Interactive Rendering For Projection-Based Augmented Reality Displays

by

Oliver Bimber
from Daaden

Accepted by the Department of Computer Science
in partial fulfillment of the requirements for the degree of

Doktor-Ingenieurs (Dr.-Ing.)

at the

Technische Universität
(University of Technology) Darmstadt

Supervision of work:	Prof. Dr. h.c. Dr.-Ing. J. L. Encarnação Technische Universität Darmstadt Principal Supervisor
	Prof. Dr. H. Fuchs University of North Carolina at Chapel Hill Second Supervisor
Date of submission:	2nd of September 2002
Date of defence:	15th of October 2002

D17
Darmstädter Dissertation 2002

Interactive Rendering For Projection-Based Augmented Reality Displays

Deutsche Dissertationszusammenfassung

Oliver Bimber

Einleitung

Der Fortschritt im Computer- und Kommunikationstechnologieumfeld verändert auf dramatische Weise alle Aspekte unseres Lebens. Es werden vor allem neuartige 3D Visualisierungen, Ausgabe- und Interaktionstechnologien dazu genutzt, unsere gewohnte physikalische Umwelt mit von Computern generierten Erweiterungen zu ergänzen. Von diesen neuen Interaktions- und Ausgabeparadigmen wird erwartet, dass sie unser Arbeits-, Lern- und Freizeitumfeld sehr viel effizienter und ansprechender gestalten.

Innerhalb verschiedener Anwendungsgebiete werden derzeit Varianten dieser Technologien für die Forschungs- und Entwicklungsarbeit eingesetzt. Die Virtuelle Realität (VR) versucht, dem Benutzer eine gewisse räumliche Präsenz (visuell, akustisch und taktil) innerhalb eines vom Computer erzeugten synthetischen Umfelds zu bieten. Sogenannte Head-Mounted Displays (HMDs) waren viele Jahre lang die traditionellen VR Ausgabegeräte.

Einer der Nachteile heutiger HMDs ist jedoch ihr unausgewogenes Verhältnis zwischen gewichtiger und großer Optik (was qualitativ hochwertige, aber globige und unbequeme Geräte zur Folge hat) und ergonomischen Geräten mit einer schlechten Bildqualität (d.h. niedrige Auflösung, kleines Blickfeld und festem Fokus).

Um einige dieser technologischen und ergonomischen Nachteile zu beheben, und um neue Anwendungsgebiete zu ermöglichen, distanzierten sich die VR Anwender und Entwickler immer mehr von HMDs, und bewegten sich hin zu projektions-basierten Displays, wie etwa immersive Displays, die in der Lage sind, den Benutzer vollständig in synthetische Umgebungen einzuschließen, oder semi-immersive Displays, die in die reale Umgebung eingebettet werden können.

Im Vergleich zu HMDs, haben diese neuen Geräte viele Vorteile (z.B. eine hohe und skalierbare Auflösung, ein großes, erweiterbares Blickfeld, eine bessere Fokussierungsunterstützung der Augen, ein geringeres Auftreten von Unbehagen aufgrund der sogenannten Simulationskrankheit, leichte Gläser, usw.).

Außerdem haben viele dieser Geräte spezielle Eigenschaften (wie Größe und Form), die sich dazu eignen, als Metaphern für applikationsspezifische Funktionalität angewandt zu werden. Manche Variationen lassen sich dadurch leichter in unser tägliches Umfeld integrieren. Ein gutes Beispiel dafür sind semi-immersive Workbenches, deren horizontale Ausgabefläche sich zur Unterstützung einer Tischmetaphor eignet.

Die erweiterte Realität (engl. Augmented Reality – AR) überlagert computergenerierte Grafik auf die Ansicht der realen Welt des Benutzers.

Im Gegensatz zu VR können bei AR virtuelle und reale Objekte gleichzeitig innerhalb des selben dreidimensionalen Raumes koexistieren.

Video-basierte und optische HMDs sind dabei die traditionellen Ausgabetechnologien, und seit Jahrzehnten die Displaygeräte, die überwiegend für AR Applikationen verwendet werden.

Eine Umorientierung von AR Anwendern und Entwicklern auf eine alternative Displaytechnologie (wie es auch im VR Umfeld der Fall war) hat bis jetzt noch nicht stattgefunden. Die meisten der derzeitigen AR Entwicklungen und Systeme haben bisher nur wenige realistische Anwendungen gefunden. Das kann zum Teil auf die eingesetzte Basistechnologie - einschließlich der Ausgabegeräte- zurückgeführt werden.

Genauso wie viele andere Technologien muss AR ausreichend robust, funktionell und flexibel sein, um wirklich Anwendung zu finden, und um nahtlos in unser gut etabliertes Lebensumfeld integriert werden

zu können. Zum Beispiel sind viele unserer Alltagsgeräte danach ausgerichtet worden, spezielle und problemspezifische Aufgaben zu erfüllen. Im Gegensatz dazu versuchen viele AR Anwendungen spezifische Probleme auf einer allgemeinen, im Generellen immer gleich bleibenden technologischen Basis zu lösen.
Deswegen besteht ein gewisser Bedarf an alternativen Displaytechnologien, die die Nachteile der traditionellen Geräte umgehen, und neue Anwendungsfelder für AR schaffen.
Kopfgebundene Displays sind Mitte der sechziger Jahre erstmals zum Einsatz gekommen, und besitzen noch heute das Displaymonopol im AR Umfeld. Im Gegensatz zur Weiterentwicklung der VR Technologie, sind HMDs in den letzten Jahrzehnten nur wenig fortgeschritten, und man kann heute wohl kaum von „ultimativen Displays“ sprechen.
Der in dieser Arbeit vorgestellte projektions-basierte AR (PBAR) Ansatz strebt an, die technologischen und ergonomischen Vorteile der weiterentwickelten und etablierten projektions-basierten VR mit dem Anwendungspotential von Augmented Reality zu vereinen. Dabei sollen neue Anwendungsfelder für AR erschlossen werden. Dieser Ansatz schlägt vor (nach dem Muster der Evolution von VR), die Displaytechnologie vom Benutzer zu trennen und sie anstelle in die Arbeitsumgebung zu integrieren. Allerdings sei erwähnt, dass nicht versucht wird, andere Displaykonzepte (wie z.B. Kopfgebundene Ansätze) völlig zu ersetzen, sondern anwendungsspezifische Alternativen zu bieten.

Definition
Im Generellen wollen wir projektions-basierte Augmented Reality (PBAR) Konfigurationen wie folgt klassifizieren: Ein in den Raum integriertes Projektionsdisplay, dass mit optischen Elementen (in erster Linie halb-transparenten Spiegeln) erweitert wurde, und ein stereoskopisches, Blickpunktabhängiges Betrachten einer grafisch überlagerten realen Szene ermöglicht.
Im Speziellen definieren wir, dass PBAR Konfigurationen folgende Eigenschaften haben:

- Sie vereinen sogenannte optische “see-through” Technologie mit räumlich angeordneten Projektionsdisplays;
- Halb-transparente Spiegel werden in erster Linie als optische Elemente eingesetzt (auch wenn das vorgeschlagene Konzept durch andere optische Elemente erweitert werden kann);
- Sie unterstützen die Anwendung von einfacher oder mehrfacher planarer Optik, oder gekrümmter optischer Elemente;
- Sie verwenden konvexe gekrümmte oder planare Spiegel, um eine virtuelle Abbildung zu erzeugen (auch wenn die vorgeschlagenen Renderingtechniken ebenfalls konkarve Spiegel unterstützen würden);
- Sie unterstützen statische oder flexible Spiegel-Bildschirm Ausrichtungen;
- Sie bieten eine Blickpunkt-unabhängige Bildpräsentation um dynamisch beliebige Perspektiven zu gewährleisten;
- Sie repräsentieren allgemeine, optische, hauptachsenverschobene (off-axis) Systeme (der hauptachsengerichtete (on-axis) Fall ist als Spezialfall eingeschlossen);
- Sie unterstützen einen oder mehrere Benutzer gleichzeitig;
- Sie verwenden verschiedene Rendering- und Bildtransformationsmethoden, die die Bildverzerrung, die durch die verwendete Optik erzeugt wird, aufheben. Diese optischen Effekte umfassen die Reflektion durch Spiegel, die Lichtbrechung, die durch Linsen (oder dicke Glassplatten) hervorgerufen wird, oder die Fehler, die durch schlecht kalibrierte Displays (z.B. Projektoren) entstehen.
- Sie setzen ein interaktives, stereoskopisches Rendering voraus.

Es ist zu berücksichtigen, dass eine Großzahl der artverwandten Systeme, die in dieser Arbeit erläutert werden, einige dieser Eigenschaften mit PBAR Displays gemein haben. Allerdings gibt es keines mit einer nahezu vollständigen Abdeckung.
Diese Eigenschaften beeinflussen allerdings die geräteabhängigen Renderingtechniken, die dann von unserem generellen Ansatz abweichen. Die meisten der diskutierten Systeme könnten als PBAR Variationen betrachtet werden, die auch die vorgestellten Rendering Methoden verwenden könnten.
Wir sprechen von der sogenannten erweiterten Virtuellen Realität (engl. extended Virtual Reality – xVR), wenn eine PBAR Konfiguration eine nahtlose Kombination von VR und AR unterstützt. Dies

wird durch eine konzeptionelle und technologische Erweiterung von traditioneller VR mittels Augmented Reality erzeugt. Wir können sagen, dass xVR einen Spezialfall des PBAR Konzepts darstellt.

Zusammenfassung der Ergebnisse

Im Rahmen dieser Arbeit wird ein projektions-basiertes Konzept der Augmented Reality vorgestellt. Dieses Konzept wird in Form von Proof-of-Concept Prototypen belegt, die die Anwendbarkeit des Konzepts in verschiedenen Anwendungsfeldern aufzeigen. Es werden angemessene Renderingtechniken entwickelt und demonstriert, die die Benutzung solcher projektions-basierten AR Displays auf einer interaktiven Basis ermöglichen.

Die vorgestellten Renderingtechniken für planare und gekrümmte Optik sind flexibel und unabhängig genug, um reibungslos in bereits existierende Softwaresysteme eingegliedert zu werden. Sie sind erweiterbar und konfigurierbar, da sie ein komponenten-basiertes Pipeline Konzept verwenden. Außerdem sind sie allgemein genug gehalten, um unterschiedliche PBAR Konfigurationen unterstützen zu können.

Die beschriebenen Renderingtechniken nutzen die Vorteile der derzeit handelsüblichen Hardware-Beschleunigung so weit wie möglich aus, und bieten interaktive Frame Rates auf preisgünstiger Rendering Hardware, wie z.B. PCs.

Wir können zeigen, dass unser bild-basierender Ansatz für gekrümmte Optiken im Falle der PBAR Konfigurationen wirksamer ist, als adaptierte Variationen von neueren Algorithmen, die auf Geometrie basieren, und die entwickelt wurden, um ein interaktives Rendering von blickpunkt-abhängiger globaler Beleuchtung innerhalb von 3D-Szenen zu unterstützen.

Speziell für Displays mit gekrümmter Optik, die nicht-lineare optische Abweichungen korrigieren, indem sie Mehr-Phasen Rendering und Imagewarping anwenden, haben wir einen neuen Algorithmus eingeführt, der entsprechende regionale Detaillevel erzeugt, anstatt eine uniforme Bildgeometrie während der Laufzeit zu deformieren.

Im Vergleich zu vorhergehenden Ansätzen gewährleistet diese Methode, den Fehler zu berücksichtigen, der durch die stückweise lineare Texturinterpolation hervorgerufen wird, und ihn zu verkleinern, indem die zugrunde liegende Bildgeometrie angepasst wird. Einerseits verhindert der Verfeinerungsalgorithmus ein Überladen der Bildgeometrie und Texturartefakte. Andererseits beschleunigt er das Rendering für solche Displays erheblich, wobei er gleichzeitig einen maximalen Fehler auf der Bildebene garantiert.

Außerdem werden wir beweisen, dass unser neues allgemeines mathematisches Modell für hauptachsenverschobene Ein- und Zwei-Phasen-Lichtbrechung folgende besonderen Fälle mit einschließt: die hauptachsenverschobene Lichtbrechung für zentrierte Systeme, die häufig in Optikliteratur erwähnt wird, und Heckbert's Achsenparallelannäherung der Brechungstransformation, die für Beam-tracing verwendet wird (oder spätere Ansätze, die auf Heckberts Methode basieren).

Die entwickelten Proof-of-Concept Prototypen stellen mögliche Lösungen zu mehreren Problemen dar, die heutigen Projektionsdisplays zugeschrieben werden können, wie etwa das Clipping Problem, das mit semi-immersiven Projektionsflächen verbunden ist, das Verdeckungsproblem bei Rückprojektionsbildschirmen, und die Unterstützung von mehreren Benutzern.

Nachteile, die heutiger kopfgebundener AR Technologie zugeschrieben werden, wie etwa das unausgewogene Verhältnis von gewichtiger Optik (was in globigen und unbequemen Geräten resultiert) und ergonomischen Geräten mit einer niedrigen Bildqualität (d.h. niedrige Auflösung, kleines Blickfeld, und fester Fokus), werden gemindert.

Außerdem werden zusätzliche Nachteile von anderen „unkonventionellen" Augmented Reality Ansätzen, wie z.B. das Verdeckungsproblem und die Einschränkungen bei der Wahl der Displayoberfläche bei Spatially AR, oder das beschränkte Blickfeld, die unflexible Projektor/Bildschirm Anpassung, und die reduzierte Durchsichtqualität von transparenten Projektionsflächen adressiert.

Die wichtigsten Beiträge dieser Arbeit können wie folgt zusammengefasst werden:

- Einführung und Formulierung der Konzepte:
- Das projektions-basierte Augmented Reality (PBAR) Konzept, das vorschlägt, optische Elemente mit heutiger Projektionstechnologie zu verbinden. Es kombiniert die technologischen und ergono-

mischen Vorteile der bewährten projektions-basierten Virtuellen Realität mit den Anwendungspotentialen von Augmented Reality;
- Das Konzept der erweiterten Virtuellen Realität (engl. extended Virtual Reality - xVR) als Sonderfall des PBAR Konzepts. Es erlaubt eine nahtlose Kombination von VR und AR, indem eine konzeptionelle und technische Erweiterung der traditionellen Virtuellen Realität durch Erweiterte Realität angestrebt wird;
- Einführung und Anwendung von neuartigen interaktiven Renderingtechniken, die planare und gekrümmte, bildformende Systeme unterstützen. Im speziellen:
- Auf Geometrie basierende Rendering Methoden für affine bildformende Systeme. Diese Methoden werden voll von handelsüblicher Hardware Beschleunigung unterstützt;
- Mehr-Phasen Rendering Methoden, die auf Bildern basieren für nicht-affine bildformende Systeme. Diese Methoden werden teilweise von handelsüblicher Beschleunigungshardware unterstützt;
- Software Beschleunigungsschemata (wie etwa selektive Verfeinerung, reaktiv-progressives Rendering, paralleles Rendering und Bildcodierung), die bild-basierte Methoden beschleunigt, und sie so für preiswerte Rendering Hardware, wie PCs, verwendbar macht.
- Einführung und Realisierung von neuartigen PBAR Geräten, die die Durchführbarkeit der Renderingtechniken beweisen, mögliche Lösungen für Probleme bieten, die bei bereits existierenden VR/AR Displays auftreten, und neue Anwendungsgebiete eröffnen. Im speziellen:
- Das Reflective Pad und das Transflective Pad als erste stereoskopische, tragbare Displays ihrer Art. Sie bieten eine mögliche Lösung für das Clipping Problem, sowie für das Verdeckungsproblem, die beide mit projektions-basierten VR Systemen verbunden werden.
- Der Extended Virtual Table und das Transflective Board, die eine konzeptionelle und technische Erweiterung der traditionellen Virtuellen Realität mittels Erweiterter Realität unterstützen, und die eine reibungslose Integration solch einer Technologie in alltägliche Arbeitsumfelder ermöglichen. Zudem bieten sie ein großes Blickfeld, verbesserte Fokuseigenschaften, und eine hohe, skalierbare Auflösung;
- Der Virtual Showcase als ein neues interaktives Präsentationsdisplay, das auch die Technologie weitgehend vom Benutzer trennt und sie stärker in unser alltägliches Lebensumfeld integriert. Außerdem bietet es die Möglichkeit, simultan mehrere Betrachter zu unterstützen und einen nahtlosen Rundumblick auf den dargebotenen Inhalt zu gestatten. Diese Eigenschaften sind einzigartig für die heutige Projektionsdisplaytechnologie und für mehr als zwei Betrachter.

Geometrische Optik als Grundlage

In Kapitel 2 der Arbeit haben wir das Wesentliche der geometrischen Optik diskutiert und die mathematischen, physikalischen und physiologischen Grundlagen für die folgenden Techniken und Konzepte gelegt. Unser Ausgangspunkt waren die Gesetze zu Reflektion und Strahlenbrechung von Snellius, die uns zu bildformenden optischen Systemen geführt haben. Spiegel und Linsen – die zwei Hauptkomponenten unserer optischen Systeme – und deren bildformendes Verhalten bei verschiedenen Oberflächentypen wurden detailliert beschrieben und entsprechende geometrische Objekt-Bild Transformationen wurden vorgestellt. Wir haben gesehen, dass nur kartesische Flächen (wie etwa Rotationsparaboloide, Rotationshyperboloide und längliche Ellipsoide) stigmatische Bildpaare erzeugen können. Nur planare Spiegel bieten jedoch wahren Stigmatismus zwischen allen Objekt-Bild Paaren und stellen absolute optische Systeme dar. Die übrigen Oberflächenarten, die Stigmatismus für eine begrenzte Anzahl von Punkten bieten (für gewöhnlich nur für ihre Brennpunkte), sind nur schwer herzustellen. Das ist der Grund, warum sich die meisten optischen Instrumente stigmatischer Bildformation nur annähern, und deshalb kleine Abbildungsfehler hervorrufen. Wir haben gesagt, dass das menschliche Auge selbst ein komplexes optisches System ist. Als letztes Glied einer optischen Kette, können die Augen zwei perspektivisch unterschiedliche Versionen der geformten Bilder erkennen, und Signale an das Gehirn senden, das sie seinerseits zu einem dreidimensionalen Bild vereint. Disparität, Vergenz und Fokus sind die Hauptmechanismen zur Unterstützung von stereoskopischer Betrachtung. Jedoch können, aufgrund der begrenzten Netzhautauflösung des Auges, kleine Abweichungen von nicht-stigmatischen optischen Systemen nicht entdeckt werden. Folglich nehmen die Augen ein einziges konsistentes Bild des jeweiligen Objekts wahr – sogar wenn sich die Lichtstrahlen, die vom

Objekt ausgesendet werden, nicht exakt in einem Bildpunkt kreuzen, nachdem sie durch das optische System gelaufen sind. Zuletzt haben wir gesehen, wie das dreidimensionale Wahrnehmungsvermögen mit Hilfe stereoskopischer Grafikdisplays ausgetrickst werden kann, indem man beiden Augen verschiedene zweidimensionale graphische Bilder präsentiert.
Zusammenfassend kann gesagt werden, dass ein optisches System für unsere Zwecke aus vier Hauptkomponenten besteht: Spiegel, Linsen, Detektoren (in unserem Fall die Augen) und Displays (hier graphisch, stereoskopische). Der Fokus dieser Dissertation ist es, blickpunkt-abhängige, interaktive Renderingtechniken vorzustellen, die stereoskopische, hauptachsenverschobene Projektionsdisplays mit arbiträren optischen, hauptachsenverschobenen Komponenten erweitert. Diese Techniken müssen die physikalischen Reflektions-/Brechungsdeformationen der projizierten Grafiken neutralisieren, so dass das optisch geformte Bild dem Betrachter orthoskopisch, stereoskopisch und perspektivisch korrekt und unverzerrt erscheint.

Vorherige und verwandte Arbeiten
In Kapitel 3 werden frühere und ähnliche Arbeiten diskutiert, die für unser projektions-basiertes AR Konzept und für die Methoden und Techniken, die dafür entwickelt wurden, höchst relevant sind. Diese Arbeiten werden von unseren Ansätzen differenziert. Es werden aber auch Parallelen aufgezeigt.
Zunächst wurde eine Klassifikation der heutigen stereoskopischen Displays aufgestellt, und mehrere Klassen von autostereoskopischen und kopfgebundenen Displays beschrieben.
Im Allgemeinen kann man sehen, dass die meisten autostereoskopischen Displays noch keine optischen see-through Verfahren der Augmented Reality unterstützen. Das ist meistens auf die technologischen Einschränkungen der angewandten Optiken zurückzuführen. Ausnahmen sind einige auf Spiegeln basierende, sogenannte „Re-Imaging-Displays“. Für den Fall dass Bildschirme innerhalb dieser Optiken abgebildet werden, kann man sagen, dass die darauf präsentierten Grafiken jedoch zweidimensional bleiben und keinen autostereoskopischen Effekt erzeugen. Obwohl mittels Video-Mixing ein indirekter „window on the world“ Blick auf das reale Umfeld machbar wäre, werden autostereoskopische Displays kaum für Aufgaben der Augmented Reality genutzt.
Während autostereoskopische Displays keine zusätzlichen Hilfsmittel benötigen, um die meisten visuellen Tiefeneffekte zu adressieren, sind Kopfgebundene Displays stark von solchen Komponenten abhängig, um eine saubere Trennung der präsentierten stereoskopischen Bilder zu gewährleisten. Video see-through und optische see-through Head-Mounted Displays (HMDs) sind die zur Zeit dominierenden AR Display. Diese haben jedoch einige ergonomische und technologische Nachteile. Um diese Nachteile zu beheben, und um neue Anwendungsgebiete zu ermöglichen, orientieren sich die Virtual Reality Anwender und Entwickler immer mehr weg von HMDs, und hin zu auf projektions-basierten, räumlichen Displays, wie etwa surround screen displays (SSDs) und embedded screen displays (ESDs). Im Vergleich zu HMDs, bieten projektions-basierende Geräte einige technologische und ergonomische Vorteile. Aber sie sind nicht mobil genug, sind in der Regel nicht mehrbetrachterfähig (Ausnahmen sind einige zwei-benutzer Ansätze, wie etwa Stanford’s Two-user Responsive Workbench oder UNC’s Two-user Protein Interactive Theatre, und das kürzlich vorgestellte IllusionHole Setup, welches drei Benutzer unterstützt). Außerdem fehlen ihnen optische see-through Fähigkeiten, die eine entsprechende AR Szenerie ermöglichen würden. Head-Mounted Projektor Displays (HMPDs) könnten einen Kompromiss darstellen, der die Vorteile von HMDs mit denen von Projektionsdisplays kombiniert. Sie weisen jedoch auch, ähnlich wie bei HMDs, ein schlechtes Verhältnis zwischen gewichtiger Optik (oder Projektoren) - was zu globigen und unbequemen Geräten führt- und ergonomischen Geräten mit einer niedrigen Bildqualität auf. Dies ist derzeit ein genereller Nachteil aller kopfgebundenen Displays, die von Miniaturdisplayelementen abhängig sind. Projektions-basierte, räumliche Displays, in Kombination mit Video-mixing, unterstützen eine immersivere „window on the world“ Betrachtung. Video-mixing schließt jedoch die Nutzer immer noch vom realen Umfeld aus, und erlaubt nur eine Interaktion von außen.
Im Vergleich zu optischem See-Through, hat Video-mixing auch einige technologische Nachteile, wie es bei Rolland und Azuma beschrieben wurde. Speziell bei projektions-basierten Displaysystemen hindern Probleme, die mit der Video-mixing Technologie zusammenhängen, die Anwendung von interaktiven und flexiblen Ansätzen der Augmented Reality. Probleme hierbei sind z.B. eine zeitverschobene Video-Präsentation (aufgrund der Zeit, die benötigt wird, um Videoströme aufzunehmen und

vorzumischen), eine niedrigere Auflösung des realen Umfelds (aufgrund der begrenzten Auflösung der Kameras), und eine starke Limitierung der Kopfbewegungen (aufgrund von eingeschränkten Bewegungsmöglichkeiten der Kamera).
Speziell im Bereich der Augmented Reality, gibt es eine hohe Nachfrage nach alternativen Displaytechnologien, die die technologischen, ergonomischen und ökonomischen Nachteile traditioneller Geräte ausgleichen und neue Anwendungsgebiete für AR eröffnen. Kopfgebundene Displays wurden erstmals in den 60er Jahren entwickelt, und haben noch heute eine Monopolstellung im AR Umfeld. Im Gegensatz zur VR Technologie wurden sie während der letzten Jahre jedoch kaum weiterentwickelt.
Mittlerweile wurden erste Konzepte der Augmented Reality vorgeschlagen, die die Displaytechnologie vom Benutzer lösen und sie stattdessen ins reale Umfeld eingliedern. Dazu zählen die sogenannte „Spatially Augmented Reality, transparente Projektionsflächen und unser Projektions-basiertes Augmented Reality Konzept. Sie alle ziehen Vorteil aus der heutigen fortgeschrittenen Projektionstechnologie, aber sie unterscheiden sich in der Art und Weise, wie sie reale und virtuelle Umgebungen kombinieren.
PBAR und transparente Projektionsflächen zielen auf eine optische Kombinierung ab, die zusätzliche räumlich ausgerichtete, optische Elemente (entweder halb-transparente Spiegel oder halb-transparente Projektionsflächen) nutzen. Da PBAR halb-transparente Spiegel aus Gründen besserer optischer Eigenschaften und höherer Flexibilität verwendet, wurde auch der Stand der Technik von heutigen Spiegeldisplays diskutiert, um die einzelnen Geräte von unserem Konzept zu differenzieren.
Bei Systemen, die Projektionsflächen in Spiegeln reflektieren, ist eine Transformation der Grafik nötig, bevor sie dargestellt wird. Das garantiert, dass die Grafiken orthoskopisch und nicht gespiegelt oder verzerrt vom Betrachter wahrgenommen werden.
Während diese Transformation für einige der beschriebenen Systeme trivial und statisch ist, da sie von einer festen mechanischen Spiegel-Bildschirm Anordnung und einem eingeschränkten Betrachtungsbereich profitieren, werden für andere Systeme das dargestellten Bilder entweder gar nicht korrigiert, oder es werden zusätzliche optische Elemente verwendet, die eine ungefähre Neutralisation der auftretenden optischen Abweichungen liefern.
Keines diese Systeme unterstützt jedoch eine Vorverzerrung der Grafik, die vom aktuellen und sich dynamisch ändernden Blickpunkt des Betrachters abhängt. Entweder beschränken sie den Betrachter auf einen einzigen Blickpunkt (hot-spot) oder einen kleinen, sehr eingeschränkten Betrachtungsbereich, oder sie akzeptieren optische Verzerrungen, wenn der Betrachter sich bewegt. Systeme, die zusätzliche Optik zur Korrektur dieser Effekte nutzen, sind zentriert und unterstützen dementsprechend keine hauptachsenverschobene Betrachtung.
Da unser PBAR Konzept flexible und nicht-statische Spiegel-Bildschirm Anordnungen und eine von der Blickrichtung abhängige und hauptachsenverschobene Bildpräsentation für einzelne oder mehrere Betrachter und für verschiedene Spiegelkonfigurationen unterstützt, ist eine Entwicklung von effektiveren Rendering- und Bildtransformationstechniken nötig.
Einige Renderingtechniken, die betrachtungsabhängige, globale Beleuchtungseffekte innerhalb graphischer 3D Szenen simulieren, wurden analysiert. Sie repräsentieren eine Basis und den Ausgangspunkt der Entwicklung unserer Rendering- und Transformationsmethoden. Wir haben diese Techniken in pixel-basierte, bild-basierte und geometry-basierte Ansätze kategorisiert, wobei geometry-basierte Ansätze weiter in virtuelle Blickpunkt- oder virtuelle Geometriemethoden unterteilt wurden. Während photorealistisches Rendering (d.h. die pixel-basierten Methoden) präzise optische Effekte simuliert und hochqualitative Bilder erzeugen kann, können alle anderen Ansätze Bilder bei interaktiven Wiederholraten erzeugen. Sie nähern sich den realistischen, optischen Effekten jedoch nur an.
Bild-basierte Methoden beinhalten die optischen Effekte und die Beleuchtungsinformationen innerhalb eines einzelnen oder mehrerer vorberechneter Bild(er), die zur Laufzeit auf die Szenengeometrie angewendet werden. Geometrie-basierte Methoden berechnen die optisch verzerrten Beleuchtungseffekte immer wieder neu.
Speziell interaktive Renderingtechniken, die immer noch eine akzeptable Bildqualität liefern und Techniken, die von kosteneffektiver Beschleunigungshardware unterstützt werden, sind für unser Konzept besonders von Interesse.

Interaktives Rendern

Im Kapitel 4 der Arbeit werden verschiedene interaktive Renderingtechniken vorgestellt, die mit unterschiedlichen PBAR Konfigurationen verwendet werden. Des Weiteren wird eine Übersicht des rechnerischen Aufwandes und der Komplexität dieser Methoden dargestellt.

Weil absolut optische Systeme (z.B. planare Spiegel) affine optische Abbildungen vom Objektraum in dem Bildraum hervorrufen, können affine geometrische Transformationen in traditionelle Transformationspipelines herkömmlicher Grafikhardware integriert werden um diese zu neutralisieren. Affine geometrische Transformationen benötigen deswegen keinen zusätzlichen Rechenaufwand. Deswegen erhöht sich die Renderingzeit der vorgestellten Techniken, die eine solche Optik unterstützen, nur mit der Anzahl der angewandten Renderingdurchgänge (z.B. im Fall von Optik, die aus mehreren Elementen besteht, oder Optik, die mehrere Benutzer unterstützt).

Für optische Elemente, die eine gekrümmte Bildtransformation voraussetzen, ist gezeigt worden, dass ein bild-basierter Ansatz effizienter ist, als ein geometrie-basiertes Verfahren. Der vorgestellte bildbasierte Ansatz umgeht einen direkten Zugriff auf die Szenengeometrie und verhindert somit rechenintensive Transformationen von vielen Szenenpunkten. Zusätzlich ist dieser Algorithmus nicht an eine geometrie-basierte erste Renderphase gebunden, sondern unterstützt jedes bilderzeugendes Verfahren. Der vorgestellte Algorithmus verwendet eine Sequenz von optionalen nicht-affinen Bildtransformationen die wir derzeit als am effizientesten für nicht-stigmatische PBAR Displays halten. Es kann gezeigt werden, dass dieses Verfahren eine Kombination des erweiterten Kamerakonzeptes und projektiver Texturen ist. Projektive Texturen verwenden eine perspektivische Texturmatrix, um die Punkte der Projektionsoberfläche in die Texturkoordinaten der Pixels abzubilden, die auf diese Punkte projezieren. Im Vergleich dazu projeziert unsere Methode Bildpunkte direkt auf die Projektionsoberfläche und lässt dabei die originalen Texturkoordinaten unverändert. Das ist nötig, da gekrümmte Spiegel für jedes Pixel einen individuellen Projektionsursprung voraussetzen. Die Benutzung von individuellen Projektionsparametern ist die fundamentale Idee des erweiterten Kamerakonzeptes – auch wenn dieses ursprünglich für Raytracingverfahren Anwendung findet. Dabei wird der Ursprung der Hauptstrahlen, die durch bestimmte Pixel auf der Bildebene laufen, abhängig von der Position des Pixels gemacht. Das bedeutet, dass die Hauptstrahlen nicht von einem einzigen Punkt ausgestrahlt werden (wie es bei einer perspektivischen Projektion der Fall ist) oder auf einer Ebene liegen (wie es bei einer orthogonalen Projektion der Fall ist). Die modifizierten Strahlen werden allerdings wie gehabt durch die Szene verfolgt und liefern am Ende die Farbwerte der Pixel. Das erzeugte Bild stellt eine verzerrte Projektion dar, die abhängig von der Funktion ist, die die Strahlen modifiziert. Der Hauptunterschied zu unserem Ansatz ist, dass das erweiterte Kamerakonzept ein deformiertes Bild via Raytracing erzeugt (d.h., jedes Pixel wird durch einen modifizierten Hauptstrahl erzeugt). Unsere Methode hingegen deformiert ein existierendes Bild, indem jedes Pixel individuell projeziert wird.

