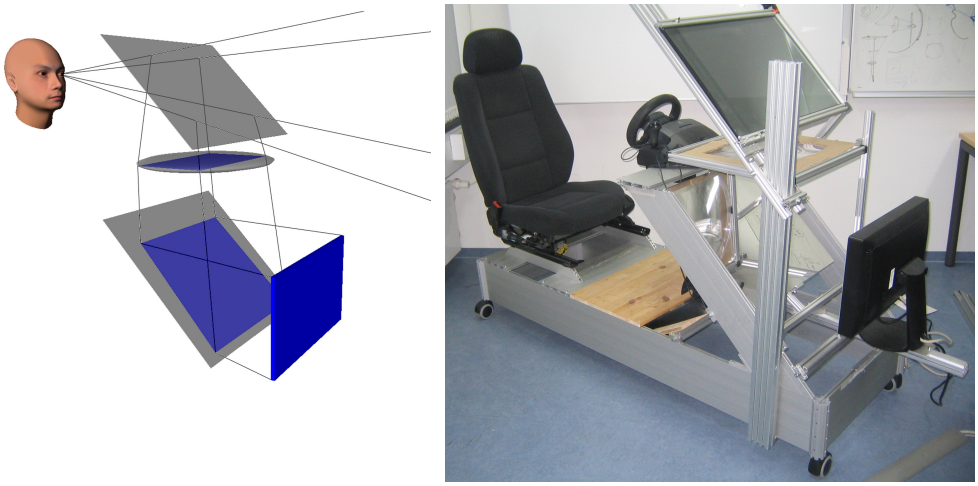
As the focal plane is located perpendicular to the floor in a specific distance to the observer, the positioning of the virtual objects is achieved by an appropriate representation of the object position, orientation and size. If the desired position of the virtual object is not in the same distance as the focal plane, the position of the observer needs to be known to adapt the image. The focal plane must be located in a reasonable distance to the observer with respect to the aspects of human depth perception.

To achieve a desired distance to the focal plane and an appropriate image size in order to cover the requested part of the environment, the deployment of lenses and curved mirrors is necessary. As the image size is restricted to the the size of the lens or the curved mirror, it is recommended to place these object as close as possible to the observer in the optical path.

With the deployment of lenses and curved mirrors aberration increases, especially the distortion of the image (figure 4.3). The problem of distortion can be either solved by a software or a hardware solution. In the first case, the distortion is determined exactly or approximated to calculate the inverse. If the source image is distorted using this inverse, the distortion of the optical system is compensated, and the final image is free from distortion. As the distortion depends on the position of the observer, this solution implies head-tracking.

The hardware solution provides the deployment of aspherically curved lenses or mirrors. Thus the distortion of the optical system can be minimized, but the cost of production increases. It is also conceivable to deploy two lenses with one producing the inverse distortion to the other, according to the software solution. However, this works exclusively for a specific position of the observer.

In some cases, for example the deployment of a conformal HUD in automotive industry with the objective of augmenting the environment with navigation information, it may be desirable to project the virtual objects onto the ground. If the display is tilted compared to the remaining optical system, the focal plane can be aligned to the ground (figure 7.2).



(a) Construction tool (b) Prototype

Figure 7.1: Virtual design and real prototype of a conformal, automotive HUD

The position and especially the distance to the virtual objects can be calibrated to fit to the environment according to the aspects of human depth perception. Apart from distortion, this approach is independent from the position of the observer. A disadvantage is the problem of the representation of perpendicular aligned virtual objects.