Für Displays die eine Korrektur von nicht-linearer Verzerrungen mit Hilfe von Mehr-Phasen-Verfahren unterstützen, bieten angemessene Level-of-Detail Verfahren (im Gegensatz zur Verwendung einer uniformen Bildgeometrie) die Möglichkeit, den regionalen Fehler der durch eine stückweise, lineare Texturinterpolation entsteht zu berücksichtigen und zu minimieren. Aus diesem Grund haben wir einen adaptiven Algorithmus entwickelt, der es ermöglicht, mit Hilfe einer regionalen Verfeinerung, Bilder in Echtzeit zu deformieren. Dieser Algorithmus kann verwendet werden, um die optische Verzerrung zu neutralisieren oder unverzerrte Bilder auf gekrümmten Oberflächen darzustellen. Der kegelförmige Virtual Showcase diente dazu, diesen Algorithmus näher zu erklären und zu evaluieren. Für diesen Fall wurden auch die displayspezifischen Komponenten des Algorithmus besprochen. Für andere Displaytypen können diese Komponenten ausgetauscht oder adaptiert werden. Im Speziellen wird eine Methode zur Objekt-Bild (Rück-) Reflektion dargestellt, die für zwei-rangige Spiegeloberflächen, wie z.B. Kegel oder Zylinder, optimiert ist. Für andere Spiegeltypen muss diese Methode ersetzt werden. Auf der einen Seite verhindert dieser Algorithmus das Überladen von Bildgeometrie und das Erzeugen von Texturartefakten. Auf der anderen Seite wird damit das Rendering für solche Displays drastisch beschleunigt - unter der Garantie eines maximalen Bildfehlers.

Dieser bild-basierte Ansatz ist flexibel genug, um in existierende Softwarepakete integriert zu werden, und generell genug, um unterschiedliche Hardware zu unterstützen. Zusätzlich werden die Vorteile der derzeit handelsüblichen Hardware-Beschleunigung so weit wie möglich ausgenutzt, und es werden zusätzlich selektive Verfeinerung, progressives und paralleles Rendering unterstützt. Während die

beiden Rendering-Phasen und die Transformationen der Primitiven vollständig auf heutigen Grafikbeschleunigern ausgeführt werden können, werden Zwischenschritte (wie z.B. individuelle Bildpunkttransformationen) nicht von traditionellen Renderingpipelines (z.B. OpenGL) unterstützt. Das bedeutet, dass diese Zwischenschritte derzeit nicht von traditioneller Grafikhardware profitieren. Werden diese Schritte Software implementiert, belastet das die CPU und den Hauptspeicher. Eine Generation von Grafikkarten wird allerdings solche punktindividuellen Operationen unterstützen, die dann auch eine vollständige Hardwarebeschleunigung unserer Verfahren ermöglichen.

Proof of Concept
In Kapitel 5 werden einige PBAR Proof-of-Concept Prototypen vorgestellt, woran die generelle Machbarkeit und die Effizienz der vorgestellten Renderingtechniken gezeigt werden. Außerdem werden verschiedene VR/AR-spezifische Techniken zur Interaktion, Objektregistrierung, Verdeckung, Kollisionserkennung und optische/nicht-optische Vorverzerrung auf unser spezielles Problem adaptiert. Die implementierten Demonstratoren fokussieren auf drei Hauptanwendungsgebiete: engineering, scientific visualization, und cultural heritage.
Jedes der vorgestellten Geräte wird im Detail besprochen. Dabei wird auf folgende Punkte eingegangen:

- Verringerung des Clippingproblems, das mit semi-immersiven Projektionsdisplays verbunden ist (z.B. durch das vorgestellte Reflective Pad);
- Lösung des Verdeckungsproblems, das mit Rückprojektionsdisplays verbunden ist (z.B. durch das Transflective Pad);
- Die Kombination von VR und AR (z.B. durch das xVR Konzept);
- Nahtlose Integration von xVR in Alltagsumgebungen (z.B. durch den Extended Virtual Table);
- Flexible Anwendung von PBAR Konfigurationen (z.B. durch das Transflective Board);
- Die simultane Unterstützung von mehreren Benutzern und eines nahtlosen Rundum-Blickes (z.B. durch den Virtual Showcase).

Im Vergleich zu artverwandten AR Displays konnten die folgenden Beobachtungen gemacht werden:

- Das Sichtfeld des Benutzers, das durch ein Display abgedeckt werden kann (field-of-view) kann entweder konstant, oder interaktiv durch große statische oder kleine tragbare Spiegel vergrößert werden;
- Grafiken können mit einer hohen Auflösung dargestellt werden (höher als es derzeit mit kopfgebundenen Displays der Fall ist). Das kann auf die Verwendung von hochauflösenden Projektoren oder Mehr-Projektorensetups zurückgeführt werden. Zusätzlich wird die reflektierte Grafik durch die Benutzung von konvexen Spiegeln innerhalb eines kleinen Bildbereichs in eine hohe Dichte von Pixel komprimiert. Dies liefert eine hohe räumliche Auflösung innerhalb dieses Bereiches;
- Die Auflösung wird nicht durch die Optik eingeschränkt (im Gegensatz zu holographischem Film, der für transparente Projektionsscheiben verwendet wird);
- Bedingt durch den optischen see-through Ansatz des PBAR Konzeptes kann die reale Umgebung in der vollen Auflösung wahrgenommen werden, die vom menschlichen Auge unterstützt wird. Bei Video see-through Applikationen ist dies durch die Auflösung der verwendeten Videokameras begrenzt;
- Die see-through Metapher wird unterstützt – und nicht das sogenannte indirekte Sehen (remote viewing);
- Einige der PBAR Setups unterstützen eine direkte Interaktion (z.B. das Transflective Pad und das Transflective Board), währenddessen andere nur eine indirekte Interaktion ermöglichen (z.B. der Extended Virtual Table und der Virtual Showcase);
- Pseudo-reale Abbildungen können durch konvexe oder planare Spiegel generiert werden. Dies ermöglicht dann eine direkte Interaktion;
- Leichte Shutterbrillen bieten ein verbessertes ergonomisches Verhalten (im Gegensatz zu kopfgebundenen Displays);

- Das Fokusproblem, das mit kopfgebundenen Displays verbunden wird, kann verbessert werden. Das liegt daran, dass die reflektierte Bildebene im Abbildungsraum der Spiegel räumlich besser dargestellt und ausgerichtet werden kann.
- Wie auch für andere räumliche Displays bieten PBAR Displays weniger Anlass zur sogenannten „simulator sickness“. Das liegt an der räumlich angeordneten Bildebene (im Gegensatz zu kopfgebundenen Displays, bei denen die Bildebene auch kopfgebunden ist);
- Es werden keine Schatten von physikalischen Objekten oder von interagierenden Benutzern geworfen. Dies liegt an der Verwendung von Rückprojektionssystemen (im Gegensatz zu Spatially AR oder head-mounted projective displays, die eine Frontprojektion verwenden);
- Das Erscheinen von virtuellen Objekten wird nicht durch die reale Umgebung eingeschränkt (im Gegensatz zu Spatially AR);
- Mehrbenutzeranwendungen und nahtloser Rundumblick sind möglich;
- Das Blickfeld wird durch die verwendete Spiegeloptik weniger eingeschränkt (im Gegensatz zu dem holographischen Film der für transparente Projektionsbildschirme verwendet wird);
- Eine flexible Ausrichtung und Konstellation von Projektoren, Projektionsdisplays, und Spiegeln ist gegeben (im Gegensatz zu transparenten Projektionsbildschirmen);
- Eine verbesserte Durchsichtqualität ist durch die halb-transparenten Spiegel gegeben (im Gegensatz zu dem holographischen Film, der für transparente Projektionsbildschirme verwendet wird);
- Die verwendeten optischen Bildkombinierer können keine korrekten Verdeckungen zwischen realen und virtuellen Objekten darstellen. Bedingt durch die optische Charakteristik der halb-transparenten Spiegel, erscheinen virtuelle Objekte immer als halb-transparent – anstelle dahinter liegende reale Objekte zu verdecken. Deswegen lassen helle, reale Oberflächen (die einen hohen Anteil des Umgebungslichtes reflektieren) dunkle, virtuelle Objekte (die eine geringe Leuchtkraft besitzen), die sie überlagern, optisch verschwinden.

Der letzte Punkt wird (zumindest für kopfgebundene Systeme) von Kiyokawa et al mit einem erweiterten HMD namens ELMO adressiert. ELMO verwendet auch halb-transparente Spiegel als optische Bildkombinierer. Allerdings wird diese Optik durch ein halb-transparentes LCD Display erweitert. Mit Hilfe des LCD Displays lassen sich bestimmte Pixel so schalten, dass sie entweder Licht durchlassen, oder abblocken. Neben den generellen Nachteilen von kopfgebundenen Displays muss ELMO einige zusätzliche Probleme lösen: die geringe Leuchtstärke des LCD Displays, und das hohe Nachleuchten und die niedrige Auflösung des LCD Displays. Allerdings wird durch ELMO als erstes funktionierendes System seines Typs das Verdeckungsproblem von optischen see-through Displays erstmals effektiv gelöst. Die generelle Idee könnte für zukünftige Verbesserungen unsers PBAR Konzept von Interesse sein.
Offensichtlich ist das Interaktionspotential für die unterschiedlichen PBAR Prototypen ein Klassifikationskriterium: Während einige Systeme selber Interaktionswerkzeuge darstellen (z.B. das Reflective Pad und das Transflective Pad), werden andere als passive Ausgabegeräte verwendet, die eine Interaktion durch zusätzliche Tools ermöglichen. Einige Prototypen unterstützen eine direkte Interaktion mit der augmentierten, realen Umgebung (z.B. das Transflective Pad und das Transflective Board), und andere unterstützen nur eine indirekte Interaktion (z.B. der Virtual Showcase). Wieder andere Prototypen bieten eine simultane Interaktion von mehreren Benutzern (z.B. der Virtual Showcase). Deswegen können wir schlussfolgern, dass nicht nur die Renderingtechniken, sondern auch die Interaktionsformen stark von der verwendeten Optik beeinflusst und eingeschränkt werden. Dies beeinflusst wiederum die Anwendbarkeit und das Anwendungsfeld von PBAR Konfigurationen.

Evaluierung
In Kapitel 6 der Arbeit präsentieren wir Auswertungen von den vorgestellten Renderingtechniken und Hardwareprototypen. Die Ergebnisse dieses Kapitels sind im Folgenden zusammengefasst:
Wir haben die Präzision unserer analytischen Lichtbrechungstransformationsannäherung mit einer präzisen numerischen Methode verglichen und den Schluss gezogen, dass der mittlere Fehler zwischen der präzisen numerischen Brechungsmethode und der analytischen Annäherung weit unter der durchschnittlichen Positionsgenauigkeit der angewendeten elektromagnetischen Trackinggeräte liegt. Somit wird durch den unakkuraten Kopfsensor ein größerer Fehler verursacht, als durch Anwendung der ana-

lytischen Lichtbrechungsannäherung. Wenn jedoch der Fehler, der durch die Lichtbrechung entsteht, gar nicht korrigiert wird, ist die resultierende optische Verzerrung größer als die, die durch ungenaues Tracking verursacht wird. Außerdem benötigt die analytische Annäherung nur einen Bruchteil der Transformationszeit, die von der numerischen Minimierung benötigt wird.
Der Rechenaufwand und die Skalierbarkeit aller vorgestellten Rendering- und Transformationstechniken wurden theoretisch besprochen. Zusätzlich haben wir konkrete Zeitmessungen für die nicht-affinen Transformationsalgorithmen dargelegt, und haben experimentelle Ergebnisse der selektiven Verfeinerungs- und der progressiven Renderingbeschleunigung geliefert.
Der Vergleich zwischen dem geometrie-basierten und dem bild-basierten Ansatz zeigte, dass bei der nicht-affinen Optik (also den gekrümmten Spiegeln) die bild-basierte Methode bessere Ergebnisse liefert als die geometrie-basierte Methode. Dies kann darauf zurückgeführt werden, dass der Großteil (~90-95%) der Renderingzeit des geometrie-basierten Ansatzes für die expliziten Punktberechnungen (d.h. für Transformation und Beleuchtung) benötigt wurde, und dass die Szenen geometrisch hoch aufgelöst sein mussten, um eine saubere gekrümmte Transformation zu unterstützen.
Die vorgestellten Messungen des bild-basierten Ansatzes haben angedeutet, dass hauptsächlich zwei Parameter modifiziert werden können, um die Renderinggeschwindigkeit zu beeinflussen: die Bildgeometrie- und die Bildauflösung.
Der selektive Verfeinerungsalgorithmus, der in Kombination mit unserer Mehr-Phasen-Rendering genutzt wurde, generiert eine Bildgeometrie mit einer regional angepassten Gitterauflösung während der Laufzeit. Zum einen verhindert der Verfeinerungsalgorithmus ein Überladen der Bildgeometrie und Texturartefakte, zum anderen beschleunigt er das Rendering und Bildwarping bei diesen Displays bedeutend, und garantiert gleichzeitig einen maximalen Fehler auf der optischen Bildebene.
Wird der selektive Verfeinerungsalgorithmus angewendet anstatt die gesamte Bildgeometrie zu transformieren, haben unsere Experimente gezeigt, dass die Beschleunigung proportional zu den Präzisionsanforderungen (Darstellungsgenauigkeit) anwächst. Das bedeutet, dass bei den dargestellten Beispielen Beschleunigungsfaktoren von bis zu 6 für eine benötigte Präzision von 0.1mm auf der Bildebene erreicht werden können.
Die Zeitmessungen, die in diesem Kapitel vorgestellt wurden, sind auf unterschiedlichen, sich im Laufe der Zeit verbessernden Hardwareplattformen durchgeführt worden. Dementsprechend sind die Messwerte der einzelnen Versuche nicht untereinander vergleichbar. Um allerdings einen Anhaltspunkt für Schlussfolgerungen zu bieten, wurden am Ende dieses Kapitels realistische Gesamtmessungen aufgelistet, die auf einer moderneren Testhardware mit neuesten Grafikadaptern, und für komplexere Szenen vorgenommen wurden. Es wurde gezeigt, dass die entwickelten Renderingtechniken auf einer handelsüblichen PC Hardware interaktive Geschwindigkeiten erreichen.

Acknowledgements

Four years have passed since I embarked on my study in Computer Graphics. During these years, I have been greatly indebted to many people and I would like to acknowledge and thank them.

My biggest and most heartfelt thank you goes to my thesis and academic supervisor, Prof. Dr.-Ing. Dr. h.c. mult., Dr. E.h., Hon. Prof. mult. José L. Encarnação for his support, advice, encouragement during these years.

I would like to express my deepest gratitude to Prof. Dr. Henry Fuchs for co-supervising my work.

I am especially thankful to a small group of people, with whom I have had the opportunity to work very closely: Dr. Miguel Encarnação (Fraunhofer CRCG, Providence, RI), Dr. André Stork (Fraunhofer IGD, Darmstadt, Germany), Dr. Dieter Schmalstieg (Vienna University of Technology, Austria), and Dr. Bernd Fröhlich (Bauhaus University Weimar, Germany). They all have been an invaluable source of knowledge and inspiration. They shaped me - I have learned a lot.

Thanks go to all my friends, former colleagues and students at the Fraunhofer Institutes for Computer Graphics in Rostock and Darmstadt, and at the Fraunhofer Center for Research in Computer Graphics in Providence, RI for helping me to realize all this. Their friendship and encouragement made previous years more enjoyable and the tough times more bearable. Unforgettable thanks to my friends in the United States for a great time.

I would like to thank my mother, who has had a difficult time in bringing up my brother and myself on her own. Without her support, love and sacrifice I would never have achieved what I have. I will never forget it.

These acknowledgments would not be complete without mentioning the following person who is the most important part of my life. I would like to thank my girlfriend, Melanie, for her endurance, support and love. This thesis is dedicated to her. I love you.

Weimar, October 15th, 2002

-Oliver Bimber-

Table of Content

5 Proof of Concept

6 Evaluation

List of Figures

List of Tables

1 Introduction

1.1 Motivation

The rapid advances in computing and communications are dramatically changing all aspects of our lives. In particular, sophisticated 3D visualization, display, and interaction technologies are being used to complement our familiar physical world with computer-generated augmentations. These new interaction and display techniques are expected to make our work, learning, and leisure environments vastly more efficient and appealing.

Within different application areas, variants of these technologies are currently being pursued in research and development efforts. Virtual Reality (VR) attempts to provide to the user a sense of spatial presence (visual, auditory, and tactile) inside computer-generated synthetic environments. Opaque head-mounted displays (HMDs) have been the traditional VR output devices for many years.

A general characteristics of today's HMDs, however, is their imbalanced ratio between heavy optics (that results in cumbersome and uncomfortable devices) and ergonomic devices with a low image quality (i.e., low resolution, small field of view and fixed focal length).

To overcome some of their technological and ergonomic shortcomings and to open new application areas, the Virtual Reality community orients itself more and more away from HMDs, towards projection-based spatial displays such as immersive surround screen displays and semi-immersive embedded screen displays. Compared to HMDs, these new devices offer many advantages (e.g., a high and scalable resolution, a large and extendable field of view, an easier eye accommodation, a lower incidence of discomfort due to simulator sickness, lightweight glasses, etc.). In addition, many of them have particular characteristics (such as shape and size) that lend themselves for being employed as metaphors for application-specific functionality, thus making them easier to integrate into our everyday environments. Good examples for this are semi-immersive workbenches whose horizontal display surface lends itself towards supporting a table metaphor for the corresponding Virtual Reality setup.

Augmented Reality (AR) superimposes computer-generated graphics onto the user's view of the real world. In contrast to VR, AR allows virtual and real objects to coexist within the same space. Video see-through and optical see-through HMDs are the traditional output technologies, and are still the display devices that are mainly used for Augmented Reality applications. A reorientation of the AR community towards an alternative display technology has not yet happened. Most of the developments and progress made so far are based on very specific applications and technology-tailored employment scenarios. The majority of AR achievements has found few real-world applications. This can partially be attributed to the underlying core technology of AR - including its display devices.

As for many other technological domains, AR needs to provide sufficient robustness, functionality and flexibility to find acceptance and to support its seamless integration into our well-established living environments. For instance, many of our real-world items, devices, and tools are developed and tuned for effectively addressing distinct and problem-specific tasks. In contrast to this, many AR applications address specific problems still on an all-purpose technological basis - making use of technologically stagnating devices.

A high demand on alternative display technologies exists that improve the shortcomings of traditional devices and open new application areas for AR. Head-attached displays have first been developed in the mid-sixties and still today own the display monopole in AR field. In contrast to VR technology, however, they have barely improved over the previous years and are still far away from being "ultimate displays".

The presented *projection-based AR (PBAR)* concept aims to combine the technological and ergonomic advantages of the well established projection-based Virtual Reality with the appli-

cation potentials of Augmented Reality. Thus, it strives for opening new application areas for AR. It proposes -taking pattern from the evolution of VR- to detach the display technology from the user to embed it into the real environment instead. However, it is not intended to substitute other display concepts, such as head-attached displays, but rather to present an application-specific alternative.

1.2 Definitions

In general, we want to define a *projection-based Augmented Reality (PBAR)* configuration to be a spatial projection screen that is enhanced with optical see-through technology and supports stereoscopic, view-dependent and off-axis viewing of a graphically superimposed real environment.
In particular, we can characterize PBAR configurations to have the following properties:

- They combine optical see-through technology with spatial projection screens;
- Half-silvered mirror-beam splitters are applied as optical combiners (although the proposed concept can be extended toward other optical combination technologies, this work focuses on half-silvered mirror beam-splitters);
- They support the application of single or multi-faced planar optics as well as curved optics;
- They apply convexly curved and/or planar mirrors that form virtual images (although the proposed rendering techniques also support concave mirrors);
- They support static as well as flexible mirror-screen alignments;
- They provide a view-dependent image presentation, to dynamically display different perspectives of the presented scene;
- They represent general off-axis optical systems (however, the special on-axis case is included);
- They simultaneously support single or multiple observers;
- They apply several rendering and image transformation techniques that compensate for the optical effects that are produced by the elements of a PBAR configuration. These optical effects include reflection-deformations caused by mirrors, refraction-distortion caused by lenses (i.e., semi-transparent mirror-beam splitters), and optical distortion caused by miscalibrated displays;
- They require interactive stereoscopic rendering to make use of stereopsis.

Note that although the majority of the related systems that are discussed in this work share some of these properties, none of them provides a nearly complete match. These properties, however, strongly influence the device-specific rendering techniques, which differ from our general approach. Thus, most of the discussed systems can be seen as special PBAR variations that could be operated with the proposed general rendering methods.
We want to speak of *extended Virtual Reality (xVR)* if a PBAR configuration supports a seamless combination of VR and AR by approaching a conceptual and technical extension of traditional Virtual Reality by means of Augmented Reality. We can say that the xVR concept represents a special case of the PBAR concept.

1.3 Objectives and Conceptual Formulation

The general objective of the proposed concept is the utilization of optically enhanced spatial projection screens for Virtual/Augmented Reality tasks. It approaches to overcome some of the shortcomings, related to the traditional VR/AR devices within certain application areas, and to open new application possibilities. Thus, the utilization of optically enhanced spatial projection screens for VR/AR tasks allows to detach the display technology from the user.

From this general objective, the following specific questions can be derived:

- How to reduce window violation of semi-immersive projection screens that prevents a wide field-of-view?
- How to avoid occlusion of the projection screen by real objects?
- How to correct distortion that is caused by the applied optics?
- How to support multiple observers with a single PBAR configuration?
- How to calibrate PBAR configurations on an easy and intuitive basis?
- How to support interactive rendering for PBAR configurations?
- Can existing techniques (such as interaction, advanced rendering, real object registration, real-virtual object occlusion and collision detection, etc.) and applications be adapted to PBAR configurations?
- What are suitable applications for PBAR configurations?

Since PBAR supports flexible and non-static mirror-screen alignments and a view-dependent image presentation for single or multiple users for a variety of different mirror configurations, appropriate rendering and image deformation techniques must be developed. From the objective to provide interactive rendering for PBAR configurations, the following questions emerge:

- How to cancel out the physical image-deformations that are caused by the applied optical elements using techniques of computer graphics?
    - How to neutralize these deformations so that the optically formed images appear orthoscopic, stereoscopically and perspectively correct and undistorted to an observer.
    - Are approximations sufficiently precise?
- How to achieve interactive frame rates during navigation, interaction and scene modification?
    - Can PBAR configurations be driven by low-cost rendering hardware (such as PCs)?
    - How can rendering techniques for PBAR configurations provide the best image quality possible without losing their interactivity?
    - Can rendering techniques for PBAR configurations benefit from off-the-shelf acceleration hardware?
- Which levels of flexibility and independency have to be reached to support arbitrary PBAR configurations and to address a large variety of different applications?
    - Can rendering techniques be found that are independent of the application and the content that has to be rendered?
    - Can the interface between the application and the rendering framework be minimized?
    - Can rendering standards (such as rendering pipelines or scene-graphs) be used?

This work focuses on the development of rendering techniques for PBAR configurations. Several setups are realized on a proof-of-concept basis to demonstrate the rendering techniques' feasibility and the concept's applicability within different application domains, and to give answers to the questions mentioned above.

1.4 Summary of Results

Within the scope of this work, a projection-based Augmented Reality concept is proposed. This concept is implemented in form of proof-of-concept hardware configurations which dem-

onstrate the concept's applicability within different application domains, and appropriate rendering techniques which support such configurations on an interactive basis.
Our final rendering techniques for planar and curved optics are flexible and independent enough to be smoothly integrated into existing software frameworks. They are extendable and configurable since they utilize component-pipeline concepts, and they are general enough to support different PBAR configurations. Additionally, they take as much advantage of off-the-shelf hardware acceleration as currently possible and provide interactive frame rates on low-cost rendering hardware such as PCs. We can show that for the case of PBAR configurations, our image-based approach for curved optics performs better than adapted variations of recent geometry-based algorithms that are developed to support interactive rendering of view-dependent global lighting phenomena within 3D scenes [Ofe98, Ofe99].
Specifically for curved-optics displays which correct non-linear distortion by applying multi-pass rendering and image warping, we introduce a novel algorithm that generates appropriate regional levels of detail instead of applying a uniform image geometry (e.g., as done in [Vanb00, Ras01, Ban01, Yan01]) during runtime. In contrast to previous approaches, this method allows to consider the error that is caused from a piecewise linear texture interpolation and to minimize it by adapting the underlying image geometry. On the one hand, the refinement algorithm prevents oversampling and texture artifacts. On the other hand, it speeds up rendering for such displays significantly while guaranteeing a maximal error on the image plane.
Furthermore, we prove that our general mathematical model for off-axis in- and in/out-refraction transformations includes the two special cases: the on-axis refraction transformation for centered systems, which is commonly referred to in the optics literature (e.g., [Per96]) and Heckbert's paraxial approximation of the refraction transformation used for beam-tracing [Hec84] (or later approaches that are based on Heckbert's method, such as [Die96, Die97]).
The realized proof-of-concept prototypes present possible solutions to several problems that can be attributed to today's projection displays, such as the window violation problem linked to semi-immersive projection screens, the occlusion problem linked to rear-projection screens, and the support of multi-user viewing. Shortcomings that are related to today's head-worn Augmented Reality technology, such as the imbalanced ratio between heavy optics (that results in cumbersome and uncomfortable devices) and ergonomic devices with a low image quality (i.e., low resolution, small field of view, and fixed focal length) are improved. Furthermore, additional disadvantages of other "unconventional" Augmented Reality approaches, such as the shadow-casting problem and the display surface restrictions of Spatially AR [Ras98c, Ras99], or the limited viewing area, the non-flexible projector/screen alignment, and the reduced see-through quality of transparent projection screens [Pro01, Ogi01] are addressed.
The main contributions of this work can be summarized as follows:

- Introduction and formulation of concepts:
  - The *projection-based Augmented Reality (PBAR)* concept that proposes to combine optical elements with today's projection technology. It combines the technological and ergonomic advantages of the well established projection-based Virtual Reality with the application potentials of Augmented Reality;
  - The *extended Virtual Reality (xVR)* concept as a special case of the PBAR concept. It allows a seamless combination of VR and AR by approaching a conceptual and technical extension of traditional Virtual Reality by means of Augmented Reality.
- Introduction and implementation of novel interactive rendering techniques that support planar and curved image forming systems. In particular:
  - *Geometry-based rendering* methods for absolute image forming systems. These methods are fully supported by off-the-shelf hardware acceleration;

    - *Multi-pass image-based rendering* methods for non-absolute optics. These methods are partially supported by off-the shelf acceleration hardware;
    - *Software acceleration schemes* (such as selective refinement, reactive progressive rendering, parallel processing and image coding) that speed up the image-based methods. Thus, making them available for low-cost rendering hardware such as PCs.
- Introduction and realization of novel PBAR devices that prove the rendering technique's feasibility, present possible solutions to problems that are related to existing VR/AR displays, and open new application areas. In particular:
    - The *Reflective Pad* and the *Transflective Pad* as first of a kind stereoscopic hand-held displays. They propose a possible solution to the window violation problem as well as the occlusion problem that are related to projection-based VR systems.
    - The *Extended Virtual Table* and the *Transflective Board* that support a conceptual and technical extension of traditional Virtual Reality by means of Augmented Reality, and a seamless integration of such technology into habitual work environments. In addition, they offer a large field of view, improved focal length characteristics, and a high and scalable resolution;
    - The *Virtual Showcase* as a new interactive presentation display that also detaches the technology mostly from the user and integrates it stronger into our habitual living environments. In addition, it offers the possibility to simultaneously support multiple viewers and to provide a seamless surround view on the presented content. These properties are unique for today's projection display technology and for more than two viewers.

1.5 Outline

Chapter 2 discusses the introductory principles of geometric optics that lay down the mathematical, physical and physiological foundations for this work. Further, relations of the derived equations to computer graphics techniques are outlined. Additionally, we introduce a general method for off-axis in- and in/out-refraction transformations and prove that the on-axis refraction transformation for centered systems, which is commonly referred to in the optics literature, is a special case of our method.

In chapter 3, we discourse the previous and related work from two points of view: the existing display technology and, the related rendering techniques. First, a general classification of today's stereoscopic displays is diagrammed to provide an overview. Then, we take a closer look at the subset of devices that are capable to support Augmented Reality applications, state their current technological shortcomings and distinguish them from our PBAR concept. Addressing our approach to use mirror beam-splitters as optical combiners in connection with off-the-shelf projection systems, we present the state-of-the-art of current mirror displays and differentiate the specific devices from our general approach. Finally, several existing rendering methods that generate view-dependent global illumination effects within graphical 3D scenes are analyzed.

Stimulated by the methods evaluated in chapter 3, in chapter 4 we present novel interactive rendering techniques that can be applied for PBAR configurations, built from planar and curved optics. We show how to neutralize affine transformations that are caused by absolute image forming elements (such as planar mirrors), and how to integrate them into traditional graphic pipelines to fully benefit from a potential hardware acceleration. Since the majority of all optical elements cause non-affine image mappings, we introduce an interactive image-based rendering concept and suitable software acceleration-schemes that support curvilinear image transformations (warping) to neutralize the effects of such optical elements. Finally, we

describe how these techniques can be extended towards non-planar projection surfaces and multiple projectors, or towards optical chains.
The applicability of the techniques that are presented in chapter 4 is proven in chapter 5 with the aid of adequate proof-of-concept prototypes. These prototypes represent novel projection-based AR devices. Additionally, a variety of adapted VR/AR techniques are described and possible applications of PBAR configurations are outlined. The realized demonstrators were focused on the three main application areas: engineering, scientific visualization, and cultural heritage. Each of the presented devices is discussed in detail, and is related to the corresponding AR display technology described in chapter 3, by comparing their advantages and disadvantages.
Chapter 6 presents several non-coherent evaluations of single rendering techniques that are introduced in chapter 4 and of proof-of-concept prototypes that are described in chapter 5. The main focus of this chapter lays on the analysis of precision of analytical approximations, the computational cost and the order-of-growth of algorithms, the performance measurements of selected techniques, and the optical characteristics of a selected proof-of-concept prototype. Especially to support the last point, a general method that allows for comparing field-of-view related characteristics of heterogeneous Augmented Reality displays which apply possible off-axis image planes, is introduced.
Finally, chapter 7 concludes this work and provides short-term and long-term perspectives for future activities.

2 Geometric Optics as Foundation

In general we understand under optics all appearances that are perceived by the human visual system [Per96]. The physical reason of these appearances, the light, has been analyzed early, and the basic principles of geometric optics and wave optics were outlined in the 19th century. In geometric optics the light is represented by individual rays that, in ordinary media, are represented by straight lines. An ordinary media is homogeneous (the same at all points) and isotropic (the same for all directions) [Kor91]. One of the basic hypotheses of geometric optics, the principle of P. de Fermat (1657), allows the representation of light rays within isotropic media that are independent of the light's wave nature. Today this hypothesis is known as the *principle of the optical path length*, and defines that the time that light travels on the path between two points is minimal.
In this chapter, we want to lay the mathematical, physical, and physiological foundations for the subsequent techniques and concepts. First, we describe Snellius' laws of refraction and reflection that can be derived from Fermat's principle. The next section discusses several variations of image formation and introduces stigmatism and absolute optical systems. Note that section 2.1 and 2.2 are summarized from [Per96], which can be referred to for in-depth details. While section 2.3 presents details on the image formation behavior of reflective optics (i.e., mirrors), section 2.4. discourses this for refractive optics (i.e., lenses). We introduce our general method for off-axis in- and in/out-refraction transformations and prove that the on-axis refraction transformation for centered systems -which is commonly referred to in the optics literature- is a special case of our method. In both sections the relations of the derived equations to computer graphics techniques are outlined. Furthermore, appendix A shows that Heckbert's paraxial approximation of the refraction transformation used for beam-tracing [Hec84] is also a special case of our general method. The final section in this chapter discusses the visual depth perception. The structure and functionality of the human eye as a complex optical system and the binocular interplay of two eyes are described. Moreover, stereoscopic displays are introduced as image presentation devices for stereo pairs.

2.1 Snellius' Laws

The following laws have been discovered by W. Snellius in 1621. They can also be derived from Fermat's principle (see [Per96] for details). They describe the reflection and refraction behavior of straight light rays at the interfacing surface between two homogenous media.
Figure 2.1 illustrates the interfacing surface that separates two homogenous media with the two indices of refraction η_1 and η_2. A light ray $\vec{r}$ intersects the surface at $\vec{i}$ (with the normal vector $\vec{n}$) and is refracted into $\vec{r}'$.

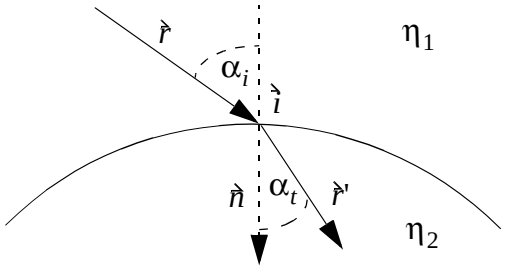


Figure 2.1: Snellius' law of refraction.

The vectorized form of Snellius' law is given by

$$\eta_2 \vec{r}' - \eta_1 \vec{r} = a\vec{n}, \tag{2.1}$$

where $\vec{r}$, $\vec{r}'$ and $\vec{n}$ are normalized and a is real.

2.1.1 Laws of Refraction

We can derive the following laws of refraction from the vectorized form of Snellius' law (eqn. 2.1).

First refraction theorem:

Because of $\vec{r}' = (\eta_1 \vec{r} + a\vec{n})/\eta_2$, the refracted ray $\vec{r}'$ lies on the plane that is spanned by $\vec{r}$ and $\vec{n}$. This plane is called the *plane of incidence.*

Second refraction theorem:

If we compute the cross-product with $\vec{n}$ and the vectorized form of Snellius' law (eqn. 2.1), we obtain $\eta_2(\vec{n} \times \vec{r}') = \eta_1(\vec{n} \times \vec{r})$. If we define the *angle of incidence* α_i and the *angle of refraction* α_t, we can substitute the cross-products and receive Snellius' law of refraction:

$$\eta_1 \sin\alpha_i = \eta_2 \sin\alpha_t \tag{2.2}$$

2.1.2 Laws of Reflection

Since the vector relation (eqn. 2.1) applies in general, we can assume $\vec{r}$ and $\vec{r}'$ to be located within the same medium with a refraction index η_1. Consequently, $\vec{r}$ is reflected at $\vec{i}$ into $\vec{r}'$ (cf. figure 1.2).

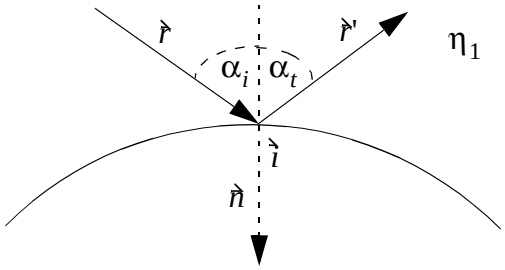


Figure 2.2: Snellius' law of reflection.

We can derive the following two theorems of reflection from the vectorized form of Snellius' law (eqn. 2.1).

First reflection theorem:

Because of $\vec{r}' = \vec{r} + (a/\eta_1)\vec{n}$, the reflected ray $\vec{r}'$ lies on the plane of incidence.

Second reflection theorem:
If we compute the cross-product with $\vec{n}$ and the vectorized form of Snellius' law (eqn. 2.1), we obtain $\vec{n} \times \vec{r}' = \vec{n} \times \vec{r}$. If we reassign α_t to be the *angle of reflection*, we can substitute the cross-products and receive Snellius' law of reflection:

$$-\sin\alpha_t = \sin\alpha_i \text{ or } -\alpha_t = \alpha_i \text{ for } -\frac{\pi}{2} \le \alpha_i \le \frac{\pi}{2}, -\frac{\pi}{2} \le \alpha_t \le \frac{\pi}{2} \tag{2.3}$$

Note that the law of reflection is formally based on the assumption that $\eta_2 = -\eta_1$ equals the reversion of the light rays.

2.1.3 Critical Angle and Total Internal Reflection

With $-1 \le \sin\alpha_i \le 1$, we can derive $-(\eta_1/\eta_2) \le \sin\alpha_t \le (\eta_1/\eta_2)$ from the second refraction theorem. It therefore holds that $-(\pi/2) \le \alpha_i \le (\pi/2)$ and $-\gamma \le \alpha_t \le \gamma$, whereby γ is called the *critical angle* and is defined by:

$$\gamma = \operatorname{asin}\left(\frac{\eta_1}{\eta_2}\right) \tag{2.4}$$

If α_i becomes sufficiently large, then α_t exceeds 90° and $\vec{r}$ is reflected from the interfacing surface, rather than being transmitted. This phenomenon is known as *total internal reflection*.

We can differentiate between two cases:

1. $\vec{r}$ enters an optically denser medium ($\eta_1 < \eta_2$): $\vec{r}$ is refracted for all angles of incidence α_i. If $\alpha_i = \frac{\pi}{2}$, then $\alpha_t = \operatorname{asin}(\eta_1/\eta_2) = \gamma$.
2. $\vec{r}$ enters an optically sparser medium ($\eta_1 > \eta_2$): If $\alpha_i \le \gamma$, then $\vec{r}$ is refracted. Else, $\vec{r}$ is reflected, due to total internal reflection.

2.2 The Formation of Point Images

Optical instruments can form *images* from a number of point-like light sources (so-called *objects*). Light rays that are emitted from an object can be reflected and refracted within the optical instrument, and are finally perceived by a detector (e.g., the human eye or a photographic film). If all light rays that are emitted from the same object $\vec{p}_o$ travel through the optical system[1] which bundles them within the same image $\vec{p}_i$, then these points are called a *stigmatic pair*. Consequently, this image-formation property is called *stigmatism* and the optical system that supports stigmatism between all object-image pairs is called an *absolute optical system*.
The basic precondition for stigmatism can also be derived from Fermat's principle. This is, that the optical path length for every light ray travelling from $\vec{p}_o$ to $\vec{p}_i$ is constant:

1. In the following sections, we also refer to single optical elements as optical systems.

$$L(\vec{p}_o \to \vec{p}_i) \;=\; \eta_1(\vec{i}_x - \vec{p}_o) + L(\vec{i}_x \to \vec{j}_x) + \eta_2(\vec{p}_i - \vec{j}_x) \;=\; const \qquad (2.5)$$

where η_1 and η_2 are the refraction indices at the entrance and the exit of the optical system.

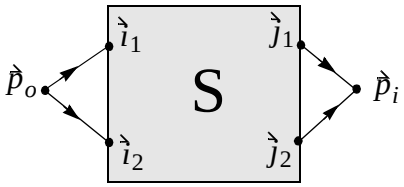


Figure 2.3: Stigmatic image formation (real object, real image).

If points (objects or images) are formed by a direct intersection of light rays, then these points are called *real*. Figure 2.3 illustrates the *real object* $\vec{p}_o$ whose emitted light rays pass through an optical system (the filled square) and intersect at the *real image* $\vec{p}_i$.

If light rays do not directly intersect within a point (e.g., if they diverge after exiting the optical instrument), they can form *virtual points*. Since human observers are only able to detect the directions of light rays, rays diverging from an optical system can appear to intersect within the system. These images are called *virtual images* (cf. figure 2.4).

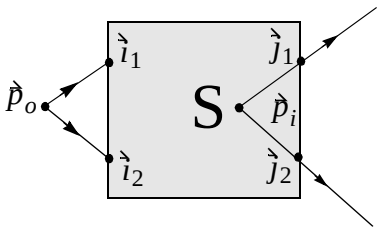


Figure 2.4: Stigmatic image formation (real object, virtual image).

The location of virtual points can be determined by extending the exiting light rays in negative direction. Consequently, this portion of the optical path is negative and must be subtracted from the total path length to satisfy equation 2.5.

As illustrated in figure 2.5, objects can also be virtual. In this case, the entering light rays have to be extended to find the location of the corresponding *real object*. As for the portion of the optical path to a virtual image, the sub-path to a virtual object has also to be subtracted from the total path length to satisfy equation 2.5.

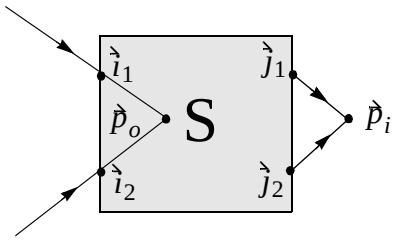


Figure 2.5: Stigmatic image formation (virtual object, real image).

The production of absolute optical systems is difficult, since the only surfaces that are easy to build and support stigmatism (some only for a single object-image pair) are planar or spherical surfaces. Therefore, most optical instruments only approximate stigmatic image formation [Per96]. The introduced deviation from the ideal image is called *aberration*.
We will give some examples of reflective and refractive optical systems in the following sections.

2.3 Reflective Optics

In case of exclusively reflective optical systems (*mirrors*), the medium that light rays travel through is homogeneous, thus $\eta_1 = \eta_2 = \eta$ and $\vec{i}_x = \vec{j}_x$. Consequently, equation 2.5 can be simplified:

$$L(\vec{p}_o \rightarrow \vec{p}_i) = \eta((\vec{i}_x - \vec{p}_o) + (\vec{p}_i - \vec{i}_x)) = const \tag{2.6}$$

If we further idealize that a mirror is surrounded by air, and that the medium air is approximately equivalent to medium vacuum ($\eta = 1$), then two stigmatic points which are formed within air are defined by the equation:

$$L(\vec{p}_o \rightarrow \vec{p}_i) = (\vec{i}_x - \vec{p}_o) + (\vec{p}_i - \vec{i}_x) = const. \tag{2.7}$$

2.3.1 Planar Mirrors

In case of planar mirrors $\vec{p}_o$ is real while $\vec{p}_i$ is virtual (cf. figure 2.6) and all points $\vec{i}_x$ of equation 2.7 describe the surface of a rotation-hyperboloid with its two focal points in $\vec{p}_o$ and $\vec{p}_i$. Planes represent a special variant of a rotation-hyperboloid, where $L(\vec{p}_o \rightarrow \vec{p}_i) = 0$. Planar mirrors are absolute optical systems that map each object $\vec{p}_o$ to exactly one image $\vec{p}_i$. Since this mapping is bijective, invertible and symmetrical for all points, they provide stigmatism between all objects and images. This means that images which are generated from multiple image points preserve the geometric properties of the reflected objects that are represented by the corresponding object points.

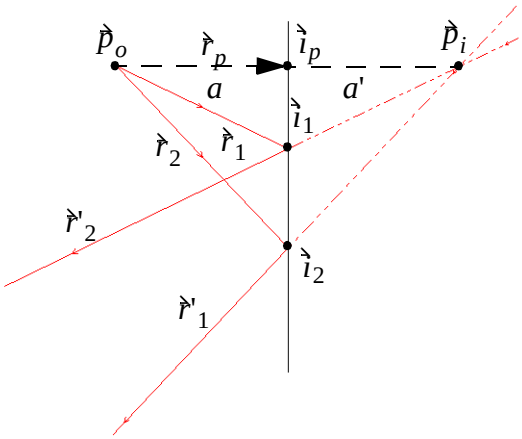


Figure 2.6: Planar mirror.

If we represent the mirror plane by its normalized plane equation within the three-dimensional space $f(x, y, z) = ax + by + cz + d = 0$, then the image for a corresponding object can be computed as follows:

With respect to figure 2.6, we can see that the distance from $\vec{p}_o$ to the mirror plane equals the distance from the mirror plane to $\vec{p}_i$ (i.e., $a = a'$). This can be derived from equation 2.7 with simple triangulation.

If we now define the ray $\vec{r}_p = \vec{p}_o + \lambda\vec{n}$, where $\vec{n} = (a, b, c)$ ($f(x, y, z)$ is normalized) is the normal vector, perpendicular to the mirror plane and λ an arbitrary extension factor of $\vec{n}$, we can insert the components of $\vec{r}_p$ into f and solve for λ:

$$f(\vec{r}_p) = \vec{n}\vec{r}_p + d = \vec{n}(\vec{p}_o + \lambda\vec{n}) + d = 0 \rightarrow \lambda = -\frac{1}{\vec{n}\vec{n}}(\vec{n}\vec{p}_o + d) \tag{2.8}$$

Since $|\vec{n}| = 1$, we can set $\vec{n}\vec{n} = 1$ and solve $\lambda = -(\vec{n}\vec{p}_o + d) = a = a'$. Consequently the intersection of $\vec{r}_p$ with f is given by:

$$\vec{i}_p = \vec{p}_o - (\vec{n}\vec{p}_o + d)\vec{n} \tag{2.9}$$

Since $a = a'$, the image point $\vec{p}_i$ results from:

$$\vec{p}_i = \vec{p}_o - 2(\vec{n}\vec{p}_o + d)\vec{n} \tag{2.10}$$

Equation 2.10 can be represented by the following homogeneous 4x4 transformation matrix:

$$R_l = \begin{bmatrix} 1-2a^2 & -2ab & -2ac & -2ad \\ -2ab & 1-2b^2 & -2bc & -2bd \\ -2ac & -2bc & 1-2c^2 & -2cd \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

where $[a, b, c] = \vec{n}$. Note that since $\vec{n}$ is normalized, the reflection matrix is equivalent to its inverse ($R_l = R_l^{-1}$). This matrix is applied by several computer graphics researchers to efficiently simulate reflections within a virtual scene [Hec84, Die96, Ofe99] without the application of time-consuming ray-tracing techniques [Glas89]. Here, either the scene's geometry or the camera position is reflected over planar mirrors that are part of the scene. The scene is then rendered multiple times -the original, untransformed scene, and its mirrored counterparts (respectively for each mirror within the scene). Finally, the different outcomes are merged within a single image. Note that these techniques are discussed in more detail within chapter 3.
With respect to Snellius' first reflection theorem, we can determine the reflection ray $\vec{r}'$ of the original ray $\vec{r}$ as follows:

$$\vec{r}' = \vec{r} - 2\vec{n}(\vec{n}\vec{r}) \tag{2.12}$$

In case of planar mirrors $\vec{n}$ is constant for all surface points $\vec{i}$. However, equation 2.12 is also valid for non-planar mirrors with individual normal vectors at each surface point. In this case, the intersection $\vec{i}$ of $\vec{r}$ with the mirror surface and the normal $\vec{n}$ at $\vec{i}$ have to be inserted in equation 2.12. Note that this is the common equation used by ray-tracers to compute specular reflection rays [Glas89]. The curvilinear behavior of reflections at non-planar mirrors can be well expressed with ray-tracing techniques, since they are based on the optical foundations of light rays. Furthermore, Ofek [Ofe99], for instance, shows a way of applying equation 2.11 as approximation to compute reflection-transformations for curved mirror surfaces. He uses the parameters of the planes that are tangential to possible intersection areas on a triangulated mirror surface.

2.3.2 Non-Planar Mirrors

In contrast to planar mirrors, non-planar mirrors do not provide stigmatism between all objects and images. In fact, only a few surface types generate just one true stigmatic pair. For all other objects (or for objects reflected by other surfaces), the corresponding images have to be approximated, since the reflected light rays do not bundle exactly within a single point.
As planar mirrors, convex mirrors generate virtual images from real objects. This is because light rays always diverge after they are reflected. Rotation-paraboloids (parabolic mirrors), for instance, can generate just one true stigmatic pair (cf. figure 2.7).

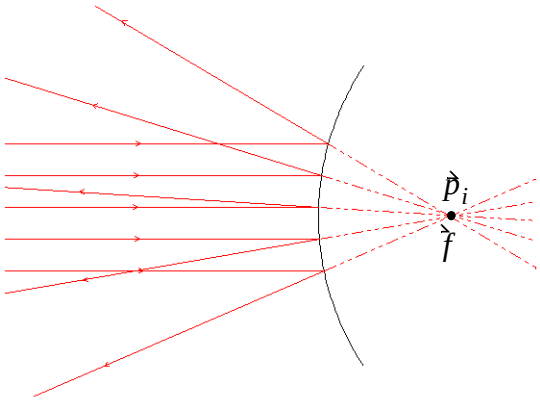


Figure 2.7: Convex parabolic mirror with object at infinity.

Only if $\vec{p}_o$ is located at infinity, the extended light rays bundle in one virtual point $\vec{p}_i$. This point is the *focal point* $\vec{f}$ of the paraboloid. The distance between the focal point and the surface is called *focal distance* or *focal length* f. For example, the focal length of a convex mirror is defined with $f = -r/2$, the focal length of a concave mirror is given by $f = r/2$, and the focal length of a planar mirror is $f = 0$, where r is the surface radius.
If $\vec{p}_o$ is not located at infinity, the extended light rays do not bundle exactly within a single image. Thus, $\vec{p}_i$ has to be approximated (cf. figure 2.8). Note that in this case images formed by multiple image points appear to be a reduced and deformed version of the reflected object that is represented by the corresponding object points.

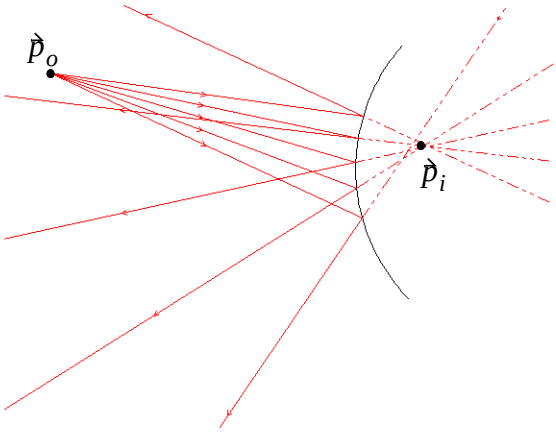


Figure 2.8: Convex parabolic mirror with finite object.

Beside rotation-paraboloids, other mirror surfaces (such as rotation-hyperboloids and prolate ellipsoids) can generate a single true stigmatic pair (see [Kor91] for more details). In general we can say that the true stigmatic pair, generated by such surfaces is always their two focal points. Mirror surfaces other than the ones mentioned above do not generate true stigmatic pairs at all.

Concave mirrors can generate both -virtual and real images from real objects because the reflected light rays converge or diverge, depending on the location of the object with respect to the focal point. As in the convex case, only the above mentioned surface types can generate just one true stigmatic pair which is their two focal points. For other surface types (or for objects that do not match the focal points), images can only be approximated.
Figure 2.9 illustrates an example of a concave parabolic mirror, where $\vec{p}_o$ is located at infinity and $\vec{p}_i$ is generated at $\vec{f}$.

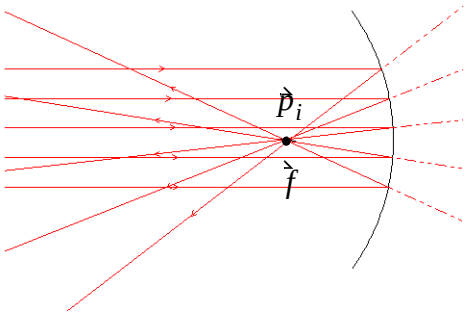


Figure 2.9: Concave parabolic mirror with object at infinity.

If $\vec{p}_o$ is not located at infinity, $\vec{p}_i$ has to be approximated. However, depending on the position of the object with respect to the focal point, the image can be either real or virtual. If, on the one hand, the object $\vec{p}_o$ is located behind the focal point $\vec{f}$ (as illustrated in figure 2.10) the reflected light rays converge and approximately bundle within the real image $\vec{p}_i$ (also located behind the focal point). Note that in this case images formed by multiple image points appear to be an enlarged, flipped and deformed version of the reflected object that is represented by the corresponding object points.

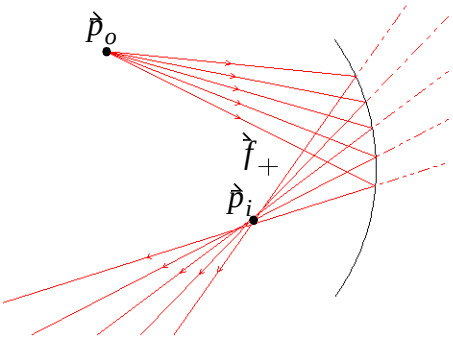


Figure 2.10: Concave parabolic mirror with finite object behind its focal point.

If, on the other hand, $\vec{p}_o$ is located between the surface and $\vec{f}$ (as illustrated in figure 2.11) the reflected light rays diverge and their extensions approximately bundle within the virtual image $\vec{p}_i$. Note that in this case images formed by multiple image points appear to be an enlarged and deformed version of the reflected object that is represented by the corresponding object points -yet, it is not flipped.

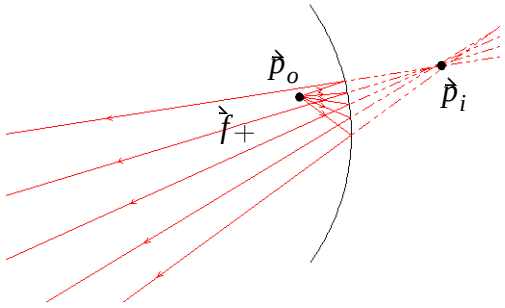


Figure 2.11: Concave parabolic mirror with finite object in front of its focal point.

Note that if $\vec{p}_o = \vec{f}$, then $\vec{p}_i$ is located at infinity (i.e., the reflected light rays are parallel). In this case, $\vec{p}_i$ is neither real nor virtual.

2.4 Refractive Optics

In case of refractive optical systems (*lenses*), the medium that light rays travel through is inhomogeneous. This means that, with respect to equation 2.5, light rays pass two different media with different densities and refraction indices $\eta_1 \neq \eta_2$, where η_1 denotes the refraction index of the medium that surrounds the lens, and η_2 is the refraction index of the lens material. Since the rays are redirected when they exchange into another medium (eqn. 2.2), their entrance and exit points differ $\vec{i}_x \neq \vec{j}_x$. We want to idealize again that a lens is surrounded by air and that the medium air is approximately equivalent to medium vacuum (i.e., $\eta_1 = 1$).

2.4.1 Planar Lenses

For the following, we want to refer to a homogeneous medium that is bound by two plane-parallel panels as a *planar lens*. Similar to planar mirrors, we can say that for planar lenses $\vec{p}_o$ is real while $\vec{p}_i$ is virtual (cf. figure 2.12).

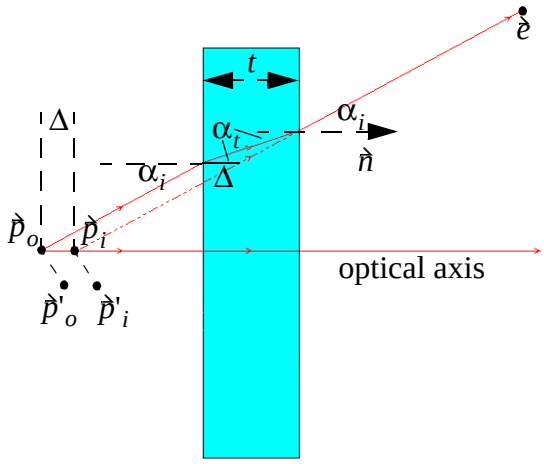


Figure 2.12: Planar lens.

Light rays are refracted twice -at their entrance points and at their exit points. This is referred to as *in-out refraction*. In case of planar lenses, the resulting out-refracted light rays have the same direction as the corresponding original rays, but they are shifted by the amount Δ parallel to their original counterparts. Since the original rays diverge from $\grave{p}_o$, the refracted rays also diverge from the lens. We have shown, that the offset Δ can be computed with [Bim01e]:

$$\Delta = t\left(1 - \frac{\tan\alpha_t}{\tan\alpha_i}\right) \tag{2.13}$$

where t is the thickness of the lens, α_i is the entrance angle of the original ray (as well as the exit angle of the out-refracted ray), and α_t is the exit angle of the in-refracted ray. It is constrained to the following boundaries:

$$lim\left(\alpha_i \to \frac{\pi}{2}\right) \Rightarrow \Delta = t \text{ and } lim(\alpha_i \to 0) \Rightarrow \Delta = t\left(1 - \frac{\sin\alpha_t}{\sin\alpha_i}\right) = t\left(1 - \frac{1}{\eta_2}\right) = const \tag{2.14}$$

True stigmatic pairs are not generated with planar lenses since the optical path length is not constant for all rays (i.e., equation 2.5 cannot be satisfied).
An approximation, however, can be made if the lens is centered. An optical system is centered (so-called *centered* or *on-axis optical system*) if the axis of symmetry of all surfaces and the optical axis coincide, whereby the *optical axis* is given by the center light ray of a light bundle with its direction pointing towards the propagation direction of the light. This situation is illustrated in figure 2.12. In this case, we can intersect the extended out-refracted light ray with the optical axis and receive the virtual image $\grave{p}_i$. The approximation is that we assume that the offset Δ is constant for all rays that diffuse from $\grave{p}_o$. This means that all extended out-refracted light rays intersect at the same point $\grave{p}_i$ on the optical axis. With respect to equation 2.14, Δ is given by

$$\Delta = t(1 - 1/\eta_2) \tag{2.15}$$

(with $\alpha_i = 0$) [Per96].

Note that equation 2.15, which is commonly referred to in the optics literature, can only be applied for on-axis (i.e., centered) optical systems. It is assumed, that a *detector* (e.g., a human eye) has to be located on the optical axis.

For centered optical systems, a further approximation is the assumption that adjacent points appear to transform similarly (from the detector's point of view). Thus, the offset Δ for $\vec{p}_o$ is the same as for $\vec{p}'_o$. The sine-condition of Abbe [Per96] describes this assumption for adjacent point-pairs that are located on the same plane, perpendicular to the optical axis. The condition of Herschel [Per96] expresses the preservation of stigmatism between adjacent point-pairs, located on the optical axis. These approximations represent the basis for all centered image-forming optical systems (mirrors and lenses) that do not provide true stigmatism for all points (i.e., for the majority of all optical systems). They describe the approximate preservation of stigmatism within the 3D free-space for centered optical systems. Nevertheless, they introduce aberrations.

Our approach (equation 2.14), on the other hand, is more general and can also be applied for off-axis (i.e., non-centered) situations. Thus, the detector does not have to be located on the optical axis. This is illustrated in figure 2.12, whereby the detector is indicated with $\vec{e}$. Rather than the on-axis equation (equation 2.15), the off-axis equation (equation 2.14) will be used by the following rendering techniques since we want to assume that the addressed optical systems that are utilized for PBAR configurations are general and not necessarily centered (i.e., the detector -or the eyes of a human observer, in our case- are not required to be located on the optical axis).

In either case, the translation of $\vec{p}_o$ to $\vec{p}_i$ can be represented by the following homogeneous 4x4 transformation matrix:

$$R_f = \begin{bmatrix} 1 & 0 & 0 & \Delta a \\ 0 & 1 & 0 & \Delta b \\ 0 & 0 & 1 & \Delta c \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.16}$$

with $\vec{n} = [a, b, c]$. However, since for a given $\vec{p}_o$ and a given $\vec{e}$ the geometric line of sight $\vec{p}_o - \vec{e}$ is know, but the geometric line of sight $\vec{p}_i - \vec{e}$, which is required to compute the correct Δ is unknown, we approximate that $\vec{p}_o - \vec{e} = \vec{p}_i - \vec{e}$. Since the angular difference between both lines of sight with respect to the plane's normal is minimal, the arising error can be disregarded. We will prove this within chapter 6 by presenting experimental measurements. Note that we make the same assumption for the following in-refractions and out-refractions at planar interfacing surfaces.

2.4.2 Planar Interfacing Surfaces

Given equation 2.13, we can also derive the off-axis *out-refraction* behavior of light rays that exchange from a denser medium into air at the intersection of a planar interfacing surface. In this case, the rays are refracted only once (cf. figure 2.13).

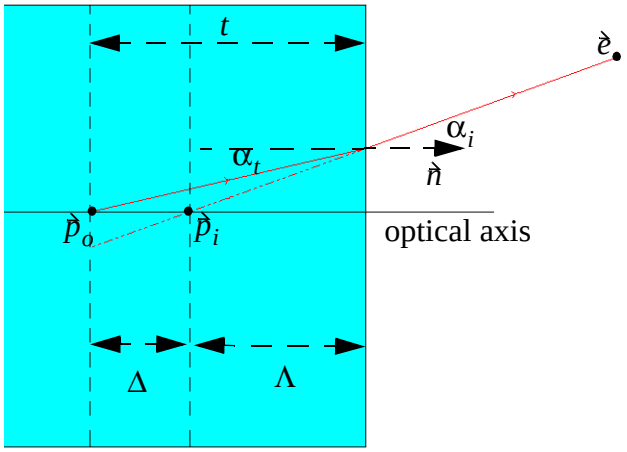


Figure 2.13: Out-Refraction at planar interfacing surfaces.

For the subsequent derivation from equation 2.13, we want to assume a planar lens that exceeds from the interfacing surface to $\vec{p}_o$. With respect to figure 2.13, we can say that $t = \Lambda + \Delta$. If we substitute Δ by equation 2.13 and simplify, we receive:

$$\Lambda = t\left(\frac{\tan\alpha_t}{\tan\alpha_i}\right) \text{ or } t = \Lambda\left(\frac{\tan\alpha_i}{\tan\alpha_t}\right) \tag{2.17}$$

If $\vec{p}_o$ is known, then t is the shortest distance between $\vec{p}_o$ and the plane of the interfacing surface, and is given by $t = \left|f(\vec{p}_o)\right| = \left|\vec{n}\vec{p}_o + d\right|$, where $f(x, y, z) = ax + by + cz + d$ and $\vec{n} = [a, b, c]$ defines the plane ($f(x, y, z)$ is normalized). If we now, in turn, use equation 2.17 to solve for Δ, we receive an additional equation for Δ:

$$\Delta = \Lambda\left(\frac{\tan\alpha_i}{\tan\alpha_t} - 1\right) \tag{2.18}$$

where Λ is constrained by the following boundaries:

$$lim\left(\alpha_i \to \frac{\pi}{2}\right) \Rightarrow \Lambda = 0 \text{ and } lim(\alpha_i \to 0) \Rightarrow \Lambda = t\left(\frac{\sin\alpha_t}{\sin\alpha_i}\right) = t\left(\frac{1}{\eta_2}\right) = const \tag{2.19}$$

With equation 2.19 we can derive the commonly referred refraction ratio for centered planar interfacing surfaces [Per96] (i.e., for the case that $\alpha_i = 0$):

$$\frac{\Lambda}{t} = \frac{1}{\eta_2} \tag{2.20}$$

This is equivalent to $\frac{\eta_2}{t} = \frac{\eta_1}{\Lambda}$ for a variable refraction index (Note that we assumed $\eta_1 = 1$ for air).
This closes the loop and proves that the on-axis refraction computations for centred systems (which are normally discussed in the optics literature) are a special case of our more general off-axis refraction method.
Note that the transformation matrix, defined by equation 2.16 can be used for out-refraction and to transform $\vec{p}_o$ to $\vec{p}_i$. For *in-refractions* (in our example: if the object is located within air and the detector is located within the denser medium) the refraction index, as well as the transformation direction have to be reversed.
In general, we can say that if the object is located within the denser medium and the detector is located within the sparser medium (i.e., the light rays are out-refracted), then the object's image appears closer to the interfacing surface. If, in turn, the object is located within the sparser medium and the detector is located within the denser medium (i.e., the light rays are in-refracted), then the object's image appears further away from the intersecting surface.

However, ray-tracers usually determine the specular refraction ray $\vec{r}'$ of an original ray $\vec{r}$ as follows [Glas89]:

$$\vec{r}' = \frac{\eta_1}{\eta_2}\vec{r} - \left(\cos\alpha_t + \frac{\eta_1}{\eta_2}(\vec{n}\vec{r})\right)\vec{n} \qquad (2.21)$$

As in the case of planar mirrors, for planar lenses or planar interfacing surfaces $\vec{n}$ is constant for all surface points $\vec{i}$. However, equation 2.21 is also valid for non-planar surfaces with individual normal vectors at each surface point. In this case, the intersection $\vec{i}$ of $\vec{r}$ with the surface and the normal $\vec{n}$ at $\vec{i}$ have to be inserted in equation 2.21. Like for reflection, the curvilinear behavior of refraction (for planar and for non-planar surfaces) can be well expressed with ray-tracing techniques. Furthermore, equation 2.17 (possibly in combination with the transformation matrix given by equation 2.16) can also be used for curved surfaces by applying it with individual normal vectors for each surface intersection.
Several approaches of realistic interactive rendering apply a homogeneous 4x4 or 3x3 matrix to simulate refraction within graphical scenes [Hec84,Die96,Ofe99]. For planar surfaces, such approaches apply the same transformation matrix to each geometry vertex -making use of hardware accelerated standard transformation pipelines [Hec84,Die96]. Ofek [Ofe99] uses the parameters of the planes that are tangential to possible intersection areas on a triangulated curved surface. These techniques are discussed in more detail within chapter 3.
Approximating refraction transformations with an affine mapping, however, does not express the curvilinear behavior of refraction. In addition, the application of these matrices provide only rough approximations of refraction transformations which are visually satisfying for rendered graphical scenes, but are less appropriate for optical systems.
In appendix A we show that Heckbert's paraxial approximation of the refraction transformation used for beam-tracing [Hec84] (which corresponds to the approximations for on-axis centered optical systems) is a special case of our general method.

2.4.3 Non-Planar Lenses

In practice, curved lenses are usually bound by two spherical surfaces. As in the case of spherical mirrors, spherical lenses do not have exact focal points - just areas of rejuvenation which

outline approximate locations of the focal points. Consequently, they cannot generate true stigmatic pairs. In contrast to lenses, mirrors are often shaped parabolically to provide exact focal points, and therefore provide one true stigmatic pair. For manufacturing reasons, however, almost all lenses that are used in optical systems are spherical. As described above, stigmatism can only be approximated for such optical systems.