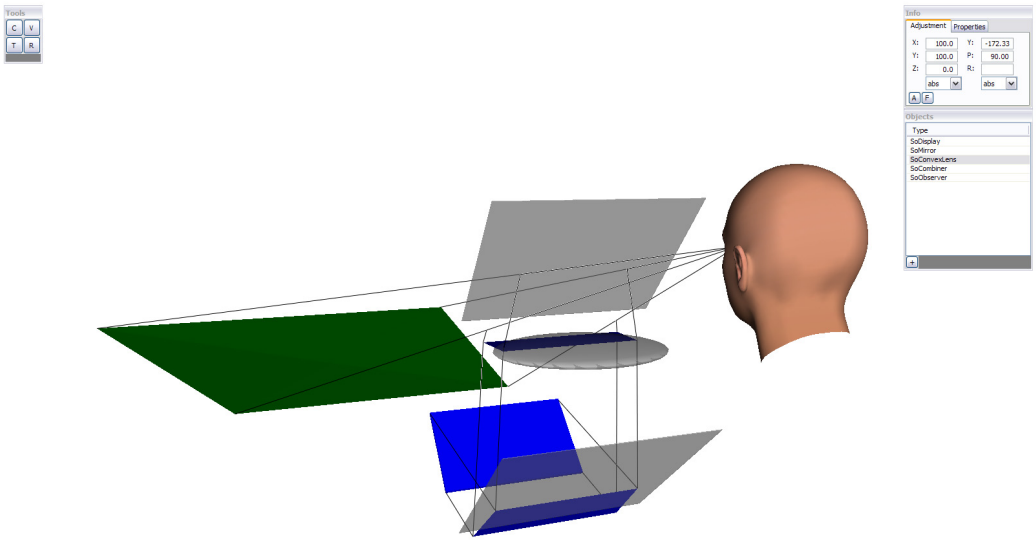


Figure 7.2: Focal plane aligned to the ground

8 Conclusion and Future Work

Summary of the project and outlook to future work

An approach for the implementation of a construction software for optical systems was introduced. The generic algorithm provides an intuitive and rapid development of optical systems as it supports real-time computation of the optical path and the virtual image. The effects and relationships between the optical elements in the arrangement clearly appear. Thus the resulting construction software is not only suited for virtual engineering but for the purpose of teaching as well. The uniform structure of the optical elements supports the extension of the range of optical elements. As the implementation was realized as an extension of the Coin3D Graphic Library, the use in any arbitrary optical application is conceivable. With the different approximation algorithms, the complexity of the computation can be adopted to the requirements of a particular stage in the engineering process and to the desired accuracy. Especially the approach of a very simplified representation ignoring restrictions based on the objects' size provides a quick and innovative proceeding.

The ways of creating and manipulating the optical setup that were introduced are suitable differently for the various stages in the engineering process. Writing the arrangement into a text file facilitates the realization of a concrete setup, for example to simulate and improve an existing construction or to create a starting point for new setups. At the end it provides the export of the final construction. The manipulators are suitable for the early stages in the engineering process. Here the main focus is to find new arrangement by intuitively moving and rotating the objects. As the effect of the manipulation appears promptly, the prevailing dependencies become obvious. To adapt a developed approach to the requirements for construction volume and the available optical elements, editing the values in the GUI turns out to be most effective.

There are a number of open issues that need to be solved in the future. As so far the main focus was to provide a rapid development of optical systems by intuitively manipulating the arrangement, the accuracy of the representation now constitutes an important issue.

To increase the accuracy of the representation, it may be desirable to give up the premise of an exclusively sequential arrangement. Especially if the virtual construction is to be engineered, it is necessary to identify possible occlusion by objects that are not the immediate neighbor in the sequential arrangement. Also the rays that miss one of the objects but hit the human eye anyway should be considered. To obtain more information about the quality of the final image, it seems also reasonable to take aspects of illumination into account.

Another point for future work is to extend the range of optical elements. To reduce aberration, different kinds of aspherical lenses should be implemented. Also aspherical curved mirrors should be considered. Especially in the case of constructing a HUD for cars, where

the individually shaped windscreen is used as a combiner, it is important to be able to reproduce the effect of such surfaces.

Up to now, the optical path to only one of the two human eyes is computed. Especially the construction of head-mounted displays requires the consideration of binocular vision. Here it seems possible to selectively take advantage of the aspects of human stereoscopic depth perception.

Bibliography

[1] O. R. ASSOCIATES, *CODE V*. http://www.opticalres.com/cv/cvprodds_f.html.

[2] O. R. ASSOCIATES, *LightTools*. http://www.opticalres.com/lt/ltprodds_f.html.

[3] J. BEAULIEU, C. GAGNE, and M. PARIZEAU, *Lens system design and re-engineering with evolutionary algorithms*, Genetic and Evolutionary Computations COnference (GECCO), (2002), pp. 155–162.

[4] COIN3D, *Graphic library*. http://www.coin3d.org.

[5] D. FEDER, *Automatic lens design methods*, J. Opt. Soc. Am, 47 (1957), p. 902.

[6] D. GREY, *Aberration theories for semiautomatic lens design by electronic computers. I. Preliminary remarks*, J. Opt. Soc. Am, 53 (1963), pp. 672–676.

[7] D. GREY, *Aberration theories for semiautomatic lens design by electronic computers. II. A specific computer program*, J. Opt. Soc. Am, 53 (1963), pp. 677–680.

[8] E. HAINES, *Essential ray tracing algorithms*, (1989), pp. 33–77.

[9] J. HOWARD, *Optical Design using computer graphics*, Appl. Opt, 40 (2001), pp. 3225–3231.

[10] B. HOWELL and M. WILSON, *Automatic ray-surface intersection method*, Applied Optics, 21 (1982), pp. 2184–2188.

[11] A. KIRKHAM, *The use of computers in optical system design*, Physics in Technology, 13 (1982), pp. 210–216.

[12] I. KRASTEV, *Modern Optical Design and Analysis Software (MODAS)*. http://members.kabsi.at/i.krastev/modas/.

[13] LIGHTTRANS, *VirtualLab*. http://www.lighttrans.com/products_virtuallab.html.

[14] F. LUCY, *Image quality criteria derived from skew traces*, J. Opt. Soc. Am, 46, pp. 46–9.

[15] K. MIYAMOTO, *Image evaluation by spot diagram using a computer*, Appl. Opt, 2 (1963), pp. 1247–1250.

[16] S. ROSEN and C. ELDERT, *Least-squares method for optical correction*, J. Opt. Soc. Am, 44 (1954), pp. 250–252.

[17] G. SPENCER and V. MURTY, *General Ray-Tracing Procedure*, J. Opt. Soc. Am, 52 (1961), pp. 672–678.

[18] D. STURLESI and D. O'SHEA, *Invited paper: Future of global optimization in optical design*, Proceedings of SPIE, 1354 (1991), p. 54.

[19] J. WERNECKE, *The Inventor mentor*, Addison-Wesley Pub. Co.

[20] J. WERNECKE, *The Inventor toolmaker*, Addison-Wesley.

[21] WXWIDGETS, *Cross-Platform GUI Library*. http://www.wxwidgets.org.