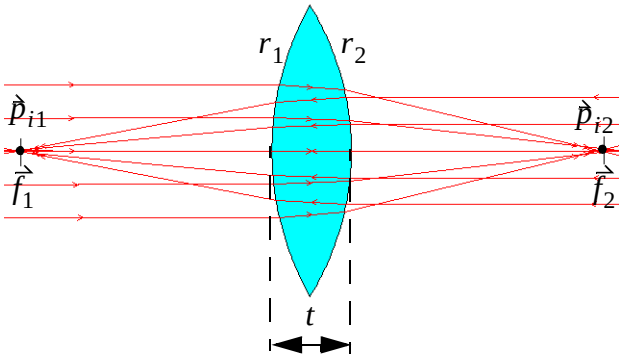


Figure 2.14: Convergent spherical lens with two objects at infinity.

The focal length of a spherical lens is defined by (cf. figure 2.14):

$$\frac{1}{f} = (n_2 - 1)\left(\frac{1}{r_1} - \frac{1}{r_2}\right) + \frac{(n_2 - 1)^2}{n_2}\frac{t}{r_1 r_2} \tag{2.22}$$

where r_1 and r_2 are the two surface radii and t the central thickness of the lens.

Convergent lenses are mostly bound by two convex spherical surfaces (cf. figure 2.14). Since light rays that are emitted from behind a focal point (i.e., further away from the lens) converge after exiting such a lens, real objects that are located behind a focal point form real images which are located behind the opposite focal point (cf. figure 2.15). Note that in this case images formed by multiple image points appear to be a reduced, flipped and deformed version of the refracted object that is represented by the corresponding object points.

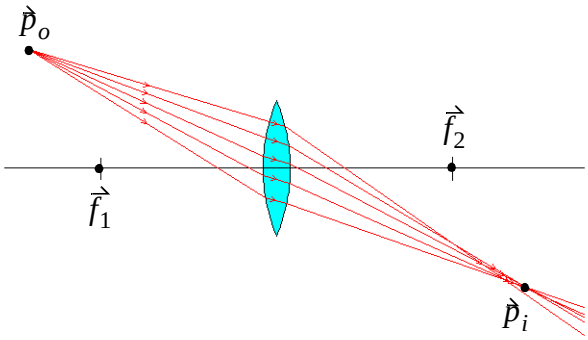


Figure 2.15: Convergent spherical lens with finite object behind a focal point.

If, however, the object is located between a focal point and the lens' surface, the light rays diverge after exiting a convergent lens. Thus, their extensions bundle within a virtual image point, in front or behind the same focal point (cf. figure 2.16). Note that in this case images formed by multiple image points appear to be an enlarged and deformed version of the refracted object that is represented by the corresponding object points -yet, it is not flipped.

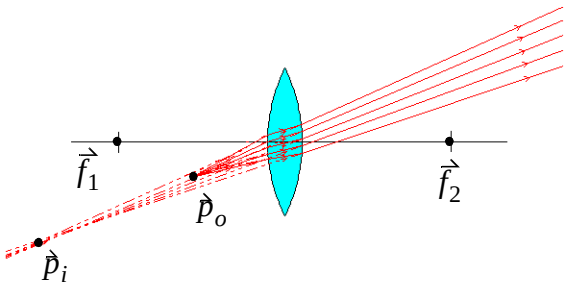


Figure 2.16: Convergent spherical lens with finite object in front of a focal point.

This behavior is very similar to the behavior of concave mirrors -although it is reversed.
Divergent lenses are mostly bound by two concave spherical surfaces (cf. figure 2.17). In case of divergent lenses, the exiting light rays always diverge. Thus, real objects always form virtual images -no matter where the object is located. This behavior can be compared with the behavior of convex mirrors. Yet, it is also reversed.

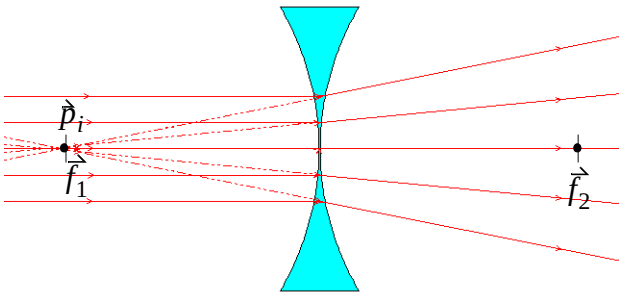


Figure 2.17: Divergent spherical lens with object at infinity.

Curved lenses that are bound by two parallel (concave or convex) spherical surfaces can also be considered. In this case, the thickness of the lens t is the same at all surface points. These lenses can only produce virtual images, since exiting light rays always diverge. This is illustrated in figure 2.18 for a convex lens, and in figure 2.19 for a concave lens.

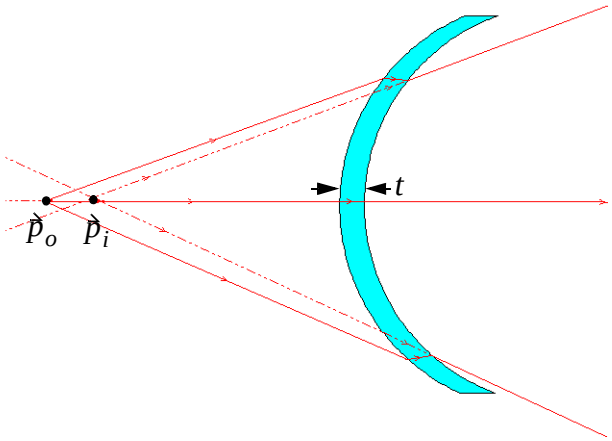


Figure 2.18: Convex parallel spherical lens with finite object.

Note that the object-image translation direction of a convex lens is reversed to the direction of a concave lens. However, in contrast to plane lenses but in correspondence with all other curved lenses, the out-refracted rays do not have the same direction as the corresponding original rays, and are not simply shifted parallel to their original counterparts. Thus, equation 2.13 and equation 2.15 in combination with approximations, such as the sine-condition of Abbe or the condition of Herschel do not apply in this case. Rather equations 2.18 or 2.21 can be used twice -for the in-refraction and for the out-refraction of a ray.

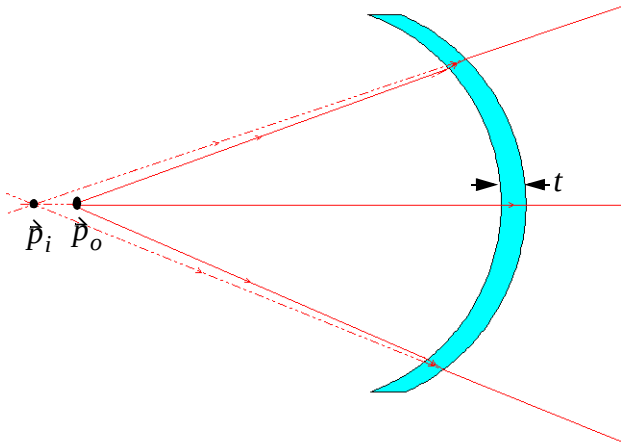


Figure 2.19: Concave parallel spherical lens with finite object.

2.5 Visual Depth Perception

In all optical systems, the light rays that are emitted by objects or images are finally perceived by light-sensitive components -the so-called *detectors*. An example of a detector is a photographic film (or disc or chip) used by cameras to preserve the received light information on a medium. The most common detector for optical systems, however, is the human eye which forwards the detected light information to the brain. The interplay of the two eyes that receive different two-dimensional images of the same environment (see from slightly different perspectives) enable the brain to reconstruct the depth information. This phenomenon is called *stereoscopic vision.* Stereoscopic vision can be fooled with s*tereoscopic displays* which present two artificially generated images of a virtual environment to the eyes of a human observer. As in the real world, these images are interpreted by the brain and fused to a three-dimensional picture.

2.5.1 The Human Eye

The human eye consists of spherical interfacing surfaces and represents a complex optical system by itself (cf. figure 2.20). Its approximate diameter is 25 mm, and it is filled with two different fluids -both having a refraction index of ~1.336. The *iris* is a muscle that regulates the amount of the incoming light by expanding or shrinking. The *cornea* and the elastic biconvex *lens* (refraction index ~1.4) below the iris bundle the transmitted light in such a way, that different light rays which are diffused by the same point light source are projected to the same point on the *retina.* Note that the projection is flipped in both directions -horizontally and vertically.

The retina consists of many small conic and cylindrical light-detecting cells (approximate size $\sim 1.5\eta m$ - $5\eta m$). The resolution of the eye depends on the density of these cells -which varies along the retina. If the distance between two point projections is too small, only one cell is stimulated and the brain cannot differentiate between the two points. To recognize two individual points, the two stimulated cells have to be separated by at least one additional cell. If the angle between two light rays that are emitted from two different point light sources and enter the eye is below 1.5 arc minutes, the points cannot be differentiated. The limited resolution of the human eye is the reason why light rays that are emitted from a single object point and pass through an optical system which does not support true stigmatism, still appear to intersect within a single image point. Thus, small aberrations of non-stigmatic and non-absolute optical

systems are not detected. Consequently, the observer perceives a single -possibly deformed- image of the object. The area with the highest resolution of detector cells is called *fovea.*

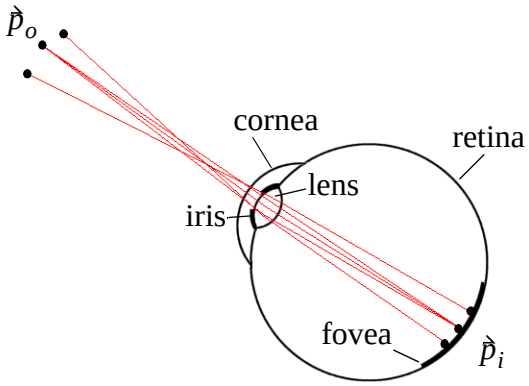


Figure 2.20: The human eye as an optical system.

In addition, the lens adjusts the focus for different distances (i.e., focal lengths) by deforming its shape. The deformation of the lens is called *accommodation*. The human eye can accommodate for focal lengths between ∞ and 100mm. Objects whose emanating light rays cannot be bundled by the lens on a single point of the retina appear unsharp. This happens, for instance, if the object is located closer to the eye than 100mm.
In-depth information on the structure and functionality of the human eye can be found in [Kre92] and [Per96].

2.5.2 Stereoscopic Vision

Two different views of the same object space are required to support depth perception. The perceptual transformation of differences between the two images seen by the eyes is called *stereopsis* [Hab73]. Due to the horizontal eye separation (*interocular distance* = ~6.3cm), the images that are perceived by the two eyes are slightly shifted horizontally, and are rotated around the vertical axis. Light rays that are emitted from an object project onto different locations on the respective retina. The relative 2D displacement between two projections of a single object onto two different focal planes (i.e., the left and the right eye's retina) is called *retinal disparity*. The stimulation of the detector cells at the corresponding locations is used by the brain to fuse the two images and to approximate the relative distance (i.e., the *depth*) of the object. Note that if the disparity between two projections becomes to large, the perceived images of the object space cannot be fused by the brain and the object space appears twice. This effect is called *diplopia* (double vision). To facilitate the accurate projection of the light rays onto the proper detector cells, the eyes have to rotate around the vertical axis until they face the focal point. This mechanism is called *vergence*. They can either rotate inwards to focus at close objects (*convergence*) or outwards to focus distant objects (*divergence*). If their alignment is parallel, an object at infinite distance is focused.

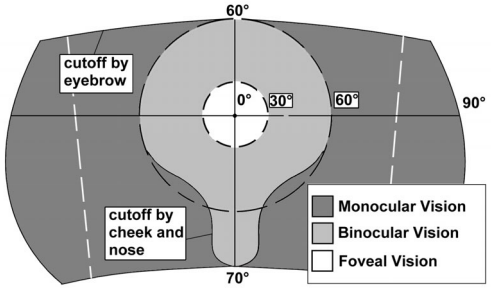


Figure 2.21: The human visual fields.

The total amount of the environment that can be seen by both eyes is called *monocular field of vision* and extends over a 180° horizontal field of view and a 130° vertical field of view (cf. figure 2.21). The portion of this visual field shared by both eyes is known as *binocular field of vision* and extends over a 120° horizontal field of view and a 130° vertical field of view. An even smaller portion within the binocular field of vision is called *foveal field of vision*. It is the area in which both eyes see in focus. It extends over a 60° horizontal and vertical field of view. Note that for stereoscopic vision, only the binocular field of vision is of interest [Hab73].
A good introductory review of stereoscopic vision from a human factors perspective is presented by Patterson [Pat92].

2.5.3 Stereoscopic Displays

Stereoscopic vision can be fooled with stereoscopic displays (cf. figure 2.22). For the subsequent explanation, we want to assume graphical stereoscopic displays which allow to draw pixels at their rasterized display surface.

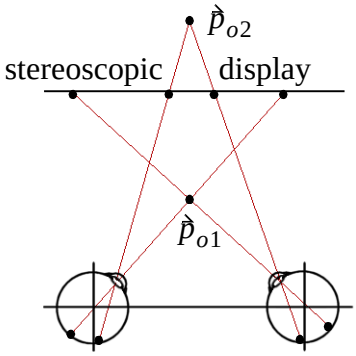


Figure 2.22: Stereoscopic vision with stereoscopic display.

Given a fictive object $\grave{p}_o$, we can determine the fictive light rays that would be emitted from $\grave{p}_o$ and intersect the eyes. For this, the eyes' positions need to be known, which are approximated by representing the eyes with single points that are located at the eye-balls' center. These rays are projected backwards onto the display surface and result in the positions of the pixels that are finally drawn on the display. The two related (left and right) projections of an object are called *stereo-pair*. Since the real light rays that are now emitted from the pixels intersect in $\grave{p}_o$ before they reach the eyes, the brain perceives the fictive object at its corresponding depth -floating in space. Such fictive objects (within the computer graphics commu-

nity also called *virtual objects*[2]) can be displayed so that they appear in front of (e.g., $\grave{p}_{o1}$), on, or behind the display (e.g., $\grave{p}_{o2}$). The apparent 2D relative motion of a distant object with respect to a close one as the viewpoint moves is called *parallax*. With respect to planar stereoscopic displays, parallax is usually defined as the distance between one object's pixel projection for the left eye and the same object's pixel projection for the right eye. In case the projections are on the same side as the corresponding eye the virtual object appears behind the display surface (such as $\grave{p}_{o2}$). This situation is called *positive parallax*. Note that the maximum positive parallax occurs when the virtual object is located at infinity. At this point the parallax equals the interocular distance.

If the virtual object appears in front of the display surface (such as $\grave{p}_{o1}$) then the projection for the left eye is on the right and the projection for the right eye is on the left. This is known as *negative parallax*. If the virtual object appears half way between the center of the eyes and the display, the negative parallax equals the interocular distance. As the object moves closer to the eyes, the negative parallax increases to infinity and diplopia occurs at some point.

In correspondence to this, the case where the object is located exactly on the display surface is called *zero parallax*.

An essential component of stereoscopic displays is the functionality to separate the left and right images for the respective eye when they are displayed. This means, that the left eye should only see the image that has been generated for the left eye, and the right eye should only see the image that has been generated for the right eye. Several mechanical, optical and physiological techniques exist to provide a proper *stereo separation*.

Examples of stereo separation techniques include:

- *Active shuttering* using head-worn glasses (shutter glasses) that open and close LCD panels in front of the eyes in synchronization with the corresponding left and right images that are display time-sequentially;
- *Passive shuttering* using head-worn glasses that apply color or polarized light filters in front of the eyes. The left and right image are pre-filtered and displayed simultaneously;
- Application of separate displays for each eye (e.g., head-mounted displays, BOOM-like displays, retinal displays, etc.);
- *Autostereoscopic displays* that utilize further optical elements (such as cylindrical or spherical lenses, holographic elements, etc.) in front of the display device. In this case, an additional head-worn device is not required.

Note that for stereo pairs that are projected onto a two-dimensional display to form a three-dimensional virtual object, the retinal disparity and the vergence are correct but the accommodation is inconsistent because the eyes focus at a flat image rather than at the virtual object. With respect to stereoscopic displays, disparity and vergence are considered as the dominate depth cues for stereoscopic viewing [Kal93, Stan98].

For further details on the implementation of stereoscopic graphics, see ACM SIGGRAPH course notes #24 on stereographics (1989).

2. Not to be confused with the previously used virtual object terminology of the optics community.

2.6 Summary

In this chapter, we discussed the basics of geometric optics and laid the mathematical, physical, and physiological foundations for the subsequent techniques and concepts. Our starting point were the laws of Snellius for reflection and refraction which led to image forming optical systems. Mirrors and lenses -as the two major components of our optical systems- and their image-forming behavior for different surface types have been described in detail and appropriate geometric object-image transformations have been presented. We have seen, that only cartesian surfaces (such as rotation-paraboloids, rotation-hyperboloids and prolate ellipsoids) can generate stigmatic image pairs. Only planar mirrors, however, provide true stigmatism between all object-image pairs and represent absolute optical systems. The other surface types, which provide stigmatism for a limited number of points (normally only for their focal points) are difficult to produce. This is the reason that most optical instruments approximate stigmatic image formation, and therefore introduce aberrations. We said, that the human eye itself is a complex optical system. Being the final link of an optical chain, the eyes can detect two perspectively different versions of the formed images, and send signals to the brain which fuses them to a three-dimensional picture. Disparity, vergence and accommodation are the main mechanisms to support stereoscopic viewing. However, due to the limited retinal resolution of the eyes, small aberrations of non-stigmatic optical systems are not detected. Consequently, the eyes perceive a single consistent image of the corresponding object -even if the light rays that are emitted by the object do not intersect exactly at the image point after travelling through the optical system. Finally, we have seen how stereoscopic viewing can be fooled with graphical stereoscopic displays by presenting different two-dimensional graphical images to both eyes.

We can summarize that for our purposes, an optical system consists of four major components: mirrors, lenses, detectors (the human eyes in our case) and displays (graphical stereoscopic displays in our case). The focus of this dissertation is to introduce view-dependent interactive rendering techniques that allow to enhance off-axis stereoscopic projection displays with arbitrary off-axis optical components. These techniques have to neutralize physical reflection/ refraction transformations of the projected graphics so that the optically formed images appear orthoscopic, stereoscopically and perspectively correct and undistorted to an observer.

3 Previous and Related Work

In this chapter, we want to discuss the previous and related work from both points of view: the existing display technology and the related rendering techniques.
While section 3.1 presents a general classification of today's stereoscopic displays, in section 3.2 we describe the subset of devices which are capable to support Augmented Reality applications, state their current technological shortcomings and distinguish them from our PBAR concept. The general objective of this concept is the utilization of optically enhanced spatial projection screens for Augmented Reality tasks to overcome some of the shortcomings, related to the traditional AR devices within certain application areas, and to open new application possibilities. Since this optical enhancement is realized by a flexible extension of off-the-shelf projection technology with mirror-beam splitters as optical combiners, section 3.3 presents the state-of-the-art of current mirror displays and differentiates the specific devices from our general approach.
Beside the technological issues, the PBAR concept requires interactive stereoscopic rendering that compensates for the view-dependent optical effects which are caused by the integrated optical elements. Therefore, several rendering methods that generate view-dependent global illumination effects (such as reflections and refractions) within graphical 3D scenes are analyzed in section 3.4. These methods stimulated and influenced the development of the rendering techniques which are used for PBAR setups.
Note that this chapter views the previous and related work from a fairly general perspective. More detailed relations to specific methods, techniques and approaches are explained in the corresponding sections of the following chapters.

3.1 Classification of Stereoscopic Displays

This section will provide a broad classification of today's stereoscopic displays (cf. figure 3.1). Note that we do not claim to present a complete list of existing systems and their variations, but rather focus on the technology that is (or might become) relevant for our work.

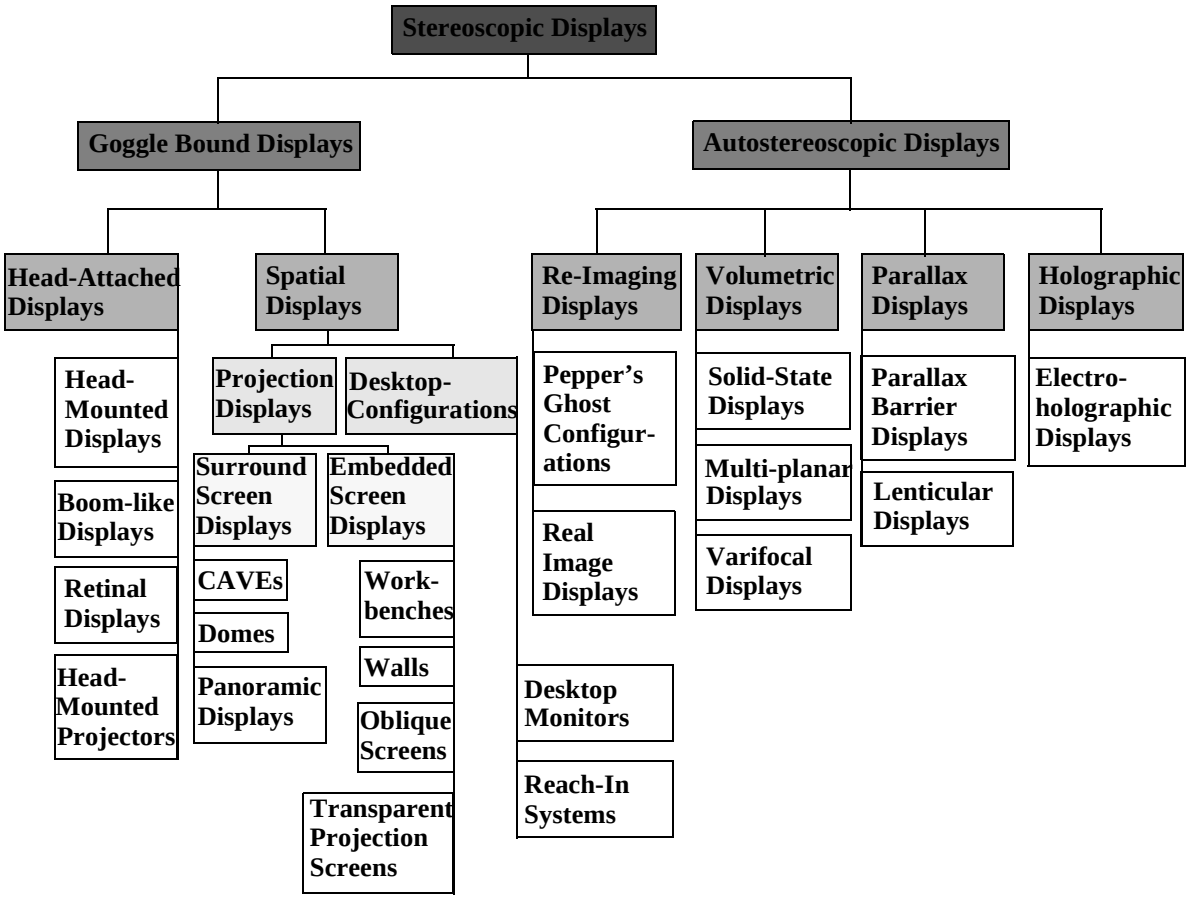


Figure 3.1: Classification of stereoscopic displays.

Stereoscopic displays can be divided into *autostereoscopic displays* and *goggle bound displays*. While goggle bound displays require the aid of additional glasses to support a proper separation of the stereo images, autostereoscopic displays do not. We will start to discuss autostereoscopic displays in more detail in section 3.1.1 and describe goggle bound displays in section 3.1.2.

3.1.1 Autostereoscopic Displays

Autostereoscopic displays [Hall97] present three-dimensional images to the observers without the need of additional glasses. Four classes of autostereoscopic displays can be found: *re-imaging displays, volumetric displays, parallax displays,* and *holographic displays*.

3.1.1.1 Re-imaging Displays

Re-imaging displays project existing real objects to a new position or depth. They capture and re-radiate the light from the real object to a new location in space. An important characteristic of re-imaging displays is that they do not generate three-dimensional images by themselves. Some re-imaging systems use lenses and/or mirrors to generate copies of existing objects. Especially half-silvered mirror setups are used by theme parks to generate a copy of a real three-dimensional environment and overlay it over another real environment. These types of mirror displays -so called *Pepper's ghost configurations* [Wal94]- generate virtual images and are further discussed below.

Other re-imaging displays apply more complex optics and additional display devices. For instance, some re-imaging displays generate a copy of a two-dimensional CRT screen which

then appears to float in front of the optics. These types of mirror displays -so-called *real image displays* [Eli72, Star83, Miz88, Wel89, Sum94, Chi95, Mck99a, Mck99b, Dim01]- generate real images and are also discussed in more detail below.
Another example of a re-imaging display was used by Sega in an arcade video game to relay and distort the appearance of a flat CRT screen into a curved surface [Hall97].
Re-imaging displays can be characterized by the following properties:

- Do not generate three-dimensional images by themselves;
- Generate visual copies of real objects (whereby the real object can be a computer-controlled screen);
- If a screen is re-displayed, the copy of the image that is shown on the screen remains two-dimensional (i.e., no autostereoscopic viewing is provided with respect to the displayed image);
- If a real object is re-displayed, the copy addresses the same visual depth cues as the original object (i.e., stereopsis, accommodation, vergence, parallax, etc.);
- With respect to re-displayed real objects, multiple observers are simultaneously supported.

Re-imaging displays are frequently applied as “eye-catchers“ for product presentation by the advertising industry, or to facilitate special on-stage effects by the entertainment industry.

3.1.1.2 Volumetric Displays

Volumetric displays [Bul00] directly illuminate spatial points within a display volume. In contrast to re-imaging displays, volumetric displays can generate synthetic images of voxelized data or three-dimensional primitives. These types of displays generate images by filling or sweeping out a volumetric image space.
Solid-state devices are variations of volumetric displays which display voxel data within a translucent substrate by generating light points with an external source (for example with lasers of different wavelengths located outside the substrate that are scanned through the image space) [Dow96].
Multi-planar volumetric displays build volumetric images from a time-multiplexed series of two-dimensional images. These images are displayed with a swiftly moving or spinning display element. This display element can be, for example, a rotating proprietary screen onto which the images are projected (e.g., using an external projector [Fav00] or lasers [Fav99]). Other systems directly move or spin light generating elements (e.g., light diodes). In either case, the human visual system interprets these time-multiplexed image slices as a three-dimensional whole.
Varifocal mirror displays [Tra67, Fuc82, Mck99a, Mck99a] are yet another group of volumetric displays. They apply flexible mirrors to sweep an image of a CRT screen through different depth planes of the image volume. These types of mirror displays are also discussed in more detail within the subsequent sections.
Regardless of the underlying technology, volumetric displays share the following characteristics [Hall97]:

- Presented volume can be perceived from a wide range of viewpoints, surrounding the display;
- Simultaneous support of multiple observers;
- Sense of ocular accommodation is supported;
- Spatial resolution of the presented graphics is limited;
- View-dependent shading and culling (required to simulate occlusion) of the presented graphics is not supported.

Particularly due to the last point, volumetric displays are mainly applied to present wire-frame or icon-based contents.

3.1.1.3 Parallax Displays

Parallax displays are display screens (e.g., CRT or LCD displays) that are overlaid with an array of light-directing elements [Hall97]. Depending on the observer's location, the emitted light that is presented by the display is directed so that it appears to originate from different parts of the display while changing the viewpoint. If the light is directed to both eyes individually, the observer's visual system interprets the different light information to be emitted by the same spatial point.
Examples of parallax displays are *parallax barrier displays* that apply a controllable array of light-blocking elements (e.g., a light blocking film or liquid crystal barriers [Perl00]) in front of a CRT screen. Depending on the observer's viewpoint, these light-blocking elements are used to direct the displayed stereo-images to the corresponding eyes.
Other examples are *lenticular sheet displays* that apply an array of optical elements (e.g., small cylindrical or spherical lenses) to direct the light for a limited number of defined viewing-zones.
Several properties characterize parallax displays [Hall97]:

- Simulation of occlusion is supported;
- Sense of ocular accommodation is supported (if no information reduction techniques are used that diminish or eliminate ocular accommodation);
- Limited viewing angle and discretized viewing zones;
- Single viewer devices (head-tracking is required if motion parallax has to be supported);
- Usually address only horizontal parallax (only lenticular sheet displays that apply spherical lenses support full parallax);
- Correct viewing distance has to be kept;
- Spatial resolution of the presented graphics is limited.

Parallax displays can be published and mass-produced in a wide range of sizes, and can be used to display photo-realistic images.

3.1.1.4 Holographic Displays

Holographic displays record the light's wavefront information that is emitted by an object within so-called *interference fringes*. The interference fringes can, under certain circumstances (if they are correctly illuminated), act as a complex diffractive lens that reconstructs the recorded light information (i.e., its direction and intensity). *Electroholographic displays* (such as the one described by Lucente [Luc97]) create these interference fringes electronically from a connected raster-engine.
Holographic displays share most of the properties of volumetric displays. The following characteristics, however, differ from volumetric displays:

- Low data bandwidth of high-quality holograms;
- Very low resolution of the presented holograms is supported (due to low bandwidth);
- Usually only horizontal parallax is supported (due to low bandwidth);
- Very limited in presenting shading and color information (due to low bandwidth);
- Restricted viewing angle.

Currently, holographic display technology is still far from producing high-quality three-dimensional images using affordable hardware [Hall97].

3.1.2 Goggle Bound Displays

Goggle bound displays require to wear additional goggle-like devices in front of the eyes to support a proper separation of the stereo images. They can be divided into *head-attached displays* and *spatial displays*.

3.1.2.1 Head-Attached Displays

Head-attached displays mostly provide individual display elements for each eye and consequently can present both stereo images simultaneously. Examples for such elements are miniature CRT or LCD screens that are applied in most *head-mounted displays* [Sut65, Sut68, Baj92, Mel96, Kai01] and *BOOM-like displays* [Fac01a]. *Retinal displays* [Kol93, Pry98] utilize low-power lasers to scan modulated light directly onto the retina of the human eye, instead of providing screens in front of the eyes. This produces a much brighter and higher resolution image with a potentially wider field of view than a screen-based display. *Head-mounted projective displays* [Pars98, Ina00] or *projective head-mounted displays* [Kij97] are projection-based alternatives that employ head-mounted miniature projectors instead of miniature displays. Such devices tend to combine the advantages of large projection displays with those of head-mounted displays.
The following characteristics can be related to head-attached displays:

- Simultaneous support of multiple observers (wearing individual devices);
- Mobile applications possible;
- Lack in resolution that is due to limitations of the applied miniature displays or projectors;
- Limited field of view that is due to limitations of the applied optics. Note that head-mounted projective displays and projective head-mounted displays address this problem;
- Sense of ocular accommodation is not supported due to a constant image depth and the resulting fixed focal length (for head-mounted displays and BOOM-like devices), or due to the complete bypass of the ocular motor-system by scanning directly onto the retina (retinal displays);
- In case of head-mounted projective displays: The inconsistency of accommodation and convergence is decreased since spatial projection surfaces are utilized;
- In case of retinal displays: Monochrome (red) images are presented since small blue and green lasers do not yet exist;
- Imbalanced ratio between heavy optics (that results in cumbersome and uncomfortable devices) and ergonomic devices with a low image quality;
- Increased incidence of discomfort due to simulator sickness in case of head-attached image planes (especially during fast head movements) [Patr00].

Head-attached displays (especially head-mounted displays) are currently the display devices that are mainly used for Augmented Reality applications.

3.1.2.2 Spatial Displays

Spatial displays apply screens that are spatially aligned within the environment. Nevertheless, the users have to wear field-sequential (LCD shutter-glasses [Nuv01, Ste01]) or light-filtering (polarization or color filters) goggles to support a correct separation of the stereo images. The stereo separation technique for spatial displays is generally known as *shuttering*, since each of the two stereo images which are presented on the same screen(s) has to be made visible to only one eye (i.e., the other image has to be blocked respectively -by shutting the eye). Depending

on the shuttering technology, the stereo images are either presented time sequentially (i.e., with field-sequential goggles) or simultaneously (i.e., with light-filtering goggles).
Spatial displays can be further divided into *desktop configurations* and *projection displays*. Using *desktop monitors* as a possible stereoscopic display is the traditional desktop-VR approach (also referred to as *fish tank VR* [War93]). Since desktop monitors (i.e., only CRT screens, but not LCD screens) provide the refresh rate of 120Hz that is required for a time-sequential shuttering, LCD shutter glasses are mostly applied for stereo separation. Note that older applications also use color-filtering glasses (e.g., red-green or blue-red filters) to separate monochrome stereo images. Fish tank VR setups are classified as non-immersive, since in contrast to large screens the degree of immersion is low. *Reach-in systems* represent another type of desktop configurations that consist of an upside-down CRT screen which is reflected by a small horizontal mirror. They are discussed in more detail below.

Projection displays currently apply cathode ray tube (CRT), liquid crystal display (LCD) or digital light (DLP) projectors to beam the stereo images onto single or multiple, planar or curved display surfaces. Two types of projections exist: With *front-projection*, the projectors are located on the same side of the display surface as the observer. Thus, the observer might interfere with the projection frustum and cast shadow onto the display surface. With *rear-projection* (or *back-projection*), the projectors are located on the opposite site of the display surface to avoid this interference problem.
Projection displays that first transmit the images through polarized light filters before they are diffused by the display surface require polarized glasses (i.e., glasses with corresponding polarization filters in front of each eye) to separate the images respectively. This technique is known as *passive shuttering*. For passive shuttering, at least two projectors are necessary to beam both polarized stereo images simultaneously onto the display surface. Note that special display surfaces are required for passive systems. These surfaces have to be built from a metallic material, since every organic material would reverse or destroy the polarization direction of the light and consequently would make the image separation fail.
Projection displays that beam both stereo images sequentially onto the display surface require field-sequential shutter glasses to separate the stereo images. This technique is known as *active shuttering*. For active shuttering, only one projector is necessary since the images are projected time sequentially. However, as with desktop monitors, these projectors have to support the required refresh rate of 120Hz.
Note that projection screens can either be *opaque* or *transparent* - depending on their application. Transparent projection screens are further discussed below.
Depending on the number and the shape of the spatially aligned display surfaces, we can divide projection displays into *surround screen displays* and *embedded screen displays.*

Surround screen displays surround the observers with multiple planar (e.g., CAVEs [Cru93], CABINs [Hir97]) or single curved display surfaces (e.g., Domes [Ben01] or panoramic displays [Tan01a]) to provide an *immersive* VR experience. Thus, the observers are completely encapsulated from the real environment. Usually, multiple projectors are used to cover the extensive range of the projection surface(s).

In contrast to surround screen displays, *embedded screen displays* integrate single, or a small number of display surfaces into the real environment. Thus, the users are not immersed into an exclusively virtual environment, but can interact with a *semi-immersive* virtual environment that is embedded within the surrounding real environment. Horizontal, *workbench-like* [Kru94, Kru95, Bar01a, Bar01b, Fak01b, Tan01a, Tan01b] or vertical *wall-like* [Sil01] display screens are currently the most common embedded screen displays.

Krueger's Responsive Workbench [Kru94, Kru95] is one of the pioneering *workbench-like projection systems*. The Responsive Workbench consists of a video projector that projects high-resolution stereoscopic images onto a mirror located under the table, which, in turn, reflects them in the direction of the table top (a ground glass screen). Analyzing the daily work situation of different types of computer users, Krueger et al chose a workbench-like system as an adaptation to the human living and working environment.
Based on the Responsive Workbench metaphor, a rich palette of similar rear-projection devices are available today that differ in size, mobility and applied projection technology. Among these systems are Wavefront's ActiveDesk, Barco's BARON [Bar01a], Fakespace's ImmersaDesk Series [Fak01b], and the Responsive Workbench [Tan01b] itself, which is sold by TAN Projectiontechnologies.
While all these systems are single-sided projection devices, a few two-sided (L-shaped) systems have been developed to offer a larger and (because of the normally limited projection area) less constrained viewing space. TAN's Holobench [Tan01c], for instance, is an extension of the Responsive Workbench, and Barco's Consul [Bar01b] has been developed based on the BARON Virtual Table.
Over the last years, an enormous variety of applications (concerning almost all VR areas) that involve table-like projection systems have been described.

Oblique screen displays represent a generalization of embedded screen displays, whereby special display surfaces are not integrated explicitly into the real environment. Rather, the real environment itself (i.e., the walls of a room, furniture, etc.) provides implicit display surfaces [Ras98a, Ras98b]. To support single or multiple front projections onto a multi-plane or curved display surface, a three-pass rendering method is applied. During the first rendering pass, the desired image of the virtual environment is generated from the observers current viewpoint. Then the generated image is projected out from the user’s point of view onto a registered virtual model of the display surface that is aligned with its real counterpart. For this, projective textures [Seg92] are applied. During the second pass, this textured model is rendered from the projector’s point of view and is finally beamed onto the real display surface. If multiple projectors are used, the second pass has to be repeated for each projector individually. The generated images have to be geometrically aligned and color and edge blended appropriately to realize a seamless transition between them. This is usually done during the third rendering pass [Ras98a].
Several general characteristics of projection displays can be found:

- High resolution (especially with tiled projection displays that apply multiple projectors [Fun00]);
- Lower incidence of discomfort due to simulator sickness than head-attached displays because of the spatially aligned image planes (fast head movements are not critical) [Patr00];
- Normally do not support multiple users;
- Passive shuttering, active shuttering in combination with rear-projection, and transparent projection screens require special display surfaces;
- Passive shuttering lacks from restricted head rotations[3] (due to the horizontal polarization direction of the light) and ghosting effects (due to the limited filtering capabilities of high contrast image portions);
- Active shuttering requires fast projectors which provide a high refresh rate;

3. Except advanced circular polarization filters are applied.

- Sense of ocular accommodation is not supported due to a spatially constant image plane. However, compared to head-attached displays, accommodation is improved since the image depth, and consequently the focal length is not constant and changes with a moving observer;
- Displays are stationary (i.e., mobile applications are not supported);
- Semi-immersive displays suffer from *window violation* (the clipping of the graphics by the display surface's physical edges).

Although head-attached (especially head-mounted) displays have a long tradition within the VR community, stereoscopic projection displays are currently the dominant output technology for Virtual Reality applications. Bryson [Bry97] sees various advantages of spatial displays over head-mounted displays.

3.2 Stereoscopic Augmented Reality Displays

If stereoscopic displays are used to present mixed (real and virtual) worlds, two basic fusion technologies exist: *video-mixing* and *optical combination*.
While video-mixing merges live record video streams with computer generated graphics and displays the result on the screen, optical combination generates an optical image of the real screen (displaying computer graphics) which appears within the real environment (or within the viewer's visual field while observing the real environment). Both technologies entail a number of advantages and disadvantages which influence the type of application they can address. A discussion on advantages and disadvantages of video-mixing and optical combination can be found in [Rol94] and [Azu97].
Several characteristics of Augmented Reality displays have been classified by Milgram et al [Mil94a, Mil94b]:

- Provides an egocentric (immersive) or exocentric (non-immersive) experience;
- Maintains an orthoscopic (1:1) mapping between size and proportions of displayed images and real environment;
- Offers a direct or indirect view on the real environment.

In this section, we discuss several types of Augmented Reality displays and want to relate them to the PBAR concept. Note that we rather present the display categories that are relevant for a comparison with the PBAR concept, than to provide a complete list of individual devices.

3.2.1 Screen-Based Augmented Reality

Screen-Based Augmented Reality has sometimes been referred to as *window on the world* [Fei93]. Such systems make use of video-mixing and display the merged images on a regular monitor. According to Milgram's classification [Mil94a, Mil94b], traditional screen-based Augmented Reality displays are exocentric, non-orthoscopic and provide a remote view on the real environment.
As fish tank VR systems which also apply monitors, window on the world setups provide a low degree of immersion. Within an Augmented Reality context the degree of immersion into an augmented real environment is frequently expressed by the amount of the observer's visual field (i.e., the field of view) that can be superimposed with graphics. In case of screen-based Augmented Reality, the field of view is limited and restricted to the monitor size, its spatial alignment relative to the observer, and its distance to the observer.
For screen-based Augmented Reality, the following disadvantages can be found:

- Small field of view that is due to relatively small monitor sizes;
- Limited resolution of the merged images (especially dissatisfying is the limited resolution of the real environment);
- Does not support the see-through metaphor, but rather provides a remote viewing;
- Direct interaction with the real environment and the graphical augmentation is not possible.

The PBAR concept also applies spatial displays to support Augmented Reality applications. However, it focuses on the utilization of large projection screens to provide a larger field of view and a higher resolution, and on the integration of optical see-through technology.
Note that screen-based Augmented Reality can be extended to large projection screens [Ras98a]. In this case, the shortcomings that are related to a small field of view, a limited resolution, and a remote viewing can be suspended (i.e., they can also be egocentric and orthoscopic).

3.2.2 Head-Mounted Displays

Head-mounted displays are currently the display devices which are mainly used for Augmented Reality applications. As illustrated in figure 3.1, they belong to the category of head-attached displays. Two different head-mounted display-technologies exist to superimpose graphics onto the user's view of the real world: *Video see-through head-mounted displays* that make use of video-mixing and display the merged images within a closed-view head-mounted display, or *optical see-through head-mounted displays* that make use of optical combiners (essentially half-silvered mirrors or transparent LCD displays). With respect to Milgram's classification [Mil94a, Mil94b], video and optical see-through head-mounted displays are egocentric, orthoscopic and provide a direct (optical see-through) or indirect (video see-through) view on the real environment.
However, several disadvantages can be related to the application of head-mounted displays as an Augmented Reality device. Note that most of these shortcomings are inherited form the general limitations of head-attached display technology:

- Lack in resolution that is due to limitations of the applied miniature displays;
- Limited field of view that is due to limitations of the applied optics;
- Imbalanced ratio between heavy optics (that results in cumbersome and uncomfortable devices) and ergonomic devices with a low image quality;
- Visual perception issues that are due to the constant image depth. Since objects within the real environment and the image plane that is attached to the viewer's head are sensed at different depths, the eyes are forced to either continuously shift focus between the different depth levels, or perceive one depth level unsharp. This is known as the fixed focal length problem, and is more critical for see-through than for closed-view head-mounted displays;
- Increased incidence of discomfort due to simulator sickness because of head-attached image plane (especially during fast head movements) [Patr00].

The PBAR concept attempts to detach the display device from the user and consequently to address some of the shortcomings that are related to head-mounted displays: The utilized projection technology is scalable -both in resolution and field of view. Addressing the ergonomic factor, glasses that have to be worn if active or passive shuttering is applied are much lighter and less cumbersome than head-mounted displays. Furthermore, since the reflected image plane can be spatially better aligned with the real environment that has to be augmented, the fixed focal length problem related to head-mounted displays (where the image plane is attached to the viewer) is reduced.

In addition, embedding the display technology into the real environment potentiality opens new application areas for Augmented Reality. However, such a concept will not substitute head-attached displays, but rather presents an application specific alternative.

3.2.3 Spatially Augmented Reality

Spatially Augmented Reality [Ras98c, Ras99] is another alternative to head-mounted displays with the same core idea than PBAR -namely to embed the display technology into the real environment. We can say that it represents an extreme case of oblique screen displays.
In Spatially Augmented Reality, front-projection devices are used to seamlessly project images directly on physical objects' surfaces instead of displaying them somewhere within the viewer's visual field, as it is done with head-mounted displays. With respect to Milgram's classification [Mil94a, Mil94b], spatially Augmented Reality is egocentric, orthoscopic and provides a direct view on the real environment. A stereoscopic projection and consequently the technology to separate stereo images is not necessarily required if only the surface properties (e.g., its color, illumination or texture) of the real objects are changed by overlaying images [Ras99]. In this case a correct depth perception is still provided by the physical depth of the objects' surfaces. This is similar to the notion of volumetric displays that also directly illuminate spatial points within a display volume to provide an autostereoscopic viewing.
However, if 3D graphics are displayed in front of the object's surfaces, a view-dependent, stereoscopic projection is required as for other oblique screen displays. In this case, the same three-pass rendering method is applied as described for the oblique screen displays in section 3.1.2.2. Raskar notes, that instead of projective texture-mapping, an image-based method can also be used that renders images with depth: (1) The virtual scene is rendered from the observers point of view. (2) While keeping the color values within the framebuffer, the depth-buffer is updated by rendering the registered virtual representation of the real objects' surfaces. (3) A 3D warp is applied to this depth image so that it is transformed into the coordinate system of the projector (using an accumulation of the inverse projection matrix, applied in (1) and (2), and the perspective projection matrix of the projector).
On the one hand, this overcomes some of the shortcomings that are related to head-mounted displays: an improved ergonomics, a theoretically unlimited field of view, a scalable resolution, and an easier eye accommodation (because the virtual objects are typically rendered near their real world location).
On the other hand Spatially Augmented Reality introduces several new problems:

- Shadow-casting of the physical objects and of interacting users that is due to the utilized front-projection;
- Restrictions of the display area that is constrained to the size, shape, and color of the physical objects' surfaces (for example, no graphics can be displayed beside the objects' surfaces);
- Restricted to a single user in case virtual objects are displayed with non-zero parallax.

As PBAR, Spatially Augmented Reality detaches the display device from the user -yet, on a different technological basis: PBAR makes use of optical see-through technology, instead of direct front-projection.

3.2.4 Transparent Projection Screens

In contrast to traditional front or rear-projection systems that apply opaque canvases or ground glass screens, *transparent projection screens* don't block the observer's view to the real environment behind the display surface. Therefore, they can be used as optical combiners that overlay the projected graphics over the simultaneously visible real environment. According to

Milgram's classification [Mil94a, Mil94b], transparent projection screens are semi-egocentric, orthoscopic and provide a direct view on the real environment. They belong to the category of embedded screen displays.
Pronova's HoloPro system [Pro01] is such a transparent projection screen. It consists of a multi-layered glass plate that has been laminated with a light-directing holographic film. The holographic elements on this film route the impinging light rays into specific directions, rather than to diffuse them into all directions (as it is the case for traditional projection screens). This results in a viewing volume of 60° horizontal and 20° vertical range in front of the screen, where the projected images are visible. Regular projectors can be used to rear-project onto a HoloPro screen. However, they have to beam the images from a specific vertical angle (36.4°) to let them appear within the viewing volume. Originally, the HoloPro technique has been developed to support bright projections at daylight.
Several shortcomings (mainly due to the applied holographic film) can be related to this technology:

- Limited and restricted viewing area;
- Static and constrained alignment of projector and projection plane (and therefore no flexibility);
- Low resolution of the holographic film (the pattern of the holographic elements are well visible on the projection plane);
- Reduced see-through quality due to limited transparency of non-illuminated areas.

Some researchers already begin to adapt this technology for Augmented Reality purposes [Ogi01]. In correspondence with PBAR, the application of transparent projection screens for Augmented Reality tasks also offers the potential to spatially embed optical see-through display technology into the real environment, but again on a different technological basis: While Ogi [Ogi01] applies a single planar transparent projection screen, PBAR applies mirror-beam splitters of different configurations (single or multiple planar, and curved mirrors) to support an optical see-through augmentation. Although multi-plane configurations are imaginable (but not yet realized), curved transparent projection screens do not exist and will be difficult to produce with holographic films that route the impinging light rays into specific directions.

3.2.5 Head-Mounted Projectors

Head-mounted projective displays [Pars98, Ina00] or *projective head-mounted displays* [Kij97] have recently been introduced as an alternative to head-mounted displays. Both devices apply head-mounted miniature projectors (LCD projectors or laser projectors), to beam the generated images from a dynamically moving center of projection. Thus, they approach to match the projector's center of projection and its projection frustum with the viewer's viewpoint and her viewing frustum. By doing this, the displayed images always appear optically undistorted -even when projected onto complex non-planar surfaces.
As head-mounted displays, we can count head-mounted projectors to the category of head-attached displays. With respect to Milgram's classification [Mil94a, Mil94b], head-mounted projector displays are egocentric, orthoscopic and provide a direct view on the real environment.
Head-mounted projective displays [Pars98, Ina00] redirect the projection frustum with a mirror beam-splitter so that the images are beamed onto *retro-reflective* surfaces that are located in front of the viewer. A retro-reflective surface is covered with many thousands of micro corner cubes. Since each micro corner cube has the unique optical property to reflect light back along its incident direction, such surfaces reflect brighter images than normal surfaces that diffuse light. Note that this is similar in spirit to the holographic films used for transparent projection

screens. However, these films are back-projected while retro-reflective surfaces are front-projected.
Projective head-mounted displays [Kij97] beam the generated images onto regular ceilings, rather than onto special surfaces that face the viewer. Two half-silvered mirrors are used to integrate the projected image into the viewer's visual field so that the projectors' parameters match the viewer's parameters (i.e., projection/viewing center and frustum).
Similar to SAR, head-mounted projective displays decrease the effect of inconsistency of accommodation and convergence that is related to HMDs. Both, head-mounted projective displays and projective head-mounted displays also address other problems that are related to HMDs: They provide a larger field of view without the application of additional lenses that introduce distorting arbitrations. They also prevent incorrect parallax distortions caused by IPD (inter-pupil distance) mismatch that occurs if HMDs are worn incorrectly (e.g., if they slip slightly from their designed position). However, they also introduce several shortcomings:

- Both, head-mounted projective displays and projective head-mounted displays are heavy and highly cumbersome;
- Head-mounted projective displays inherit the shadow casting problem from front-projection systems;
- The integrated miniature projectors offer limited (and unscalable) resolution and brightness;
- Head-mounted projective displays might require special display surfaces (i.e., retro-reflective surfaces) to provide bright images;
- For projective head-mounted displays, the brightness of the images depends on the environmental light conditions;
- Projective head-mounted displays can only be used indoors, since they require the presence of a ceiling.

Although such displays technically tend to combine the advantages of projection displays with the advantages of traditional HMDs, their cumbersomeness currently prevents them from being applicable. As head-attached displays in general, they suffer from the imbalanced ratio between heavy optics (or projectors) that results in cumbersome and uncomfortable devices or ergonomic devices with a poor image quality.

3.3 Mirror Displays

Beside several optical see-through head-mounted displays and head-mounted projector displays, a number of other display systems exist that apply full or half-silvered mirrors to achieve optical effects, such as an optical combination of graphics with the real environment. In this section, we want to discuss the different variations of *mirror displays* and state how they are related to our projection-based AR concept. Note that we describe only selected systems to introduce the corresponding display category they belong to, rather than presenting a complete list of systems.

3.3.1 Pepper's Ghost Configurations

Pepper's Ghost Configurations [Wal94] are a common theatre illusion from around the turn of the century named after John Henry Pepper - a professor of chemistry at the London Polytechnic Institute. They belong to the class of re-imaging displays. At its simplest, a Pepper's ghost configuration consists of a large plate of glass that is mounted in front of a stage (usually with a 45° angle towards the audience). Looking through the glass plate, the audience is able to simultaneously see the stage area and, due to the self-reflection property of the glass, a mir-

rored image of an off-stage area below the glass plate. Different Pepper's ghost configurations are still used by entertainment and theme parks (such as the Haunted Mansion at Disney World) to present their special effects to the audience. Some of those systems reflect large projection screens that display prerecorded 2D videos or still images instead of real off-stage areas. The setup at London's Shakespeare Rose Theatre, for instance, applies a large 45° half-silvered mirror to reflect a rear-projection system that is aligned parallel to the floor.
Although we also propose the application of half-silvered mirrors as optical combiners within the scope of the PBAR-concept, several general differences to Pepper's ghost configurations exist: PBAR applies stereoscopic projection displays that are extended by optical elements to overlay stereoscopic 3D graphics of a real environment. In an augmented reality context, this requires a view-dependent rendering. The main drawback of a Pepper's ghost configuration is that the viewers' parallax motion is very restricted because it forces the audience to observe the scene from predefined viewing areas. Moreover, our concept also includes the use of multi-face or curved mirror optics, rather than a single planar mirror only. This, for instance, allows us to observe an augmented environment from different perspectives - either sequentially by the same viewer, or simultaneously by different viewers.

3.3.2 Reach-In Systems

Reach-In Systems [Kno77, Sch83, Pos94, Wie99] are desktop configurations that normally consist of an upside-down CRT screen which is reflected by a small horizontal mirror. They can be considered as screen-based Augmented Reality systems which provide optical see-through. Nowadays, these systems present stereoscopic 3D graphics to a single user who is able to reach into the presented visual space by directly interacting below the mirror while looking into the mirror. Thus, occlusion of the displayed graphics by the user's hands or input devices is avoided. Such systems are used to overlay the visual space over the interaction space, whereby the interaction space can contain haptic information rendered by a force-feedback device such as a PHANTOM [Mas94]. While most reach-in systems apply full mirrors [Pos94, Wie99], some utilize half-silvered mirrors to augment the input devices with graphics [Kno77, Sch83] or temporarily exchange the full mirror by a half-silvered one for calibration purposes [Wie99].
Knowlton [Kno77], for instance, overlaid monoscopic 2D keycap graphics on the user's view of an otherwise conventional keyboard by using a half-silvered mirror that reflected a CRT screen. This allowed the graphics to annotate the user's fingers within the illuminated workspace below the mirror instead of being blocked by them.
Schmandt's Stereoscopic Computer Graphic Workstation [Sch83] is another early example of a reach-in arrangement that applies an electro-magnetic tracking device for input in combination with a CRT screen and a half-silvered mirror. He superimposed 3D graphics over the transmitted image of the working area below the mirror.
Poston and Serra [Pos94] developed the Virtual Workbench, but used a mechanical input device to overcome the magnetic field distortion problems of Schmandt's setup, which were caused by the interference between the CRT screen and the electro-magnetic tracking device.
A more recent development is the apparatus by Wiegand, Schloerb and Sachtler [Wie99] (also named Virtual Workbench). Their system offers a trackball for input, a Phantom for input and additional force feedback, and stereo speakers for auditory feedback.
Due to the small working volume of these devices, their applications are limited to near-field operations. Although some of these systems employ half-silvered mirrors instead of full mirrors for calibration purposes, only a few support Augmented Reality tasks. The maturity of systems, however, renders exclusively virtual (visual and haptic) information. Several of these devices are commercially available (e.g., the Reach-In Display by Reach-In Technologies

[Rea01] or the Dextroscope by the Medical Imagine Group Med[01]) and are mainly used for medical/industrial simulation and training, or psychophysics and training research [Wie99].
As with Pepper's ghost configurations, several distinctions to the PBAR-concept can be made: Similar to Pepper's ghost configurations, single planar mirrors with a static screen-mirror alignment (e.g., 30°-45°) are applied for reach-in systems - providing only one correct perspective (i.e., in case of reach-in systems for a single viewer, only). Although reach-in systems mostly present stereoscopic 3D graphics, a view-dependent rendering is normally not applied, since the user's head movements are naturally constrained by the near-field system itself.

3.3.3 Real Image Displays

Real Image Displays [Eli72, Star83, Miz88, Wel89, Sum94, Chi95, Mck99a, Mck99b, Dim01] are display systems that consist of single or multiple concave mirrors. Again, they belong to the class of re-imaging displays. As discussed in chapter 2, two types of images exist in nature -real and virtual. A real image is one in which light rays actually come from the image. In a virtual image, they appear to come from the reflected image - but do not. In case of planar or convex mirrors the virtual image of an object is behind the mirror surface, but light rays do not emanate from there. In contrast, concave mirrors can form reflections in front of the mirror surface where emerging light rays cross - so called "real images". Several real image displays are commercially available (e.g., [Dim01]), and are mainly employed by the advertising or entertainment industry. On the one hand, they can present real objects that are placed inside the system so that the reflection of the object forms a three-dimensional real image floating in front of the mirror. On the other hand, a projection screen (such as a CRT or LCD screen, etc.) can be reflected instead -resulting in a free-floating two-dimensional image in front of the mirror optics that is displayed on the screen (some refer to these systems as "pseudo 3D displays" since the free-floating 2D image has an enhanced 3D quality). Usually, prerecorded video images are displayed with such real image displays.
The main difference between real image displays and the PBAR-concept is that we exclusively apply convex and planar mirrors (although the proposed rendering techniques also support concave mirrors) -thus, they form virtual images, rather than real images. One fundamental point of our concept is to use half-silvered mirrors to superimpose the real environment with reflected graphics. This requires that the displayed virtual objects appear within the same spatial space as the real objects to be augmented. However, if the real environment was located within the same spatial space as the real image formed by a real image display (i.e., in front of the mirror surface), these objects would occlude the mirror optics and consequently the reflected image. Thus, if virtual objects have to be superimposed over real ones, real image displays suffer from similar occlusion problems as regular projection screens. The second distinction of our concept to real image displays is that they usually do not make use of stereopsis and, in addition, are normally not able to dynamically display different view-dependent perspectives of the graphically presented scene. Note that some approaches apply additional optical elements (lenses) to cause an autostereoscopic viewing for a static viewpoint (i.e., a very limited viewing area) [Mck99a, Mck99b].

3.3.4 Varifocal Mirror Displays

Varifocal Mirror Displays [Tra67, Fuc82, Mck99a, Mck99a] apply flexible mirrors and belong to the class of volumetric displays. In some systems the mirror optics is set in vibration by a rear-assembled loudspeaker [Fuc82]. Other approaches utilize a vacuum source to manually deform the mirror optics on demand to change it's focal length [Mck99a, Mck99b]. Vibrating devices, for instance, are synchronized with the refresh-rate of a display system that is reflected by the mirror. Thus, the spatial appearance of a reflected pixel can be exactly controlled - yielding images of pixels that are displayed approximately at their correct depth (i.e.,

they provide an autostereoscopic viewing and consequently no stereo-separation is required). Due to the flexibility of varifocal mirror displays, their mirrors can dynamically deform to a concave, planar, or convex shape (generating real or virtual images). However, these systems are not suitable for optical see-through tasks, since the space behind the mirrors is occupied by the deformation hardware (i.e., loudspeakers or vacuum pumps). In addition, concavely shaped varifocal mirror displays face the same problems as real image displays. Therefore, only full mirrors are applied in combination with such systems.

3.3.5 Hand-Held Mirror Displays

A *hand-held mirror display* for real-time tomographic reflection has been introduced by Stetton, et al. [Stet01]. It consists of a ultrasound transducer that scans ultrasound slices of objects in front of it. The slices are displayed time-sequentially on a small flat-panel monitor and are then reflected by a planar half-silvered mirror in such a way that the virtual image is exactly aligned with the scanned slice area. Stereoscopic rendering is not required in this case, since the visualized data is two-dimensional and appears at its correct three-dimensional location.

3.3.6 Image Transformation and Rendering Issues

For systems that reflect projection screens in mirrors, a transformation of the graphics is required before it is displayed. This ensures that the graphics are perceived orthoscopic, and not mirrored or distorted by the viewers. For systems such as Pepper's ghost configurations and reach-in systems (as well as for the described hand-held mirror display [Stet01]), this transformation is trivial (e.g., a simple mirror-transformation of the frame-buffer content [Kno77, Stet01] or of the world-coordinate-axes [Pos94, Sch83, Wie99]) since they constrain viewing to restricted areas and benefit from a static mechanical mirror-screen alignment. Some approaches combine this transformation with the device-to-world-transformation of the input device by computing a composition map during a calibration procedure and multiplying it with the device-coordinates during the application [Pos94]. Other approaches determine the projection of virtual points on the reflected image plane via ray-tracing and then map it to the corresponding frame-buffer location by reversing one coordinate component [Sch83, Wie99]. Mirror displays that apply curved mirrors (such as real image displays and varifocal mirror displays) generally don't pre-distort the graphics before they are displayed. Yet, some systems apply additional optics (such as lenses) to stretch the reflected image [Mck99a, Mck99b]. However, if a view-dependent rendering is required or if the mirror optics is more complex and does not require a strict mechanical alignment, these transformations become more complicated. The PBAR-concept supports flexible and non-static mirror-screen alignments and a view-dependent and off-axis image presentation for single or multiple users. It supports the application of single or multi-faced planar mirrors as well as curved mirrors, and proposes several interactive rendering and image transformation techniques that compensate for the optical effects that are produced by the elements of a PBAR configuration. These optical effects include reflection-deformations caused by mirrors, refraction-distortion caused by lenses (i.e., mirror beam-splitters in our case), and optical distortion caused by miscalibrated displays.

3.4 Rendering View-Dependent Global Illumination Effects

Image generation methods that simulate view-dependent global illumination effects (such as reflections and refractions) within graphical 3D scenes represent the basis for our rendering techniques. Different types of methods exist: *pixel-based, image-based, geometry-based (virtual viewpoint* or *virtual geometry)* and *hybrid methods*.

Pixel-based methods generate view-dependent global illumination effects on a per-pixel basis, while *image-based methods* express these effects within a single or multiple pre-computed image(s) that are applied to the geometry during runtime.
Geometry-based methods apply geometric rendering techniques and can be further subdivided into virtual viewpoint and virtual geometry methods. *Virtual viewpoint methods* generate multiple images from transformed viewpoints and merge them during runtime. In contrast to this, *virtual geometry methods* transform the geometry rather than the viewpoint. They might also generate multiple images of the differently transformed geometry which are finally merged during runtime.
Hybrid methods implement a composition of image-based and geometry-based methods.
While pixel-based methods largely generate photo-realistic images at non-interactive rendering rates, image-based, geometry-based or hybrid methods support interactive presentations of approximated images with less realism.
In the subsequent sections, we want to analyze several related rendering approaches that are able to generate view-dependent optical effects, such as reflections and refractions. These methods -especially the interactive ones- stimulated and influenced the development of our rendering techniques.

3.4.1 Ray-Tracing

The recursive *forward ray-tracing* algorithm [Whi80] is the traditional pixel-based approach to generate photo-realistic reflections and refractions on planar or curved surfaces. It traces light rays from the eyes of the viewer (who are physically located in front of a screen) through each screen pixel into the virtual scene that is defined behind the screen. If a ray intersects a scene object, it is reflected, refracted or absorbed -depending on the objects's material properties. Therefore, the laws of optics (i.e., Snell's laws) are directly applied to the original rays. The deflected rays are further traced through the scene on a recursive basis until they either intersect with a light source or pass freely without intersecting any obstacle. In this case, the exit condition of the recursion is reached. Since rays can split into multiple rays at the intersection points (e.g., if the intersected object's material is both -reflective and refractive), each ray describes a tree-like path through the scene. This is called a *ray-tree* and is usually the preferred data structure to store the ray information. The illumination values that are generated at each intersection are stored within the corresponding nodes of the ray-tree. Once the ray-tree is complete, it is evaluated bottom-up and the illumination values at each node are accumulated. The final results represent the color values of the related pixels on the screen.
The main drawback of this basic ray-tracing concept, however, is its lack in performance. To find the next object that a ray intersects is computationally expensive, and much work in organizing the virtual scene with efficient data structures (e.g., *space partitioning methods* using nonuniform octrees [Gla84] or uniform spatially enumerated auxiliary data structures (SEADS) [Fuj86]) and simplifying the intersection tests (e.g., using *single bounding volumes* [Whi80] or *hierarchical bounding volumes* [Rub80] in form of trees or directed acyclic graphs) has been carried out.
Adaptive depth control [Hal83] is another acceleration method that controls the recursion depth of the ray-tracer by attenuating the importance of rays the further they pass through the scene.
Note that since all ray computations are based on the original laws of optics, the optical effects that are caused by the (curved and planar) scene objects can be simulated on a very realistic and exact basis with forward ray-tracing.
While forward ray-tracing traces rays through the scene that originate from the viewpoint until they possibly intersect a light source, *backward ray-tracing* [Arvo86] first traces rays that are cast from the light sources until they intersect a diffuse surface. This is done in a first step to

detect indirect illumination effects that arise if the deflected light rays diffuse in different directions on the intersected surface points. Note that only specular surfaces (i.e., perfect reflectors and refractors) can be handled with regular forward ray-tracing, since the deflected rays are not diffused. A second pass then applies usual forward ray-tracing to compute the pixel colors under consideration of the previously generated indirect illumination values. This method is extremely slow and is characterized by an exponential complexity.

3.4.2 Beam-Tracing

Beside space subdivision and bounding volumes, another acceleration concept proposes the idea of tracing bundles of spatially coherent rays (beams of light), rather than treating every single ray separately. In *Beam-tracing* [Hec84], pyramid-like beam frustums are traced through the scene, until they hit a planar polygon of a reflector or refractor and generate deflected (reflected or refracted) beam frustums. By tracing the beams through the scene, a *beam-tree* is recursively generated which is similar in spirit to a ray-tree. While the beam-tree's links represent cones of light, its nodes contain the reflector/refractor polygons intersected by the light cones and corresponding model-view transformations which express the reflection and refraction characteristics of these polygons. But unlike a link in a ray-tree which always terminates on a single reflector/refractor polygon, the beam link may intersect many polygons. These polygons are depth sorted within each node to support a proper depth handling while the image is rendered. During the rendering pass, the beam-tree is traversed and for each node, the viewpoint is transformed with respect to the model-view transformation stored within the node (i.e., the scene geometry is transformed respectively into the actual viewing coordinate system). The transformed and projected scene polygons are shaded by computing the per-vertex illumination values within the untransformed world space. Finally, the scene polygons are drawn into the frame buffer using a polygon scan-conversion algorithm. Note that depth handling is correct due to the depth-sorted polygons stored within the beam-tree.

Beam-tracing belongs to the geometry-based virtual viewpoint methods, rather than to the pixel-based methods. The initial beam frustum is equivalent to the entire viewing frustum spanned over the edges of the screen. Reflected beam frustums are computed by reflecting the frustum's origin over the polygon plane of the intersected mirror (using equations 2.10 or 2.11) and tracing it over the reflector/refractor polygon's edges in the opposite direction. Note that this technique is limited to planar polygon surfaces. Reflections and refractions generated by curved surfaces would deform the beams and consequently destroy their spatial coherency. In addition, refractions of planar surfaces are approximated with a linear transform (perpendicular to the refracting plane) to conserve the beam coherence. For this approximation, Heckbert [Hec84] assumes, considering only paraxial rays (i.e., light rays that are exactly or nearly perpendicular to the refracting plane) that objects seen through a polygon with the refraction index η appear to be η times their actual distance. Note that this corresponds to our general refraction method (see eqn. 2.19 for the special case that $lim(\alpha_i \rightarrow 0)$). Consequently, a virtual focus point can be approximated for all paraxial rays which represents the origin of the refracted beam frustum and defines the model-view transformation stored at the corresponding node within the beam-tree. However, this approximation does not express the real curvilinear behavior of refraction and does not address in-out refractions.

In appendix A we show that Heckbert's paraxial approximation of the refraction transformation used for beam-tracing [Hec84] is a special case of our general off-axis refraction transformation method, described in section 2.4.2.

3.4.3 Environment Mapping

Environment Mapping [Bli76] is one of the early image-based techniques that approximates reflections on curved surfaces on interactive frame rates. Here, pre-generated textures that represent plenotopic functions at a single point, describing the incoming and outgoing light, are mapped onto curved scene objects. Since the environment maps represent directional information as a two-dimensional texture, a mapping from directions to texture coordinates is required. This mapping is called *parametrization*. Beside the traditional spherical environment maps that use circular textures in combination with a simple sphere parametrization [Hae93], other approaches use *cubical maps* (or *cube maps*) [Gre86, Voo94] that apply six perspective images, taken from the center of a cube through each of its faces, or *parabolic maps* (or *dual paraboloid maps*) [Hei98, Hei99] that utilize paraboloids instead of spheres for parametrization and texturing. Note that environment mapping usually makes extensive usage of hardware accelerated texture mapping.

Since with traditional environment mapping parallax is ignored, the resulting approximation is acceptable only if reflected objects are not located close to the reflector, and if the reflectors curvature is not too low. However, several environment mapping variations (such as *near environment mapping* and *depth-based mapping* [Ofe99]) exist which improve these problems but do not solve them.

Most applications per-compute the environment maps and consequently force the environment to be static. An on-the-fly computation for dynamic environments can only be achieved with multi-pass rendering technique and appropriate hardware support since the environment maps have to be re-computed on every frame.

Environment mapping can support interactive reflections on curved objects, but in contrast to ray-tracing it neither does provide a realistic accuracy nor does it address refraction.

See [Hei00] for a good discussion on state-of-the-art environment mapping techniques.

3.4.4 Reflection Mapping

Reflection Mapping [Fol90, Die96, Die97] is another virtual viewpoint method that simulates reflection on planar mirror surfaces using multiple rendering passes. For each mirror, the first rendering passes generate images of the scene from virtual viewpoints, which are computed by reflecting the original viewpoint over the corresponding mirror planes. These images are then merged with the primary image that is generated from the original viewpoint during the final rendering pass by, for instance, applying texture mapping [Fol90] or using the stencil buffer [Die96, Die97]. If the stencil buffer has more than one bit, multiple reflections can be simulated by recursing this procedure as described by Diefenbach [Die96, Die97].

An approximation of reflection mapping has also been discussed for curved surfaces [Hei00]. This method applies projective texture-mapping and alpha-blending between multiple reflection images. These images are generated for each reflector vertex whereby the viewpoint is mirrored over the plane that is tangential at the vertex. This method, however, was classified as inefficient in [Hei00].

Refractions are treated in a similar way as reflections by Diefenbach [Die96, Die97]. Instead of reflecting the original viewpoint over a mirror plane, a virtual viewpoint that is used to render the refracted image is generated by rotating the original viewpoint with respect to the refracting plane. The angle is determined by computing the refracting angle that results from the incident angle with the reflecting plane when Snell's law of refraction is applied. Diefenbach then noticed that this simple rotation transform does not provide a realistic refraction behavior and finally applies Heckbert's linear transform for paraxial rays as used for beam-tracing [Hec84]. However, as mentioned in section 3.4.2, this approximation does not express the real curvilinearity of the refraction transformations and does not address in-out refractions.

3.4.5 Virtual Object Method

The Virtual Object Method [Ofe98, Ofe99] is a virtual geometry technique that generates reflections and refractions on curved surfaces at interactive rates. In contrast to reflection mapping, the virtual object method reflects and refracts scene vertices over the tangent planes of reflector and refractor triangles, rather than to render from a reflected viewpoint. The transformed scene vertices are referred to as *virtual objects* and are rendered just like ordinary polygons. Here, the scene must be pre-tessellated with an appropriate resolution to express the curvilinear deformations of the geometry caused by curved mirrors and lenses. As in reflection mapping, the virtual object method merges different reflection and refraction images with the primary image (which shows the original scene without reflections and refractions) via texture mapping or the application of the stencil buffer.

Reflected virtual objects are generated with a method called *reflection subdivision*. The spatial space is subdivided into *reflection cells* (truncated tri-pyramid frustums) that are spanned by each triangle of the reflector's polygon mesh and the virtual viewpoint that has been determined by reflecting the original viewpoint over the plane that is tangential to the triangle. In addition, *hidden cells* are computed by casting truncated tri-pyramid frustums from the original viewpoint over each triangle's vertices into the opposite direction of the reflection cells. While scene vertices that are located within a reflection cell in front of a reflector triangle are reflected over its tangent-plane, scene vertices that are located within a hidden cell behind a reflector triangle are discarded. For scene polygons that lie partially within a hidden cell and partially within a reflected cell, the corresponding polygon vertices are simply doubled to provide a closed polygon for the underlying rendering pipeline. However, hidden vertices lie outside the reflection image and are automatically clipped during rasterization (i.e., after the reflection images have been merged with the primary image).

To accelerate the search for the corresponding cell that contains a scene vertex, another structure has been proposed by Ofek [Ofe98, Ofe99]. A so-called *explosion map* is determined for each reflector. The explosion map is represented by an image and is similar to a spherical environment map [Hae93]. It's pixel values are the IDs of the reflector triangles that project onto the reflector's bounding sphere. Projecting the scene vertices onto the explosion map is a fast way of determining the corresponding reflector triangle.

Similar to Diefenbach's first approach, Ofek [Ofe99] applies an affine shear transformation to simulate refraction. But in contrast to Heckbert [Hec84], Foley [Fol90] and Diefenbach [Die96, Die97] (who all compute a virtual viewpoint to generate reflection and refraction images for planar surfaces), Ofek transforms the scene geometry instead to support curved reflectors and refractors. However, as mentioned above and criticized by Diefenbach [Die96], this type of transformation is a poor approximation of refraction -even if only paraxial rays are considered.

Obviously, the space subdivisions and the explosion maps are viewpoint dependent and have to be recomputed each time the viewpoint or the reflectors/refractors move. In addition to an appropriate tesselation of the scene objects, high-resolution polygon meshes for the reflectors and refractors are required to support an acceptable approximation of the optical effects caused by these curved surfaces. Taking only multiple concave reflectors and refractors into account results in a worst-case time complexity of $O(r \times v)$, where r is the number of reflectors and refractors within the scene and v the number of vertices of the tessellated scene. In comparison to this, the time complexity of reflection mapping and beam-tracing is $O(r)$ -however, they only support planar surfaces. Additionally, Ofek's method does not generate recursive reflections as it is the case in Diefenbach's approach.

Note that although concave surfaces produce errors because of overlapping cells which (in some situations) result in an undefined mapping, Ofek treats concave surfaces exactly like con-

vex ones. For surfaces of mixed convexity, he subdivides the surface into convex and concave portions.

3.4.6 Pre-Computed Reflections

Bastos et al [Bas99a, Bas99b] propose a hybrid method (mixed -image-based and geometry-based) that generates reflection images for planar mirrors from a number of different reflected viewpoints during a pre-computation process (i.e., not at rendering time). As in several other approaches that are discussed above, Bastos merges the final reflection images with the primary scene image via stenciling.

Their method is based on the notion that a reflection is a sampling of the plenoptic function at a position in space over a range of viewing angles. In addition, they extended the general plenoptic function sampling to preserve depth information along each sampled direction. Consequently, the so-called *radiance map* stores the outgoing radiance values and the 3D location of the point that is intersected by the corresponding line-of-sight through each pixel.

During the pre-processing phase, a planar reflector is first uniformly tessellated into a number of *reflector elements*. For each reflector element, a hemisphere is then created behind it. Finally, for a finite number of sample points on the hemisphere, a radiance map is generated by rendering a (possibly off-axis) perspective image from each point, over the edges of the reflector element. In addition to the color values, this image contains depth values for each pixel.

At runtime, the original viewpoint is reflected over all mirror planes that are located within the scene. For each mirror, a subset of best-matching radiance maps for its individually reflected viewpoint are determined by finding the smallest camera-direction angles between all pre-generated perspectives and the current reflected perspective.

After the pixels of the selected radiance maps are re-projected to world space by applying the inverses of the transformation matrices which were used to compute the radiance maps, they are re-rendered into a new image from the new reflected viewpoint. Note that a re-projection from multiple radiance maps is required to reconstruct points which are occluded in one radiance map, but are visible from the current reflected viewpoint. However, since the radiance maps contain depth values, the z-buffer of the rendering framework ensures the correct occlusion of the re-projected points seen from the reflected viewpoint.

The final image needs then to be reconstructed from the new, but scattered and scaled (magnified or minified) image points. This disorder of the new image points results from re-projecting single pixels of different radiance maps and re-rendering them from a new viewpoint. Bastos corrects this by changing the size of the projected image points depending on the size of the projected radiance maps compared to its original size. This is similar in spirit to splatting techniques [Wes91, Rus00].

In contrast to geometry-based rendering methods, this hybrid approach requires a constant rendering time per reflector. The rendering time depends on the size of the radiance map and not on the complexity of the scene. However, this only pays off if the time to re-project the radiance maps, to re-render them, and to reconstruct the final image is shorter than the time to actually render the scene based on the geometry.

The major drawback of this approach is that it requires static scenes and can only be applied for planar mirrors. In addition, refractions are not addressed by this method at all.

3.5 Summary and Relations to Objectives

In this chapter we discussed the previous and related works which are most relevant to our projection-based AR concept, and to the methods and techniques that have been realized for it. We distinguished them from our approaches and pointed out parallel directions.

First, a classification of today's stereoscopic displays has been presented and several classes of autostereoscopic displays and goggle-bound displays have been described.
In general, we can see that most autostereoscopic displays do not yet support optical see-through Augmented Reality applications. This is generally due to the technological constraints of the applied optics. Exceptions are some mirror-based re-imaging displays. However, in the case that screens are re-displayed with such devices, the presented graphics remain two-dimensional and do not yield any autostereoscopic effect. Although an indirect "window on the world" view on the real environment supported by video-mixing would be feasible, autostereoscopic displays are barely used for Augmented Reality tasks.
While autostereoscopic displays do not require additional devices to address most visual depth cues, goggle-bound displays strongly depend on head-worn components to support a proper separation of the presented stereoscopic images. Video see-through and optical see-through head-mounted displays are today's dominant AR display devices. However, they entail a number of ergonomic and technological shortcomings. To overcome these shortcomings and to open new application areas, the Virtual Reality community orientates itself more and more away from head-mounted displays, towards projection-based spatial displays such as surround screen displays and embedded screen displays. Compared to head-mounted displays, projection-based devices provide a high and scalable resolution, a large and extendable field of view, an easier eye accommodation, and a lower incidence of discomfort due to simulator sickness. But they lack in mobility, in the ability to support multiple viewers (exceptions are several two-user approaches such as Stanford's two-user Responsive Workbench [Agr97] or UNC's two-user Protein Interactive Theatre [Arth98], and the recent IllusionHole setup [Kit01] that supports three users), and in optical see-through capabilities. Head-mounted projector displays might represent a compromise that tends to combine the advantages of HMDs with those of projection displays. However, they suffer from the imbalanced ratio between heavy optics (or projectors) that results in cumbersome and uncomfortable devices, and ergonomic devices with a low image quality. Currently, this is a general drawback of all head-attached displays that are dependent on miniature display elements.
Projection-based spatial displays in combination with video-mixing support a more immersive "window on the world" viewing. Video-mixing, however, still banishes the users from the real environment and, in combination with such devices, allows only a remote interaction. Compared to an optical combination, video-mixing also has a number of technological shortcomings, as outlined by Rolland [Rol94] and Azuma [Azu97]. Especially for projection-based display systems, problems that are related to the video-mixing technology, such as a time delayed video-presentation (due to the time required to capture and pre-mix the video streams), a reduced resolution of the real environment (due to the limited resolution of the cameras), and a strong limitation of head movements (due to restricted movements of the cameras) handicap the implementation of interactive and flexible Augmented Reality applications on this basis.
Especially within the Augmented Reality domain, a high demand on alternative display technologies exists that improve the technological, ergonomic and economic shortcomings of traditional devices and consequently open new application areas for AR. Head-attached displays have first been developed in the mid-sixties and still today own the display monopole in AR field. In contrast to VR technology, however, they barely improved over the previous years.
Initial Augmented Reality concepts are being proposed that suggest to detach the display technology from the user and to embed it into the real environment instead. Among these are Spatially AR, transparent projection screens and our projection-based AR. They all take advantage of today's advanced projection technology, but they differ in the way they combine real and virtual.
Projection-based AR and transparent projection screens approach an optical combination using additional spatially aligned optical elements (either half-silvered mirrors or semi-transparent

projection screens). Since projection-based AR applies half-silvered mirror-beam splitters because of their better optical properties and higher flexibility, the state-of-the-art of current mirror displays has been discussed and single devices have been differentiated from our concept.

For systems that reflect projection screens in mirrors, a transformation of the graphics is required before it is displayed. This ensures that the graphics are perceived orthoscopic and not mirrored or distorted by the viewer. While for some of the discussed systems this transformation is trivial and static, because they benefit from a fixed mechanical mirror-screen alignment and a constrained viewing, others do not pre-disport the graphics at all, or apply additional optical elements that provide an approximate neutralization of the appearing optical effects. However, all these systems do not support a pre-distortion of the graphics depending on the observer's actual and dynamically changing viewpoint. Either, they restrict the observer to a single point of view or a small viewing-area, or they accept optical distortions if the observer moves. Systems that apply additional optics to correct these effects are centered and consequently do not support an off-axis viewing.

Since our projection-based AR concept supports flexible and non-static mirror-screen alignments and a view-dependent and off-axis image presentation for single or multiple users for a variety of different mirror configurations, the development of more powerful rendering and image transformation techniques is required.

Several rendering techniques that simulate view-dependent global illumination effects within graphical 3D scenes have been analyzed. They represent a basis and the starting point for the development of our rendering and transformation methods. We categorized these techniques into pixel-based, image-based, and geometry-based approaches, whereby geometry-based approaches have been divided further into virtual viewpoint or virtual geometry methods. While photo-realistic rendering (i.e., the pixel-based methods) can precisely simulate optical effects and generate high-quality images, all other approaches can generate images at interactive rates. However, they approximate the optical effects more coarsely.

Image-based methods contain the optical effects and the resulting illumination information within a single or multiple pre-computed image(s) that are applied to the geometry during runtime. Geometry-based methods, however, recompute the optically distorted illumination effects for every frame.

Especially, interactive rendering techniques that still produce an acceptable image quality and techniques which can be supported by low-cost off-the-shelf acceleration hardware are of interest for our projection-based Augmented Reality concept.

4 Interactive Rendering

In this chapter we describe interactive rendering techniques that can be applied for planar (see section 4.1) and curved optics (see section 4.2). Depending on the applied optical components, these techniques have to neutralize physical reflection and refraction transformations of the projected graphics so that the optically formed images appear orthoscopic, stereoscopically and perspectively correct and undistorted to an observer. All four types of optical elements (as defined in chapter 2) are taken into account: reflectors, refractors, projectors, and perceptors.
First, we assume that a single planar display device (e.g., a rear-projection system) is used, and that the display device and a single optical element are defined within the same world-coordinate-system. In these examples, the projection plane matches the x/y-plane of the word-coordinate-system and the origin is located at the center of the projection plane. Later, in section 4.4 and section 4.5, we describe how these techniques can be extended towards non-planar projection surfaces and multiple projectors, or towards optical chains.
Further, we want to assume that the position of the viewpoints (i.e., the perceptors) are determined via head-tracking technology.
Since only planar reflectors are absolute optical systems and provide true stigmatism between all object/image pairs, we will be able to find affine model and viewpoint transformations for such elements. These transformations can be integrated into traditional transformation pipelines and can consequently be supported by hardware acceleration. In contrast to this, refractors (i.e., lenses) and curved reflectors do not map objects to images on an affine basis, but rather require curvilinear transformations that individually modify the per-vertex properties of each scene element (i.e., its coordinates, normal vectors, illumination parameters, etc.). We will introduce interactive rendering techniques in section 4.2 and acceleration schemes (such as selective refinement, progressive rendering and parallel processing) in section 4.3 that support interactive curvilinear transformations without the usage of expensive special-purpose acceleration hardware.
Particularly for optics that require non-affine transformations, we will focus on an image-based rendering approach. This approach is introduced in form of a pipeline which consists of single modular components, such as rendering steps, image transformations, and acceleration options.
The feasibility of the techniques that are presented in this chapter are proven with the aid of adequate proof-of-concept prototypes, described in chapter 5. However, they are not limited to these prototypes, but can be applied in general for any PBAR configuration.
Note that the following methods and algorithms are outlined in an OpenGL [Nei93] context. In the subsequent sections and chapters, we refer to transformations as *transforms*.

4.1 Planar Optics

In this section, we want to present rendering and transformation techniques that support PBAR configurations built from planar optical elements, such as planar reflectors (i.e., mirrors), refractors (i.e., lenses) and projectors (i.e., screens). Affine transformations that can be found for planar mirrors and screens are directly integrated into standard transformation pipelines (see appendix B) without causing additional computational cost. In addition, efficient approximations will be introduced to support non-affine transformations that are required for planar lenses or miscalibrated projectors.

4.1.1 Reflected View Transform

Planar full mirrors enable us to perceive the reflection of stereoscopically projected virtual scenes three-dimensionally and perspectively correct:
Instead of rendering the stereo images based on the user's physical point of view, the corresponding reflection of the user's viewpoint within the image space (i.e., the space behind the mirror plane) has to be employed.We refer to this as the *reflected view transform* [Bim00a].
Because of the symmetry between the real world and its reflected image, the physical eyes perceive the same perspective by looking from the object space through the mirror into the image space, as the reflected eyes perceive by looking from the image space through the mirror into the object space (cf. figure 4.1).

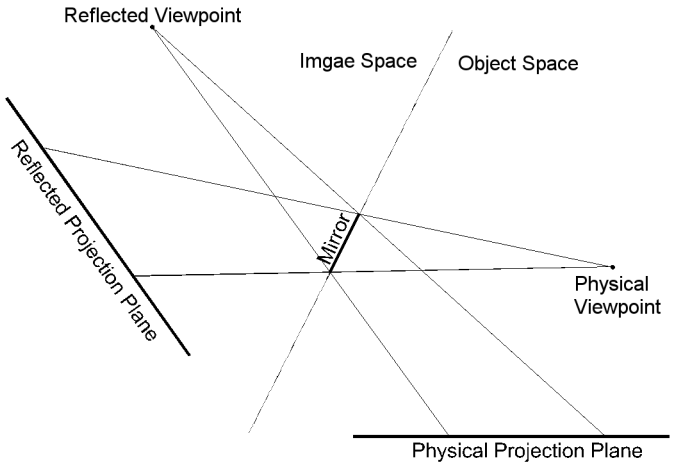


Figure 4.1. Perspective symmetry of the reflected view transform.

Note that the reflected view transform is similar to the virtual viewpoint methods, discussed in section 3.4. The difference, however, is that such methods generate images of a virtual scene from the reflected viewpoints to interactively simulate global illumination effects within the virtual scene itself (e.g., on virtual reflecting surfaces). Such methods apply a perspective on-axis projection for image generation. We, however, utilize real mirrors to reflect physical screens that stereoscopically display three-dimensional virtual scenes, applying perspective off-axis projection[4]. By using the reflected viewpoint as center of projection, the image of the scene on the screen appears as a reflection in the mirror and will be perceived three-dimensionally by the observer with the correct mirrored perspective. Thus, the real mirror reflects the virtual world and, in doing so, follows the same physical principles as in the real world.
Figure 4.2 illustrates how the model-view component of a transformation pipeline (see appendix B) can be modified to support the reflected view transformation. Instead of applying the view transformation (e.g., using gluLookAt) with the physical viewpoint ($\grave{e}$), the corresponding reflected viewpoint ($\grave{e}'$) is employed. The reflected viewpoint (as well as possible reflected virtual headlights) can be computed with equation 2.10 or by multiplying the physical viewpoint with the reflection matrix defined in equation 2.11.

4. Off-axis projections are discussed in appendix C.

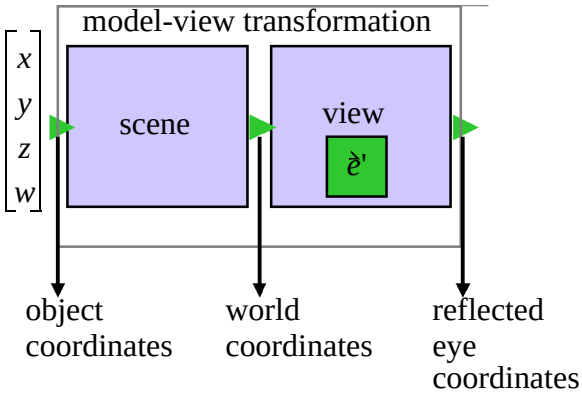


Figure 4.2: Modified model-view transformation- supporting the reflected view transform.

Given the algorithm for off-axis projection in appendix C, we can express the reflected view transform with: ProjectOffAxis(*è'*, *BB*, *width*, *height*).
To make use of the binocular parallax and to support stereoscopic projection, we have to compute the reflection of both viewpoints. This implies that ProjectOffAxis(*è'*, *BB*, *width*, *height*) is called twice - once for each eye.
Note that since planar mirrors are absolute optical systems, the reflected view transform is affine and can be integrated into standard transformation pipelines (see appendix B). Thus, no additional computational cost is required during rendering (i.e., the additional transformation cost is independent of the scene complexity) and the application of acceleration hardware is supported. The Reflective Pad is introduced as a proof-of-concept prototype for the reflected view transform in section 5.1.

4.1.2 Reflected Model-View Transform

Planar mirror beam-splitters (semi-transparent mirrors) enable us to optically combine stereoscopically projected 3D graphics with the real environment and to perceive both environments (real and virtual) in conjunction:
A planar mirror beam-splitter divides the environment into two subspaces (cf. figure 4.3).

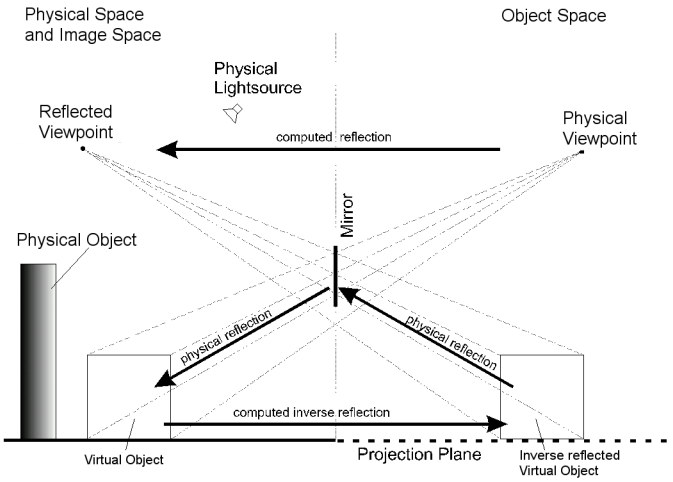


Figure 4.3: Perspective and geometric symmetry of the reflected model-view transform.

We call the subspace that contains the physical viewpoint and the projection plane (optically: real objects in form of illuminated pixels on the projection screen) the *object space* and the subspace that contains the physical objects and additional physical light sources the *physical space*. Note that from a geometric optics point of view, the physical space equals (or overlays) the mirror's *image space* (i.e., the space that appears behind the mirror by looking at it).
As in the immersive case, virtual objects that consist of graphical elements (such as geometry, normals, textures, clipping planes, virtual light sources, etc.) are defined within the 3D freespace (i.e., within the global coordinate system of the virtual environment). In our case, this coordinate system actually exceeds the boundaries of the projection space and extends into the surrounding physical space.
The challenge is to complement the physical objects located within the physical space with additional virtual augmentations. To achieve this, the graphical elements of the virtual objects are either defined directly within the physical space, or they are transformed to it during an object registration process. In both cases, graphical elements are virtually located within the physical space and, due to the lack of projection possibilities in the physical space, are not visible to the observer without additional aids.
We now consider the mirror and compute the reflection of the viewer's physical eye locations (as well as possible virtual headlights). We then apply the inverse reflection to every graphical element that is located within the physical space. In this manner these graphical elements are transformed and can be projected at their corresponding inverse reflected position within the object space. Thus, they are physically reflected back by the mirror into the mirror's image space. We refer to this as the *reflected model-view transform* [Bim00b].
When the setup is sufficiently calibrated, the physical space and the image space overlay exactly. The graphical elements appear in the same position within the image space as they would within the physical space without the mirror (if a projection possibility was given within the physical space).
As for the reflected view transform, the reflected viewpoint can be computed with equation 2.10 or by multiplying the physical viewpoint with the reflection matrix defined in equation 2.11. The inverse reflection of a virtual object that is located within the physical space is simply computed from its reflection with respect to the mirror plane. Since we assume that the physical space and the mirror's image space exactly overlay, we can also assume that the reflection of the graphical elements located within the physical space results in the inverse reflection of the image space, that is, they are transformed to their corresponding positions within the object space and can be displayed on the projection plane. Consequently, the additional model transformation (i.e., the inverse reflection of the scene) is achieved by multiplying the reflection matrix (equation 2.11) onto the current model-view matrix of the transformation pipeline (between scene transformation and view transformation). The modified transformation pipeline is illustrated in figure 4.4.
If we properly illuminate the physical objects located within the physical space, the mirror beam-splitter transmits their images. However, it also physically reflects the image of the inverse reflected graphical elements, which is projected within the object space. Thus both the transmitted image of the real objects and the reflection of the displayed graphics are simultaneously visible to the observer by looking at the mirror.

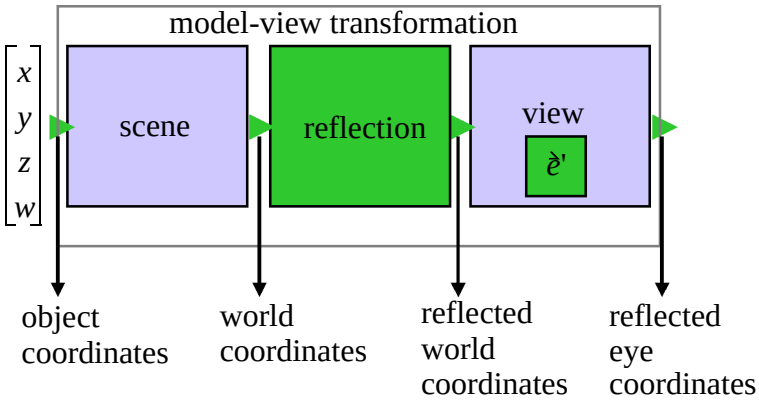


Figure 4.4: Modified model-view transformation- supporting the reflected model-view transform.

By applying the reflection matrix, every graphical element is reflected with respect to the mirror plane. A side effect of this is that the order of reflected polygons is also reversed (e.g., from counterclockwise to clockwise) which, due to the wrong front-face determination, results in a wrong rendering (e.g., lighting, culling, etc.). This can easily be solved by explicitly reversing the polygon order.
The following OpenGL pseudo code defines the reflected model-view transform:

ReflectedModelViewTransform($\grave{e}, a, b, c, d$ ***):***

1: R_l =BuildReflectionMatrix(a, b, c, d);

2: $\grave{e}' = R_l \grave{e}$;

3:ProjectOffAxis($\grave{e}'$, BB, $width$, $height$);

4:glMultMatrix(R_l);

5:glFrontFace(GL_CW+GL_CCW-glGetIntegerv(GL_FRONT_FACE));

Note that the order of the function-calls is reverse to the conceptual order of the algorithm. The reason for choosing this kind of notation is to remain consistent to the order of matrix multiplications, required by OpenGL.
To make use of the binocular parallax and to support stereoscopic projection, we again have to compute the reflection of both viewpoints. This implies that ReflectedModelViewTransform($\grave{e}, a, b, c, d$) is called twice - once for each eye.
Since in this case both -the view transformation and the model transformation- are affine, no additional computational cost is required during rendering (i.e., the additional transformation cost is again independent of the scene complexity). The Transflective Pad, the Extended Virtual Table and the Transflective Board are introduced as proof-of-concept prototypes for the reflected model-view transform in sections 5.2 - 5.4.

4.1.3 Refracted Model Transform

Mirror beam-splitters that are built from thick plates of glass cause optical distortion that results from refraction.
With respect to figure 4.3: All scene vertices that are registered to the physical space are virtual points that are not physically located behind the mirror, and consequently are not physically refracted by the glass, but are reflected by the front surface mirror. Since the transmitted light which is emitted by the physical objects and perceived by looking through the mirror is

refracted, but the light that is reflected by the front surface mirror is not, the transmitted image of the physical space cannot be precisely registered to the image space (i.e., the reflected object space), even if their geometry and alignment match exactly within our world coordinate system.

All optical systems that use any kind of see-through element have to deal with similar problems. While for HMDs, aberrations caused by refraction of the lenses are mostly assumed to be static[5] (as stated by Azuma [Azu97]), they can be corrected with paraxial analysis approaches. For other setups, such as the reach-in systems that were mentioned in chapter 3 or other spatially aligned mirror beam-splitters, aberrations caused by refraction are dynamic, since the optical distortion changes with a moving viewpoint. Wiegand et al [Wie99], for instance, estimated the displacement caused by refraction for their setup to be less than 1.5 mm, predominantly in +y-direction of their coordinate system. While an estimation of a constant refraction might be sufficient for their apparatus (i.e., a near-field virtual environment system with a fixed viewpoint that applies a relatively thin (3 mm) mirror), some PBAR setups require a more precise definition, because they are not near-field VE systems but rather mid-field VR/AR systems, they consider a head-tracked viewpoint, and they might apply a relatively thick beam-splitter.

Since we cannot pre-distort the refracted transmitted image of the physical space, we artificially refract the virtual scene within the image space instead, in order to make both images match. However, since refraction is a complex curvilinear transformation and does not yield a stigmatic mapping in any case, we can only approximate it.

A simple solution is the assumption that, against its optical nature, refraction can be expressed as an affine transformation. For beam-tracing, Heckbert [Hec84] assumes that, considering only paraxial rays, objects seen through a polygon with the refraction index η appear to be η times their actual distance. For this approximation, he does not take the incidence angles of the optical lines of sight into account but, instead, assumes a constant incidence angle of $\alpha_i = 0$ that is define by the optical axis which is perpendicular to the refracting polygon. The following algorithm illustrates Heckbert's approach within an OpenGL context:

RefractedModelTransform(a, b, c, d, η ***):***

1: R_f =BuildRefractionMatrix(a, b, c, d, η);

2:glMultMatrix(R_f);

The refraction matrix R_f is given by equation A1.3. Consequently, Heckbert performs an affine mode-view transformation (by either transforming the model or the viewpoint[6]) which is only a valid approximation for on-axis situations and for refractors that do not provide a wide field of view. Heckbert does not address in-out refractions, but only in-refractions or out-refractions.

A better way of approximating refraction is to take off-axis situations into account. Since Heckbert's approach is a special case of our refraction method (as shown in appendix A), we can extend Heckbert's affine mapping towards off-axis transformations and in-out refractions:

5. We do not consider rotations of the eyeballs.
6. For on-axis projections, moving the model in one direction is equivalent to moving the viewpoint in the opposite direction.

RefractedModelTransform($\vec{r}$, a, b, c, d, t, η ***):***

1: Δ =ComputeDelta($\vec{r}$, a, b, c, d, t, η);

2: R_f =BuildRefractionMatrix(a, b, c, d, Δ);

3:glMultMatrix(R_f);

In this case, the incidence angle of a dynamically changing line of sight $\vec{r}$ is taken into account, rather than considering only paraxial rays. In this case, $\vec{r}$ is the viewing direction, determined by a head-tracker. The transformation offset Δ can be either computed using equation 2.13 for in-out refractions considering the material's thickness t and its refraction index η, or by using equation 2.18 for in-refractions or out-refractions. The refraction matrix R_f is given by equation 2.16. Although this approach uses a single representative line of sight to determine the refraction transformation for off-axis situations more exactly, it does not consider the curvilinearity of refraction and still assumes an affine refraction mapping.
To approximate the curvilinearity of refraction more precisely, we have to apply individual transformations for different lines of sight:

RefractedModelTransform($\vec{e}$, a, b, c, d, t, η ***):***

1:**forall** model vertices $\vec{p}$

2: $\vec{r} = \vec{p} - \vec{e}$;

3: Δ =ComputeDelta($\vec{r}$, a, b, c, d, t, η);

4: R_f =BuildRefractionMatrix(a, b, c, d, Δ);

5: $\vec{p}' = R_f \cdot \vec{p}$

6:**endfor**

This algorithm illustrates a per-vertex transformation to simulate refraction. An individual line of sight from the current viewpoint to each scene vertex is determined. Since each line of sight yields a different incidence angle, variable transformation offsets are computed for each vertex. The transformation offsets, in turn, generate different refraction transformations and consequently vertex-individual transformations. As above, the refraction matrix R_f is given by equation 2.16, and Δ can be computed using equation 2.13 or equation 2.18.
The resulting per-vertex transformation is curvilinear rather than affine, thus a common transformation matrix cannot be applied anymore.
The general idea of a per-vertex transformation for optical pre-distortion is similar to Rolland's and Hopkins' approach for the correction of optical distortion that is caused by the lenses integrated into head-mounted displays [Rol93]. However, they pre-compute the static distortion of the HMD's centered on-axis optics and modify vertices of virtual objects' polygons that are projected onto the image plane, before rendering. Note that a per-vertex transformation (either on the image plane or within the 3D free-space) requires subdividing polygons which cover large areas to sufficiently express the curvilinear mapping. This is also the case for our approach.
As discussed in section 2.4.1, the transformation offset Δ that is computed for each point $\vec{p}$ is not the correct offset of this point, but a close approximation. Rather than that, the Δ used is the transformation offset that belongs to the point $\vec{p}''$ which visually appears at the spatial position of $\vec{p}$ after being refracted. However, since the angular difference between the geometric

lines of sight spanned by $\dot{p} - \dot{e}$ and $\dot{p}'' - \dot{e}$ is small, the arising error can be disregarded. This is described below in more detail and will be evaluated in section 6.1.

The methods that have been discussed so far consider only a single viewpoint. For binocular vision, however, we need to take both eyes into account. Since both viewpoints differ, they generate different lines of sight and consequently cause individual refraction transformations. This could be realized by applying the above model transformation twice -before the generation of each stereo image. But because of its high computational cost and its dependency on the scene complexity, we propose an alternative approximation:

As described in section 2.5.2, for binocular vision the observer's eyes $\dot{e}_1, \dot{e}_2$ have to converge to see a point in space $\dot{p}'$ so that the geometric lines of sight (colored in black in figure 4.5) intersect in $\dot{p}'$. If the observer sees through a planar lens, the geometric lines of sight are bent by the lens and she perceives the point in space $\dot{p}$ where the resulting optical lines of sight (colored in red in figure 4.5) intersect - i.e., she visually perceives $\dot{p}$ at the spatial position of $\dot{p}'$ if refraction bends her geometric lines of sight.

To artificially refract the virtual scene, our goal is to translate every scene vertex $\dot{p}$ to its corresponding point $\dot{p}'$ -following the physical rules of refraction. Note again, that for mirror beamsplitters all points $\dot{p}$ are virtual points that are not physically located behind the mirror, and consequently are not physically refracted by the plate of glass, but are reflected by the front surface mirror. The resulting transformation is curvilinear rather than affine. Thus, a common transformation matrix cannot be applied.

For each eye, we can compute an optical line of sight for a corresponding geometric line of sight. Note that an optical line of sight is the refractor that results from the geometric line of sight which is spanned by the viewer's eye and the point in space $\dot{p}$ at which she is looking.

For in-out refractions at planar lenses, the optical lines of sight can be computed as follows:

$$\dot{r}'_1 = (\dot{e}_1 - \Delta_2 \dot{n}) + \lambda \dot{r}_1, \; \dot{r}'_2 = (\dot{e}_2 - \Delta_2 \dot{n}) + \lambda \dot{r}_2 \qquad (4.1)$$

Note that to compute in-refractors or out-refractors only, we can apply equations 2.17 and 2.18, or equation 2.21 to determine the optical line of sight for a given geometric line of sight.

Since no analytical correction method exists, we can apply a numerical minimization to precisely refract virtual objects that are located behind the mirror by transforming their vertices within our world coordinate system. As already mentioned, similar to Rolland's approach [Rol93] our method also requires subdividing large polygons of virtual objects to sufficiently express the refraction's curvilinearity.

Our goal is to find the coordinate $\dot{p}'$ to which the virtual vertex $\dot{p}$ has to be translated so that $\dot{p}$ appears spatially at the same position as it would appear as real point, observed through the mirror - i.e., refracted. To find $\dot{p}'$, we first compute the geometric lines of sight from each eye $\dot{e}_1, \dot{e}_2$ to $\dot{p}$. We then compute the two corresponding optical lines of sight and determine their intersection $\dot{p}''$ (e.g., using equation 5.7 to compute the intersection or the closest point between two straight lines).

During a minimization procedure (e.g., using Powell's direction set method [Pre92]) we minimize the distance between $\dot{p}$ and $\dot{p}''$ while continuously changing the angles (α, β) (simulating the eyes' side-to-side shifts and convergence) and γ (simulating the eyes' up-and-down movements), and use them to rotate our geometric lines of sight over the eyes' horizontal and

vertical axes that can be determined from the head-tracker. The rotated geometric lines of sight result in new optical lines of sight and consequently in a new $\grave{p}''$.
Finally, $\grave{p}$ is the intersection of the (by some α, β, γ) rotated geometric lines of sight where $|\grave{p} - p''|$ is minimal (i.e., below some threshold ε). This final state is illustrated in figure 4.5 left.

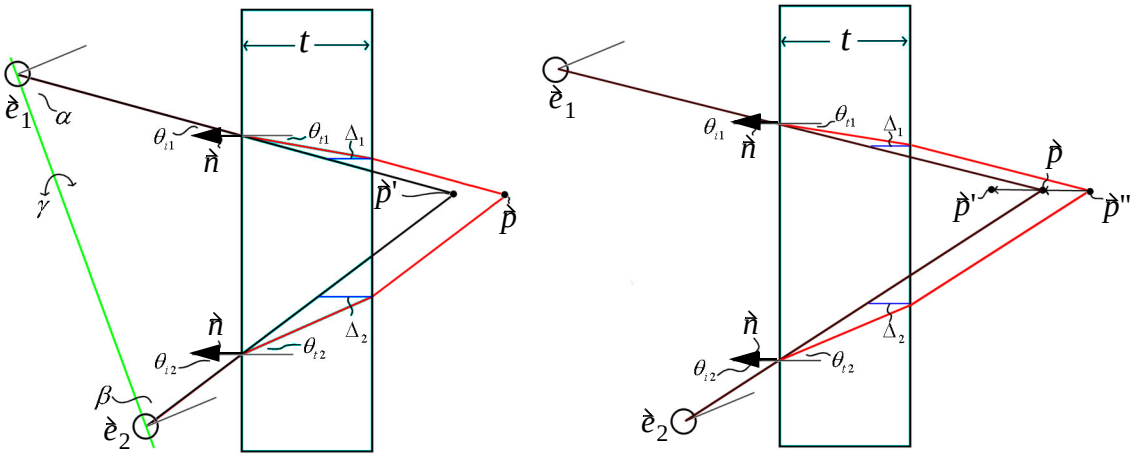


Figure 4.5: Refracted model transform approximation for planar lenses.

In summary, we have to find the geometric lines of sight whose refractors (i.e., the corresponding optical lines of sight) intersect in $\grave{p}$ and then calculate the precise coordinates of $\grave{p}'$ as intersections of the determined geometric lines of sight. Since $\grave{p}'$ is unknown, the resulting minimization problem is computationally expensive and cannot be solved in real-time.
To achieve a high performance on an interactive level, we implemented an analytical approximation of the presented numerical method.
With reference to figure 4.5-right: We compute the refractors of the geometric lines of sight to the vertex $\grave{p}$ and their intersection $\grave{p}''$. Since the angular difference between the unknown geometric lines of sight to the unknown $\grave{p}'$ and the geometric lines of sight to $\grave{p}$ is small, the deviations of the corresponding refractors are also small. We approximate $\grave{p}'$ with $\grave{p}' = \grave{p} + (\grave{p} - \grave{p}'')$ [Bim01e].

RefractedModelTransform($\grave{e}_1, \grave{e}_2, a, b, c, d, t, \eta$ ***):***

1:**forall** model vertices $\grave{p}$
2: $\grave{r}_1 = \grave{p} - \grave{e}_1$;
3: $\grave{r}_2 = \grave{p} - \grave{e}_2$;
4: $\grave{r}'_1$ =ComputeOpticalLineOfSight($\grave{r}_1, a, b, c, d, t, \eta$);
5: $\grave{r}'_2$ =ComputeOpticalLineOfSight($\grave{r}_2, a, b, c, d, t, \eta$);
6: p'' =Intersect($\grave{r}'_1, \grave{r}'_2$);
7: $\grave{p}' = \grave{p} + (\grave{p} - \grave{p}'')$
8:**endfor**

An error analysis of this analytical approximation, compared to the results of the slow but precise numerical method, is presented in section 6.1.

Figure 4.6 illustrates where the *refracted model transform* is carried out within a transformation pipeline: between the regular scene transformation and the reflected model-view transform.

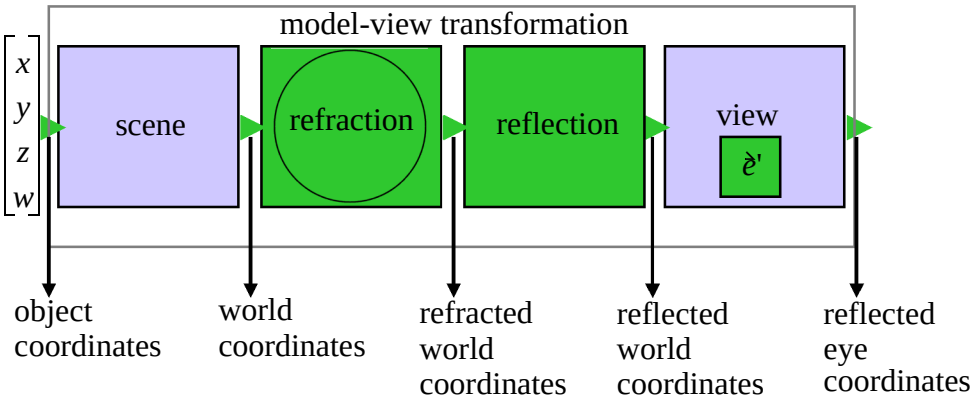


Figure 4.6: Modified model-view transformation - supporting the refracted model transform and reflected model-view transforms.

The circle in the refraction transformation module denotes a possible per-vertex operation (i.e., a vertex individual transformation), while the modules without a circle denote global matrix operations (i.e., affine transformations).
Note that depending on the degree of approximation, the refracted model transform can be either an affine model-view transformation matrix, or a per-vertex operation applied to a tessellated scene geometry. The Extended Virtual Table serves as a proof-of-concept prototype for the refracted model transform in section 5.3.

4.1.4 Projected Image Transform

Projectors that beam graphics onto projection planes can be miscalibrated and can consequently display geometrically distorted images.
Projector-specific parameters (such as geometry, focus, and convergence) are usually adjusted manually or automatically using camera-based calibration devices. While a precise manual calibration is very time consuming, an automatic calibration is normally imprecise, and most systems do not offer a geometry calibration (only calibration routines for convergence and focus).
For exclusive VR purposes, however, we can make use of the fact that small geometric deviations are ignored by the human-visual system. In AR scenarios, on the other hand, even slight misregistrations can be sensed.
To precisely and easily calibrate the projector's geometry, we apply a two-pass method and render a regular planar grid U that largely fits the projection plane [Bim01e]. The distorted displayed grid geometry has to be sampled on the table top with a precise 2D positioning system. After a transformation of the sampled grid D from the device coordinate system of the positioning device into our world coordinate system, it can be used to pre-distort the projected image. Since D contains the projector's geometric distortion, it can be used to determine the resulting geometric deviation $U - D$. A pre-distorted grid P can then be computed with $P = U + (U - D)$. If we finally project P instead of U, the pre-distortion is neutralized by the physical distortion of the projector, and the visible grid appears undistorted.
To pre-distort the projected images, however, we first render the virtual scene into the frame-buffer, then map the frame-buffer's content as texture onto P (while retaining the texture indi-

ces of U and applying a bilinear texture-filter), and render P into the beforehand cleaned frame-buffer, as described by Watson and Hodges [Wat95] for HMDs. Note that this is done for both stereo-images at each frame.

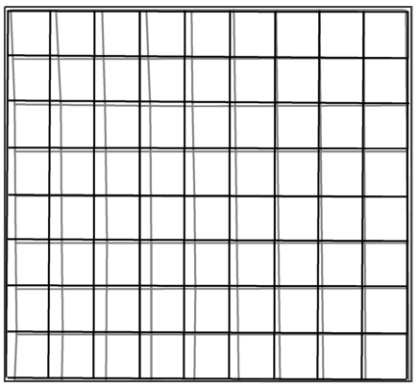


Figure 4.7: Sampled distorted grid (grey) and pre-distorted grid (black) after projection and re-sampling.

Figure 4.7 illustrates the sampled distorted grid D (grey), and the pre-distorted grid P (black) after it has been rendered and re-sampled. Note that figure 4.7 shows real data from one of our calibration experiments.

Since optical distortion that is introduced by miscalibrated projectors is static[7], the calibration procedure has to be carried out once or once in a while, since the distortion behavior of the projector can change over time.

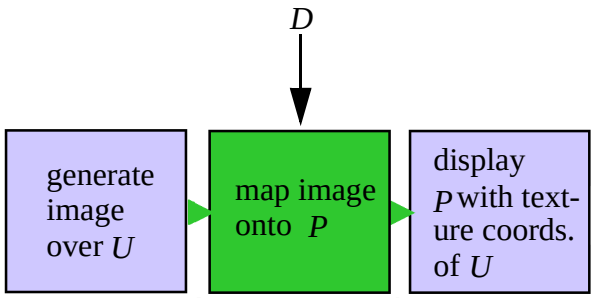


Figure 4.8: Overview of the projected image transform's two-pass method.

Figure 4.8 gives an overview of the *projected image transform's* two-pass method. The first rendering pass generates an image which is the result of sending the scene's geometry through the entire rendering pipeline. To support PBAR configurations, a modified transformation pipeline is applied during the first pass to support reflection and refraction transformations (as discussed in the previous sections). The resulting image is mapped onto the pre-distorted grid geometry P which is finally displayed during the second rendering pass while remaining the undistorted grid's texture coordinates. The Extended Virtual Table serves as a proof-of-concept prototype for the projected image transform in section 5.3

7. Not taking into account the time varying distortions caused by warming up of the projector.

4.1.5 Convex Multi-Section Optics

In this subsection we assume that the applied optics is assembled form multiple planar sections which form a convex construct, rather than from a single planar optical element. The pyramid-like Virtual Showcase, described in section 5.5.3 serves as a proof-of-concept prototype for the following rendering techniques (cf. figure 4.9).

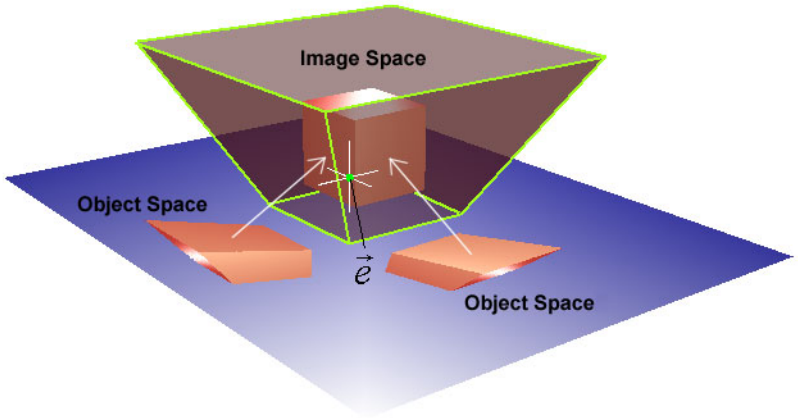


Figure 4.9: A convex multi-section optical system: Depending on the viewpoint, individual images are rendered within the object space for each front-facing mirror plane. Their reflections merge into a single consistent image space.

To transform the known *image space geometry* (i.e., the geometry that has been defined within the image space) appropriately into the object space, we can apply the previously discussed transformations multiple times -e.g., within a multi-pipeline configuration [Bim01d]. Figure 4.10 presents an example.

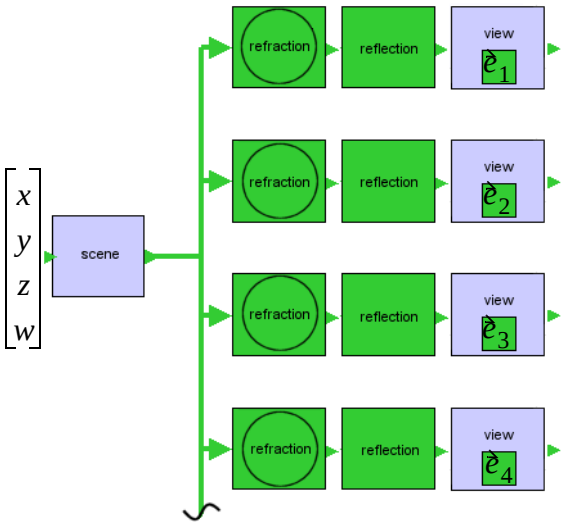


Figure 4.10: *The refracted model transform and the reflected model-view transform within a multi-pipeline configuration: This example assumes that four viewers look at individual mirror elements -seeing the same image space simultaneously from different perspectives.*

Each subpipeline uses different parameters: Since individual plane parameters exist for each mirror element, different refracted model transforms and reflected model-view transforms (with individual reflection matrices and reflected viewpoints) have to be applied for each front-facing mirror element, respectively. Thus, for a given viewpoint $\grave{e}$, the image space geometry is transformed and rendered multiple times (for each front-facing mirror element individually).
Since convex mirror assemblies unequivocally tessellate the object space into mirror individual reflection zones which do not intersect or overlap, a single object that is displayed within the object space appears exactly once within the image space. Consequently, a definite one-to-one mapping between the object space and the image space is provided by convex mirror assemblies.
Observed from $\grave{e}$, the different images that are projected into the object space optically merge into a single consistent image space by reflecting the projection plane[8], whereby this image space visually equals the image of the defined image space geometry.
For instance, individual views onto the same image space (seen from different perspectives) can either be perceived by a single viewer while moving around the optics, or by multiple viewers, while looking at different mirror elements simultaneously. While for the first case, the same viewpoint $\grave{e}$ is used for each subpipeline, individual viewpoints (e.g., $\grave{e}_1$, $\grave{e}_2$, $\grave{e}_3$, $\grave{e}_4$) have to be applied for the second case. Both cases are illustrated in figures 5.26 and 5.27.
Note that in terms of generating stereo images, all transformation and rendering steps have to be applied individually for each eye of each viewer. This means that for serving four viewers simultaneously, for instance, the transformation pipeline is split into four subpipelines after a common scene transformation. Following the application of the mirror-specific reflection transformations, the subpipelines have to be split again to generate the different stereo images for each eye[9]. The subsequent eight sub-pipelines use different viewpoint transformations with individually reflected viewpoints, corresponding to each eye-position.
Diverging from the example presented in figure 4.10, individual scenes (i.e., different image spaces) can be presented to each viewer. In this case, an individual scene transformation has to be applied within each subpipeline. A static mirror-viewer assignment is not required, but individual mirror sections can be dynamically assigned to moving viewers. In case multiple viewers look at the same mirror, an average viewpoint can be computed (this results in slight perspective distortions).
Note that due to the independence among the subpipelines, the application of parallel rendering techniques (e.g., using multi-pipeline architectures) is obvious. If the curvilinear refracted model transform is not applied, the modified transformation pipeline remains affine and does not require access to the image space geometry. Thus, it can be realized completely independent of the application, and can even be implemented in hardware.
To provide a single consistent image space (consistent, regardless of the viewpoint), the planes of the mirror elements have to be registered very precisely. Slight misregistrations optically result in gaps between the reflections, and therefore in multiple inconsistent image spaces. This problem is comparable to misregistered multi-plane projection devices or tiled displays, although for mirror planes the optical distortion view-dependent and dynamically changes with a moving viewpoint.

8. The projection plane is colored blue.
9. This is not illustrated in figure 4.10.

4.2 Curved Optics

If the single elements of such assemblies, described in section 4.1.5 are made infinitely small, we can say that the optics is curved, rather than built from multiple planar sections. Approximating an appropriate transformation with a finite number of subpipelines, however, is inefficient for curved optics.
Compared to calibrating multiple optical elements, the calibration effort for a single element can be reduced and optical aberrations that result from miscalibration can be decreased by several orders of magnitude. However, curved optical elements lead to new problems. Curved mirrors, for instance, reveal the following difficulties (with reference to figure 4.11):

- The transformation of the image space geometry into the object space $\vec{v} \rightarrow \vec{v}'$ is not affine but curvilinear;
- The transformation of the image space geometry depends on the viewpoint (i.e., the image space geometry transforms differently for different viewpoints);
- The viewpoint transformations $\vec{e} \rightarrow \vec{e}'$ depend on the image space geometry (i.e., each vertex $\vec{v}$ within the image space yields an individual $\vec{e}'$).

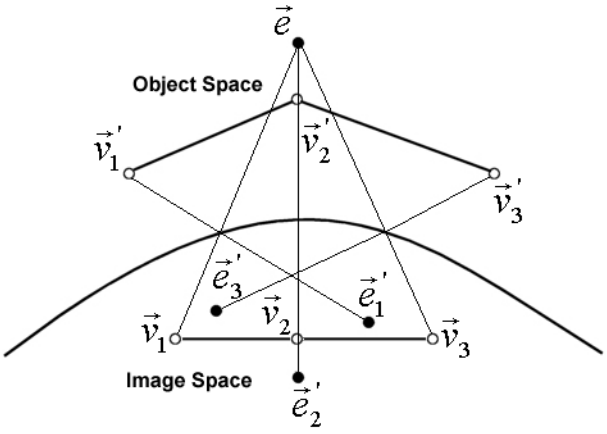


Figure 4.11: Curved mirrors require curvilinear transformations of the image space geometry into the object space. The geometry transformations depend on the viewpoint, and the viewpoint-transformations depend on the geometry.

As mentioned in section 4.1.5, convex mirror assemblies unequivocally tessellate the object space into mirror individual reflection zones which do not intersect or overlap and consequently provide a definite one-to-one mapping between object space and image space. This is also true for curved convex mirrors. In chapter 2, we said that curved convex mirrors cannot provide true stigmatism between all object-image pairs, but rather a close approximation which introduces small optical aberrations. Since due to the limited resolution of the eyes, human observers can usually not perceive these aberrations, we want to disregard them for the subsequent sections.
As described in section 2.3.2, images formed by convex mirrors appear to be reduced and deformed versions of the reflected objects. Hence, the known image space geometry has to be stretched before displaying it within the object space to result in the original (unscaled) image space after the physical reflection.
Remember, that the refraction transformation is always curvilinear -regardless of the type of refractor (i.e., planar or curved), and that the reflection transformation for planar mirrors is affine. To map the image space geometry appropriately into the object space, curved mirrors

also require per-vertex viewpoint and model transformations. However, as already mentioned while discussing the refraction transformations, only highly tessellated image space geometries can be transformed with such methods to provide an acceptable curvilinearity deformation behavior.
Curved mirror displays, for instance, that stereoscopically produce three-dimensional images generally don't pre-distort the graphics before they are displayed. Yet, some systems apply additional optics (such as lenses) to stretch or undistort the reflected image (e.g. [Mck99a, Mck99b]). But these devices constrain the observer to a single point of view or to very restricted viewing zones. However, if a view-dependent rendering is required to support freely moving observers, interactive rendering and real-time image warping techniques are needed which provide appropriate error metrics.
We have developed several non-affine geometry and image transformation (so-called *warping*) techniques for curved mirrors, which are discussed within the following subsections.
The cone-like Virtual Showcase, described in section 5.5.4 serves as a proof-of-concept prototype for the following rendering techniques.

4.2.1 A Geometry-Based Approach

Our initial geometry-based approach transforms the tessellated scene geometry explicitly. Thus, the transformation parameters are computed individually for each vertex. These transformations depend on the intersections of the lines of sight (spanned by the viewpoint and the scene vertices) with the optics.

4.2.1.1 Geometry-Based Reflected Model-View Transform

The GeometryBasedReflectedModelViewTransform algorithm outlines our initial approach to map the image space geometry into the object space with respect to a curved mirror optics (cf. figure 4.12). Note that this approach is similar in spirit to Ofek's virtual object method [Ofe98, Ofe99]. But rather than applying a geometry-based reflection transform, we apply a vertex-individual reflected model-view transform and a subsequent projection.

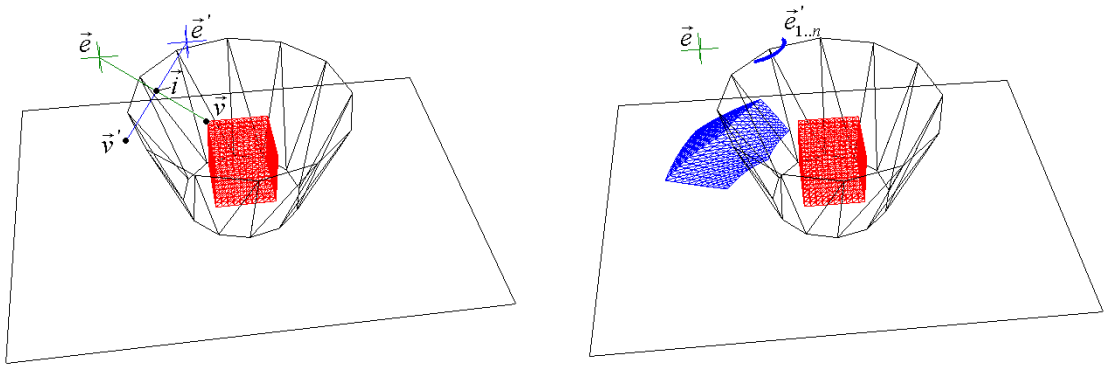


Figure 4.12: Geometry-based reflected model-view transform: Each vertex within the image space (vertices of red cube) require an individual transformation that depends on the viewpoint. A curved image within the object space (blue cube) is the result of the transformations.

GeometryBasedReflectedModelViewTransform($\grave{e}$ **,** $T_{0\ldots n}$ **,** Σ ***):***

1:**forall** geometry vertices $\grave{v}_k$ of $T_{0\ldots n}$

2: $\grave{v}_k$ is not visible

3:**endfor**

4:**forall** front-facing geometry triangles T_j
5: **forall** triangle vertices of T_j, $\vec{v}_{jk}$
6: **if** $\vec{v}_{jk}$ is not visible
7: **if** front-facing intersection $\vec{i}$ of $\vec{r} = \vec{e} + (\vec{v}_{jk} - \vec{e})$ with mirror Σ
8: compute normal $\vec{n}$ at $\vec{i}$
9: compute triangle plane $[a, b, c, d]$ *at* $\vec{i}$
10: build reflection matrix R_l from $[a, b, c, d]$
11: compute reflected viewpoint $\vec{e}' = R_l \cdot \vec{e}$
12: **begin** transform $\vec{v}_{jk}$
13: $\vec{v}'_{jk} = P \cdot V \cdot R_l \cdot S \cdot \vec{v}_{jk}$ (off-axis perspective projection with $\vec{e}'$)
14: perspective division $\vec{v}''_{jk} = \frac{\vec{v}'_{jk}}{w}$
15: viewport transformation in device coordinates, e.g.:

$$\vec{v}'''_{jkx} = (\vec{v}''_{jkx} + 1)\frac{dw}{2} - \frac{dw}{2},\ \vec{v}'''_{jky} = (\vec{v}''_{jky} + 1)\frac{dh}{2} - \frac{dh}{2},$$

$$\vec{v}'''_{jkz} = \vec{v}''_{jkz} \cdot \varepsilon$$ (ε used to preserve depth relations)

16: **end**
17: $\vec{v}_{jk}$ is visible
18: **endif**
19: **endif**
20: **endfor**
21:**endfor**

As for the planar case, the geometry of the curved mirror and the viewpoint have to be known. Furthermore, the image space geometry has to be known too, and has to be tessellated to support a smooth curvilinear transformation into the object space. In the example illustrated in figure 4.12, the cube (which normally consists of 8 vertices) has been tessellated into 600 vertices.

Note that our internal representation of the image space geometry is an indexed triangle set $T_{0\ldots n}$. For performance reasons, we make sure that a transformation is computed only once for each vertex. Therefore, lines 1-3 initialize a visibility flag for each vertex. Only non-visible vertices are transformed. Once transformed, a vertex is defined as visible (line 17).

For all vertices of front-facing triangles, the intersection of the geometric line of sight (i.e., the ray that is spanned by the eye and the vertex) with the mirror geometry is computed first (line 7, green line in figure 4.12-left). Next, the normal vector at the intersection has to be determined (line 8). The intersection point, together with the normal vector, gives the tangential plane at the intersection. Thus, they deliver the plane parameters for the vertex individual reflection transformation (lines 9-10). Note that an intersection is not given if the viewpoint and the vertex are located on the same side of a tangential plane.

Using the reflection matrix, the reflected viewpoint is computed first (line 11). Finally, the vertex is transformed and projected (lines 12-16). Here, P denotes the perspective projection transformation matrix, V the view transformation matrix, and S the scene transformation matrix for an off-axis projection (see appendix B).

Doing this for all front-facing image space vertices, results in the projected image within the object space (blue cube in figure 4.12-right). The *geometry-based reflected model-view transform* can be thought of as a modified per-vertex transformation pipeline, and we can illustrate it by combining figure 4.6 and figure B.1, whereby all components are carried out on a per-vertex basis (i.e., they have to be computed individually for each vertex).
Note that standard graphic pipelines (such as the one implemented with the Open GL package) only support primitive-based transformations and not per-vertex transformations. Thus, the transformation pipeline used for this approach has been re-implemented explicitly -bypassing the OpenGL pipeline. However, to preserve valid values for z-buffering while the object space is rendered, the z-coordinates of the transformed vertices are not set to zero. They are rather multiplied by some constant (e.g., ε=0.0001, with respect to the z-buffer threshold), after the perspective division. The viewport transformation is computed within the same world coordinate system in which the projection device is embedded, rather than in window coordinates. Here, dw and dh denote the device width and height in world coordinates.
Having a geometric representation to approximate the mirror's shape (e.g., a triangle mesh, as illustrated in figure 4.12) supports a flexible way of describing the mirror's dimensions. However, the computational cost of the per-vertex transformations increases with a higher resolution mirror geometry. For triangle meshes, a fast ray-triangle intersection method (such as [Moe97]) is required that automatically delivers the barycentric coordinates of the intersection within a triangle. The barycentric coordinates can then be used to interpolate between the three vertex normals of a triangle to approximate the normal vector at the intersection.
A more efficient way of describing the mirror's dimensions is to apply an explicit function. This function can be used to calculate the intersections and the normal vectors (using its 1st order derivatives) with an unlimited resolution. However, not all shapes can be expressed by explicit functions. For instance, the explicit function for our cone-like mirror, illustrated in figure 4.12 and its 1st order derivatives are:

$$f(x,y,z) = \frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} = 0\,,\ \vec{n} = \left[\frac{f}{\delta x}, \frac{f}{\delta y}, \frac{f}{\delta z}\right] = 2\left[\frac{x}{a^2}, \frac{y}{b^2}, -\frac{z}{c^2}\right] \tag{4.2}$$

where a, b are the cone's radii with its center located at the world-coordinate-system's origin, and c is the cone's height along the z-axis.
After a geometric line of sight has been transformed from the world-coordinate-system into the cone-coordinate-system, it can be intersected easily with the cone by solving a quadratic equation created by inserting a parametric ray representation into the cone equation.
Obviously, we have the choice between a numerical and an analytical approach, in some mirror cases. If an analytical solution is given, it should be preferred over the numerical variant. Higher order curved mirrors, however, require the application of numerical approximations.
So far, we have seen how to describe the curvilinear transformation of the image space vertices into projected object space vertices. Per-vertex properties (such as normal vectors, lighting and texture information, alpha values, etc.) have not been considered.
For the reflected view transform described in section 4.1.1, the reflected model-view transform described in section 4.1.2 and for all virtual viewpoint methods described in chapter 3, the virtual light sources, for instance, are simply reflected as a part of the virtual scene, and consequently support a correct shading of the reflection. In addition, the shading computations are completely hardware supported in these case.

For curved mirrors, Ofek [Ofe98, Ofe99] computes the shading values for the unreflected vertices within the world coordinate system, and then assigns these values to the reflected vertices. This is inefficient, since the shading computations have to be done explicitly in software.
A model-view reflected vertex does have the same properties as its corresponding unreflected vertex. In fact, this is our initial aim -namely to generate an object space from a known image space that, if physically transformed by the mirror optics, results in exactly the same image space. Hence, computing scene properties for each unreflected vertex and then using them for their reflections (as proposed by Ofek) is a correct way for generating images that are displayed with PBAR configurations. However, it is still inefficient, since it has to be done explicitly (i.e., in software). In addition, this approach would cover the shading computations only -still not considering the other per-vertex and surface properties, such as transparency, etc.
A performance evaluation of the geometry-based reflected model-view transform in combination with explicit per-vertex shading is presented in section 6.3.1.

4.2.1.2 Transforming all Surface Properties

To efficiently map all surface properties from the image space into the object space (i.e., not only geometry and shading properties), we have implemented a two-pass rendering method.
The first rendering pass creates a picture of the image space and renders it into the texture memory[10], rather than into the frame-buffer (cf. figure 4.13-left). This is outlined by the GenerateImage algorithm, that requires the viewpoint $\grave{e}$, the scene's bounding sphere parameters (position: $\grave{p}$, radius: θ) with respect to the current scene transformations S, and the desired image resolution (tw and th).

GenerateImage($\grave{e}$, $\grave{p}$, θ, tw, th ***):***
1: **begin** compose GL transformations pipeline
(on-axis perspective projection with $\grave{e}$)
2: $\lambda = |\grave{p} - \grave{e}|$, $\delta = \frac{\theta(\lambda - \theta)}{\lambda}$
3: $left = -\delta$, $right = \delta$, $bottom = -\delta$,
$top = \delta$, $near = \lambda - \theta$, $far = \lambda + \theta$
4: set projection transformation:
glFrustum($left$,$right$,$bottom$,top,$near$,far)
5: set view transformation:
gluLookAt($\grave{e}_x$, $\grave{e}_y$, $\grave{e}_z$, $\grave{p}_x$, $\grave{p}_y$, $\grave{p}_z$, 0, 0, 1)
6: set view port transformation:
glViewPort(0, 0, tw,th)
7: **end**
8: apply scene transformations S to scene and
render scene into texture memory

To generate this image, an on-axis projection is carried out. The size of the projection's viewing frustum is determined from the image space's bounding sphere (lines 2-3). After all necessary transformations (i.e., projection, view and view-port) have been set up (lines 4-6), the image is finally rendered into the texture-buffer (line 8). Here, tw,th are the texture width and

10. Into the texture buffer (if supported) or into allocated memory blocks.

height (i.e., the desired image resolution). For instance, we can choose these parameters to be: $th = tw = i^2$, with i being an arbitrary value that is limited by the applied rendering hardware. Note that the texture size does not necessarily have to be a power of two. In an OpenGL context, however, this ensures that the allocated texture memory is fully exploited.
A texture matrix is defined as outlined by the SetTextureMatrix algorithm -using the parameters, computed for the image generation.

SetTextureMatrix($\grave{e}$ **,** $\grave{p}$ **,** *left*,*right*,*bottom*,*top*,*near*,*far* ***):***

1:**begin** compose texture matrix (on-axis perspective projection with $\grave{e}$)
2: set texture normalization correction:
glScale(0.5,0.5,0.5), glTranslate(1,1,0)
3: set projection transformation:
glFrustum(*left*,*right*,*bottom*,*top*,*near*,*far*)
4: set view transformation:
gluLookAt($\grave{e}_x$, $\grave{e}_y$, $\grave{e}_z$, $\grave{p}_x$, $\grave{p}_y$, $\grave{p}_z$, 0, 0, 1)
5:**end**

The resulting texture matrix applies almost the same projection transformations to texture coordinates, as the transformation matrix generated during image generation to the scene geometry. The only difference is the texture normalization (line 2) that provides a mapping from normalized device coordinates (i.e., $[-1,1]$ for OpenGL) to normalized texture coordinates (i.e., $[0, 1]$ for OpenGL).
The texture-buffer content is mapped onto the transformed image space vertices (generated with GeometryBasedReflectedModelViewTransform), which is then rendered during the second rendering pass (cf. figure 4.13-right).
To compute the correct texture coordinates for each transformed vertex within the object space, the texture matrix that has been pre-computed is applied. To do so, the untransformed image space vertices serve as texture coordinates which, multiplied by the texture matrix, result in the correct texture coordinates for each.
The second rendering pass does not require illumination computations, since lighting information (as well as all other per-vertex properties) are generated during the first rendering pass. In addition, back-facing polygons have to be culled and the polygon order has to be reversed, as discussed in section 4.1.2.
Since a primitive-based (or fragment-based) antialiasing does not apply in case of a transformed texture, bi-linear or tri-linear texture filters can be applied instead. As antialiasing, texture filtering is usually supported by the graphics hardware.

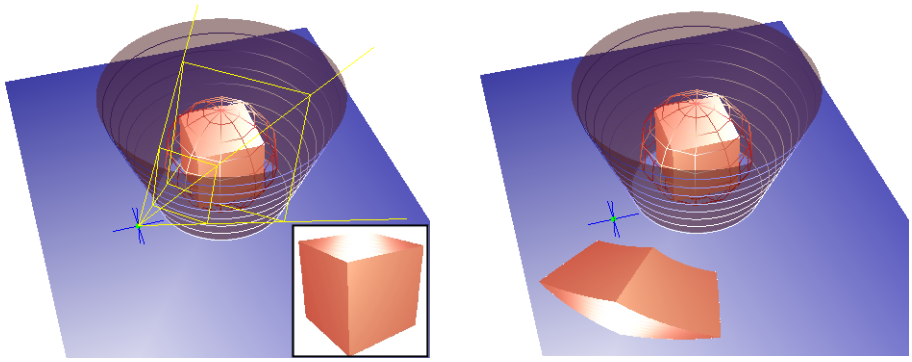


Figure 4.13: Geometry-based multi-pass approach: The first rendering pass (left) generates a picture of the scene geometry (left, lower-right) from the current viewpoint. The second rendering pass (right) maps this picture onto the transformed scene geometry.

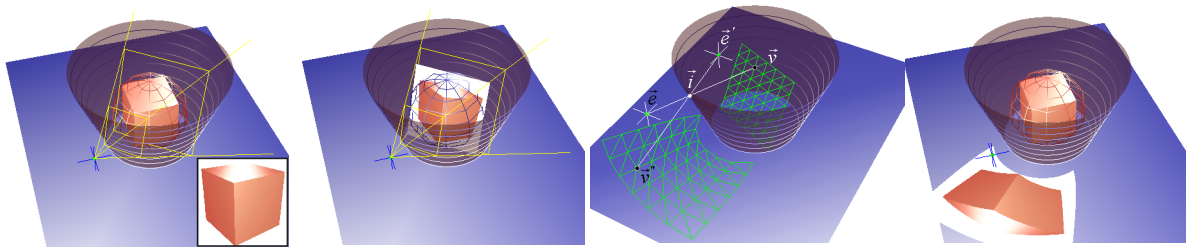
If the image space is observed from the viewpoint è (i.e., seeing the reflection of the deformed picture, projected in the object space) it results in an undeformed picture that is equivalent to the image generated in the first rendering pass. This happens, because the transformation of the mirror optics neutralizes the deformation of the graphics.
Nevertheless, the geometry based reflected model-view transform that has been described above, entails a number of critical shortcomings:

- The image space geometry has to be tessellated with a proper resolution. This speeds down the rendering performance and more complex scenes cannot be displayed at the required frame rates (i.e., not in real-time). Consequently, rendering strongly depends on the scene complexity;
- The geometry-based method requires access to the image-space geometry. Thus, it is restricted to geometric models and cannot be employed in connection with other scene representations.

Note that for the geometry-based approach, the refracted model transform (see section 4.1.3) and the projected image transform (see section 4.1.4) can be optionally applied.

4.2.2 An Image-Based Approach

To address the problems of the geometry-based approach, we want to introduce an image-based two-pass rendering method in this subsection [Bim01d].
The basic idea is to transform (warp) the image that has been generated during the first rendering pass, rather than the image space geometry (cf. figure 4.14). This is possible because (in case perspective projection is used) an image space vertex and its projection on the image plane are located on the same geometric line of sight ŕ. Consequently, the reflection transformation R is the same for both -the vertex and its projection.



(a) (b) (c) (d)

Figure 4.14: Image-based multi-pass approach: The first rendering pass (a) generates a picture of the image space from the current viewpoint. A geometric representation of this picture (b) is transformed (c). Finally, the transformed picture is rendered during the second rendering pass (d).

Most traditional displays are designed as centered optical systems. The optics used for head-mounted displays (HMDs), for instance, are normally placed perpendicular on the optical axis (on-axis) and consequently allows for an efficient pre-distortion to correct geometrical aberrations during rendering. Rolland and Hopkins [Rol93] describe a polygon-warping technique as one possible distortion correction method for HMDs. Since the optical distortion for HMDs is constant, a two-dimensional lookup table is pre-computed that maps projected vertices of the virtual objects' polygons to their pre-distorted location on the image plane. This approach requires subdividing polygons that cover large areas on the image plane. Instead of pre-distorting the polygons of projected virtual objects, the projected image itself can be pre-distorted, as described by Watson and Hodges [Wat95], to achieve a higher rendering performance.
Several projection-based displays apply multi-pass rendering and image warping to present undistorted images on non-planar and off-axis surfaces. Raskar et al. [Ras98c], for instance, apply projective textures [Seg92] and three-pass rendering to seamlessly project images onto static real surfaces. Subsequently, Bandyopadhyay et. al. [Ban01] demonstrated the same approach for movable surfaces. Yang, et al. [Yan01] propose to warp uniform grids in combination with multi-pass rendering to support multiple roughly aligned projectors displaying a unified high resolution image onto a planar surface. Van Belle et al. [Vanb00] integrate geometric pre-distortion methods that apply image warping into the projector hardware.
Projecting undistorted images with zero-parallax onto planar or non-trivial static (i.e., possibly movable, but not dynamically deformable) surfaces, however, does not require a view-dependent warping – although the generation of the image content might be view-dependent.
In contrast to the display approaches described above pre-distortion for non-centered off-axis displays that possibly integrate additional elements (such as curved mirrors or lenses) into the optical path do require a view-dependent image generation and warping.

4.2.2.1 Image Generation

The first rendering pass is defined by the GenerateImage algorithm (see section 4.2.1.2). Note that the rendering method of the first pass can be exchanged. Instead of using a geometric renderer, other techniques (such as image-based and non-photo-realistic rendering, interactive ray-tracing, volume rendering, etc.) can be employed to generate the picture of the image space. Beside an ordinary geometric renderer, a volumetric renderer [Eck98] and a progressive point-based (or splatting) renderer [Rus00] have been applied (see section 5.5.4 for examples).

4.2.2.2 Image-Based Reflected Model-View Transform

The image that has been generated during the first rendering pass (cf. figure 4.14-a) has now to be transformed in such as way that its reflection is perceived undistorted within the mirror. To support the subsequent image deformations, a geometric representation of the image plane is pre-generated. This *image geometry* consists of a uniformly tessellated grid (represented by an indexed triangle mesh) which is transformed into the current viewing frustum inside the image space in such a way that, if the image is mapped onto the grid each line of sight intersects its corresponding pixel (cf. figure 4.14-b). Thus, the image is perpendicular to the optical axis and centred with the image-space geometry. Finally, each grid point is transformed with respect to the mirror geometry, the current viewpoint and the projection plane, and is textured with the image that has been generated during the first rendering pass (cf. figure 4.14-c).
The GenerateImageGeometry algorithm describes how the image geometry is transformed into the viewing frustum. Here, gw and gh define the grid resolution (i.e., its height and width).

GenerateImageGeometry($\dot{e}$ **,** $\dot{p}$ **,** θ **,** gw **,** gh ***):***
1:**begin** create uniform triangle grid
2: *size*: $s = -(\theta, \theta),(-\theta,\theta)$ (image dimensions equal radius of bounding sphere)
3: *position*: $\dot{q} = \dot{p}$ (center of grid equals center of bounding sphere)
4: *orientation*: $[2\pi - \angle([0,-1,0], [\dot{e}_x - \dot{p}_x, \dot{e}_y - \dot{p}_y, 0]), 0, 0, 1]$
$\ldots[\angle(\dot{e} - \dot{p}, [0, 0, 1]), 1, 0, 0]$
(image perpendicular to optical axis)
5: *resolution*: gw, gh
6:**end**

The ImageBasedReflectedModelViewTransform algorithm differs slightly from the geometry-based reflected model-view transform. The main difference is that a simple image grid G is transformed, rather than a complex scene geometry T. This is described in detail below.

ImageBasedReflectedModelViewTransform($\dot{e}$ **,** $G_{i\ldots n}$ **,** Σ ***):***
1:build projection matrix P from $\dot{n}_p = [0, 0, 1]$, $d_p = 0$ and $\dot{i}$
2:**forall** grid vertices $\dot{v}_k$ of G
3: **if** front-facing intersection $\dot{i}$ of $\dot{r} = \dot{e} + (\dot{v}_k - \dot{e})$ with the mirror Σ
4: compute normal $\dot{n}$ at $\dot{i}$
5: compute tangential plane $[a, b, c, d]$ at $\dot{i}$
6: build reflection matrix R from $[a, b, c, d]$
7: **begin** transform $\dot{v}$:(off-axis perspective projection with $\dot{i}$)
8: $\dot{v}'_k = P \cdot V \cdot R_l \cdot S \cdot \dot{v}_k$
9: perspective division $\dot{v}''_k = \frac{\dot{v}'_k}{w}$
10: **end**
11: $\dot{v}''_k$ is visible
12: **else**

13: $\grave{v}''_k$ is not visible
14: **endif**
15: **endfor**

As for the geometry-based reflected model-view transform, we ensure that only visible grid triangles (i.e., the ones with three visible vertices) are rendered during the second rendering pass. Therefore, lines 11 and 13 set a marker flag for each transformed/untransformed vertex.
For all grid vertices, the intersection of the geometric line of sight (i.e., the ray that is spanned by the eye and the vertex) with the mirror geometry is computed first (line 3). Next, the normal vector at the intersection has to be determined (line 4). The intersection point, together with the normal vector, gives the tangential plane at the intersection. Thus, they deliver the plane parameters for the per-vertex reflection transformation (lines 5-6). Note that as for the geometry-based case, an intersection is not given if the viewpoint and the vertex are located on the same side of a tangential plane.
A transformation matrix that, given a projection origin and plane parameters, projects a 3D vertex onto an arbitrary plane, is pre-generated in line 1. This matrix represents a spatial transformation within our 3D world coordinate system. Note that in contrast to the projection for planar mirrors, only the beam that projects a single reflected vertex onto the projection plane is of interest. Thus, the generation and application of a perspective projection defined by an entire viewing frustum in combination with the corresponding viewpoint transformation (e.g., glFrustum and gluLookAt) would implicate too much computational overhead (this was our initial approach for the geometry-based case). Consequently this would slow down the image deformation process. In addition, the reflection of the viewpoint becomes superfluous. Since the reflected viewpoint, the mirror intersection and the final projection of the transformed vertex lie on the same beam, we can use $\grave{i}$ as projection origin, instead of $\grave{e}'$ (again, using $\grave{e}'$ was our initial approach for the geometry-based case). Doing this, we save the matrix multiplication for the viewpoint transformation and the determination of the viewing-frustum.
The new projection matrix which is normally applied to produce hard shadows [Bli88, Nei93 (pp.401)] and variations of this projective transform are used to produce soft shadows (e.g., [Hec96, Hec97]). It is defined by:

$$P = \begin{bmatrix} \kappa - a_p x & -b_p x & -c_p x & -d_p x \\ -a_p y & \kappa - b_p y & -c_p y & -d_p y \\ -a_p z & -b_p z & \kappa - c_p z & -d_p z \\ -a_p & -b_p & -c_p & -d_p \end{bmatrix} \tag{4.3}$$

where $\grave{n}_p = [a_p, b_p, c_p], d_p$ are parameters of the projection plane, $[x, y, z, 1]$ are the coordinates of the projection center, and $\kappa = [a_p, b_p, c_p, d_p] \cdot [x, y, z, 1]$.

Finally, the vertex is transformed and projected (line 8). Since P is still a perspective projection, a perspective division has to be done accordingly to produce correct device coordinates (line 9). Doing this for all image vertices results in the projected image within the object space (cf. figure 4.14-c).
Note that in contrast to the transformation of scene geometry, depth-handling is not required for the transformation of the image geometry. Therefore, an explicit ray-casting (e.g., using equation 2.12 for the reflection computations and an additional surface intersection computation) can be applied instead of a matrix operation. Both methods have been realized, but they

do not differ much in performance. However, matrix operations better represent the notion of a transformation pipeline and will possibly benefit more from next-generation acceleration hardware (such as nVidia's programmable vertex geometry processor [Lin01, Nvid01a], and similar new architectures) than explicit ray computations.

4.2.2.3 Image Rendering

During the second rendering pass, the transformed image geometry is finally displayed within the object space -mapping the outcome of the first rendering pass as texture onto it's surface (cf. figure 4.14-d). Note that only triangles are rendered that consist of three visible vertices.

Since the *image-based reflected model-view transform* delivers device coordinates, and the projection device as well as the mirror optics can be defined within our world coordinate system, a second projection transformation (e.g., glFrustum), and the corresponding perspective divisions and viewpoint transformation (e.g., gluLookAt) are not required (in contrast to our initial geometry-based approach). If a plane projection device is used, a simple scale transformation is sufficient to normalize the device coordinates (e.g., glScale($\frac{1}{dw/2}$, $\frac{1}{dh/2}$)). A subsequent viewport transformation finally up-scales them into the window coordinate system (e.g., glViewport(0,0,ww, wh)).

Time-consuming rendering operations that are not required to display the two-dimensional image (such as illumination computations, back-face culling, depth buffering, etc.) should be disabled to increase the rendering performance. The polygon order has not to be reversed before rendering, as discussed in section 4.1.2.

The required grid resolution of the image geometry also depends on the shape of the mirror. Pixels between the triangles of the deformed image mesh are linearly approximated during rasterization (i.e., during the second rendering pass). Thus, some image portions stretch the texture while others compress it. This results in different regional image resolutions. In contrast to using a uniformly tessellated image grid, we will introduce an adaptive tesselation method to address this problem in section 4.3.1. Since a primitive-based (or fragment-based) antialiasing does not apply in case of a deformed texture, bi-linear or tri-linear texture filters can be utilized instead. As antialiasing, texture filtering is usually supported by the graphics hardware.

Note that the background of the image, and the empty area on the projection plane have to be rendered in black, since black does not emit light and will therefore not be reflected into the image space. For the case that multiple images have to be composed to support multi-user applications, the black background has to be blended appropriately (e.g., by using glBlendFunc(GL_ONE,GL_ONE)). However, to avoid that the pixels of the grid-polygon edges and the texels that overlay these edges are rendered together and cause blending artefacts above the edges, depth-buffering has to be enabled. Note that no artefacts arise if blending is disabled - even with a disabled depth-buffer. This is because the pixels of the polygon edges have the same color as the overlaying texels.

In contrast to the geometry-based approach, the image-based approach avoids a direct access to the image space geometry and its tesselation, and consequently prevents the time-consuming transformations of many scene vertices.

A performance evaluation of the image-based approach and a comparison to the geometry-based approach is presented in section 6.3.2. In addition, section 6.3.2 analyzes clues for possible acceleration approaches which have driven the implementation of the methods that are introduced in section 4.3.

The following subsections will define additional image operations that can be optionally carried out between the two rendering passes. A graphical overview of the entire image-based rendering approach is presented in the summary section.

4.2.2.4 Refracted Image Transform

The RefractedImageTransform algorithm demonstrates how to apply refraction transformation to the image that has been generated during the first rendering pass (cf. figure 4.15). Note that the image is refracted before the image-based reflected model-view transform is applied to the image geometry.

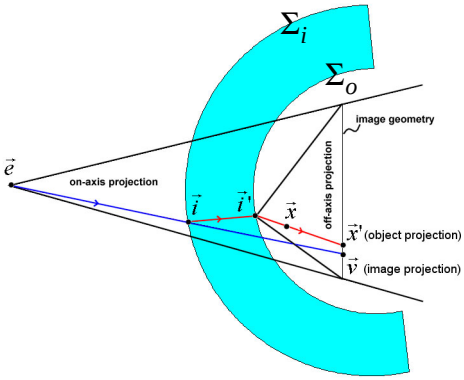


Figure 4.15: Refracted image transform for curved lenses: Geometric lines of sight (blue) through mediums with different densities are refracted. The corresponding optical lines of sight (red) can be computed with Snell's law of refraction.

RefractedImageTransform($\grave{e}$ **,** $\grave{v}$ **,** Σ_i **,** Σ_o **,** η ***):***

1:compute intersection $\grave{i}$ of $\grave{r} = \grave{e} + (\grave{v} - \grave{e})$ with Σ_i and determine $\grave{n}$ at $\grave{i}$

2:compute in-refracted ray $\grave{r}'$ from $\grave{r}$, $\grave{n}$, and η

3:compute intersection $\grave{i}'$ of $\grave{r}'$ with Σ_o and determine $\grave{n}'$ at $\grave{i}'$

4:compute out-refracted ray $\grave{r}''$ from $\grave{r}'$, $\grave{n}'$ and η

5:transform $\grave{i}'$ and an arbitrary point $\grave{x}$ on $\grave{r}''$ into the coordinate system of the image geometry

6:**begin** set texture matrix (off-axis perspective projection with $\grave{i}'$)

7: set normalization correction:

$$S = Scale(0{,}5,\ 0{,}5,\ 0{,}5),\ S = S \cdot Translate(1,\ 1,\ 0)$$

8: set projection transformation:

$$\phi = \frac{(\grave{i}'_z - 1)}{\grave{i}'_z},\ left = -\phi(1 + \grave{i}'_x),\ right = \phi(1 - \grave{i}'_x),$$

$$bottom = -\phi(1 + \grave{i}'_y),\ top = \phi(1 - \grave{i}'_y),\ near = \grave{i}'_z - 1,\ far = \grave{i}'_z + 1$$

$$P = Frustum(left,right,bottom,top,near,far)$$

9: set view transformation:

$$V = LookAt(\grave{i}'_x,\grave{i}'_y,\grave{i}'_z,\ \grave{i}'_x,\grave{i}'_y,0,\ 0,1,0)$$

10:**end**

11:compute new texture coordinate $\grave{x}'$ for the particular $\grave{x}(\grave{v})$:

$$\grave{x}' = S \cdot \frac{(P \cdot V \cdot \grave{x})}{w}$$

Given an image vertex $\grave{v}$ and the viewpoint $\grave{e}$, the according geometric line of sight is computed. Using Snell's law of refraction (eqn. 2.2) together with the vectorized equation for specular refraction rays (eqn. 2.21), the corresponding optical line of sight can be determined by computing the in/out refractors at the associated surface intersections (lines 1-4).
We could now determine the refraction of the image vertex $\grave{v}$ by computing the geometric intersection of the out-refractors with the image geometry. To simulate refraction, however, we only need to ensure that the pixel at $\grave{x}'$ (i.e., the projection of an object) will be seen at the location of $\grave{v}$ (i.e., the projection of the image at which the corresponding object optically appears). Instead of generating a new image vertex at $\grave{x}'$ and transforming it to the location of $\grave{v}$, we can also assign the texture coordinate at $\grave{x}'$ to the existing vertex $\grave{v}$.
In this case, we can keep the number of image vertices (and consequently the time required for the reflection transformation) constant. The intersection of the in-refractor with the outer mirror surface $\grave{i}'$ and an arbitrary point on the out-refractor $\grave{x}$ are transformed into the coordinate system of the image geometry, next (line 5).
The composition of an appropriate texture matrix that computes new texture coordinates for the image vertex is outlined in lines 6-10. As illustrated in figure 4.15, an off-axis projection transformation is applied, where the center of projection is $\grave{i}'$. Multiplying $\grave{x}$ by the resulting texture matrix and performing the perspective division projects $\grave{x}$ to the correct location within the normalized texture space of the image (line 11). Finally, the resulting texture coordinate $\grave{x}'$ has to be assigned to $\grave{v}$.
Note that if the image size is smaller than the size of the allocated texture memory, this difference has to be considered for the normalization correction (line 7). In this case, the image's texture coordinates are not bound by the value range of $[0, 1]$, as it was assumed in the RefractedImageTransform algorithm.
As mentioned for the image-based reflected model-view transform the matrix operations can be replaced by an explicit ray-casting (e.g., using equation 2.21 for the refraction computations and an additional surface intersection computation).
Note that the derivation of the optical lines of sight for planar lenses is less complex, since in this case the optical lines of sight equal the parallel shifted geometric counterparts (cf. figure 4.16).

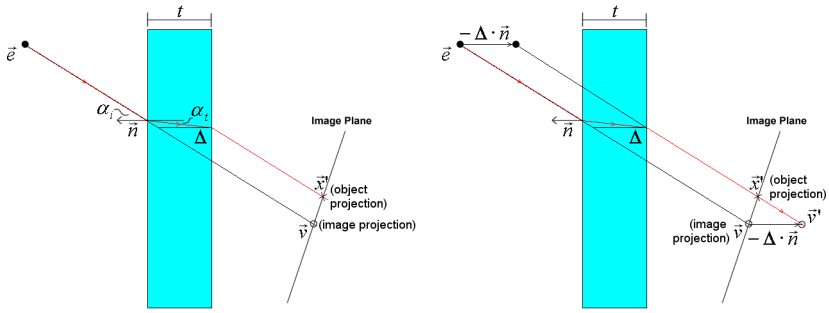


Figure 4.16: Refracted image transform for planar lenses.

In this case, Σ_i and Σ_o share the same normal vector, but are offset by the thickness t. Consequently, the optical line of sight does not have to be traced through the lens to determine the entrance vector and normal at the exit point $\vec{i}'$ (lines 1-5). Instead, the entrance and exit angles α_i, α_t have to be computed (using Snell's law, eqn. 2.2) for the given $\vec{r}$, $\vec{n}$ and η. Then, the transformation offset Δ can be computed with α_i, α_t and t using equation 2.13. The texture matrix is composed in a similar way as outlined by RefractedImageTransform algorithm. However, due to the refraction properties of planar lenses, the center of projection is given by $\vec{e} - \Delta\vec{n}$ (instead of using $\vec{i}'$). To determine the correct texture coordinate $\vec{x}'$ for a given image vertex $\vec{v}$, $\vec{v}' = \vec{v} - \Delta\vec{n}$ has to be projected onto the image plane.
Note that as for the image-based reflected model-view transform the shadow matrix given by equation 4.3 can be used instead of the matrix that results from a composition of OpenGL's glFrustum and gluLookAt [Nei93]. However, a different texture normalization S is required, since the shadow matrix delivers world coordinates, rather than normalized device coordinates.
Nevertheless, both refraction methods face the following problems for outer areas on the image:

- Given a geometric line of sight to an outer image vertex, its corresponding optical line of sight does not intersect the image. Thus, an image vertex exists but its new texture coordinate cannot be computed. This results in vertices with no, or wrong texture information;
- Given an optical line of sight to an outer pixel on the image, its corresponding geometric line of sight does not intersect the image. Thus, a texture coordinate can be found but an assignable image vertex does not exist. Consequently, the portion surrounding this pixel cannot be transformed. This results in image portions that aren't mapped onto the image geometry.

A simple solution to address these problems does not avoid them, but ensures that they do not occur for image portions which contain visible information: The image size depends on the radius of the scene's bounding sphere. We can simply increase the image by adding some constant amount to the bounding sphere's radius before carrying out the first rendering pass. An enlarged image does not affect the image content, but subjoins additional outer image space that does not contain any visible information (i.e., just black pixels). In this way, we ensure that the above mentioned problems emerge only at the new (black) regions. Yet, these regions will not be visible as reflections in the mirror.
Note that the *refracted image transform* represents another transformation of the image generated during the first rendering pass. In contrast to the image-based reflected model-view transform which transforms image vertices, the refracted image transform re-maps texture coordinates. However, all image transformations have to be applied before the final image is displayed during the second rendering pass.
In contrast to the refracted model transform, described in section 4.1.3, the refracted image transform computes correct refractions, rather than approximations. This is because the refracted model transform tends to compute image positions for given object positions. However, equation 2.21 supports the reverse way, only (as stated in sections 2.4.1 and 4.1.3). The approximation of the refracted model transform is the assumption that the angular difference between the geometric lines of sight spanned by the viewpoint and the image or spanned by the viewpoint and the object is small, and that consequently the transformation offsets Δ for both equal. This approximation is necessary for a geometry-based approach because the scene vertices represent object positions, and the image positions are unknown.

Optically, the refracted image transform, however, determines projections of objects $\grave{x}'$ for discrete image projections $\grave{v}'$, and interpolates object projections for intermediate image projections by making use of texture mapping. In this case, image projections and corresponding object projections can be found at different pixel locations within the image.

4.2.2.5 Implicit Projected Image Transform

Since our image-based rendering approach already contains two rendering passes, the projected image transform that has been described in section 4.1.4 for geometry-based transformations can be incorporated implicitly -as a further image transformation step.

For the following, we assume that the pre-distorted grid P has been computed from the undistorted grid U and the distorted grid D as described in section 4.1.4.

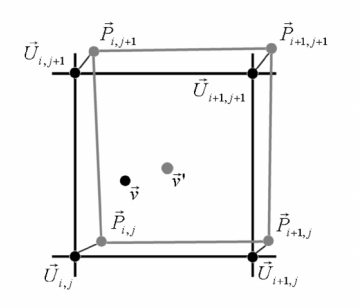


Figure 4.17: Implicit projected image transform within a single grid cell.

The implicit ProjectedImageTransform algorithm is outlined below.

ProjectedImageTransform($\grave{v}$ ***,*** U ***,*** D ***):***

1:**begin**

2: find the undistorted grid cell that encloses $\grave{v}$ within U:

$$\grave{U}_{i,j}, \grave{U}_{i+1,j}, \grave{U}_{i,j+1}, \grave{U}_{i+1,j+1}$$

3: compute normalized parameters u, v of $\grave{v}$ within grid cell:

$$u = \frac{\grave{v}_x - \grave{U}_{i,j,x}}{\grave{U}_{i+1,j,x} - \grave{U}_{i,j,x}},\ v = \frac{\grave{v}_y - \grave{U}_{i,j,y}}{\grave{U}_{i,j+1,y} - \grave{U}_{i,j,y}}$$

4: compute $\grave{v}'$ by linear interpolating between the corresponding pre-distorted grid cell points: $\grave{P}_{i,j}, \grave{P}_{i+1,j}, \grave{P}_{i,j+1}, \grave{P}_{i+1,j+1}$:

$$\grave{v}' = (1-u)(1-v)\grave{P}_{i,j} + u(1-v)\grave{P}_{i+1,j} + (1-u)v\grave{P}_{i,j+1} + uv\grave{P}_{i+1,j+1}$$

5:**end**

The grid cell within U that encloses $\grave{v}$ (after projecting onto the display surface) and the normalized cell coordinates u, v of $\grave{v}$ within this cell have to be determined (line 2-3).

Finally, a pre-distorted vertex $\grave{v}'$ can be computed by linear interpolating within the corresponding pre-distorted grid cell of P, using the normalized cell coordinates (line 4). This is illustrated in figure 4.17.

Displaying the transformed image vertex at its pre-distorted position lets it appear at its correct location on the projection plane, since the pre-distortion is neutralized by the projector's geometry distortion.

Note that the *implicit projected image transform* simply represents yet another image transformation. It is applied after the image-based reflected model-view transform and before the second rendering pass is carried out. As the refracted image transform, the projected image transform is optional and can be switched off to safe rendering time- even though, it does not slow down rendering performance significantly.

4.2.3 Concave Mirrors and Mirrors of Mixed Convexity

As discussed in chapter 2, concave mirrors can generate both -real images and virtual images. In sections 4.1.5 and 4.2, we mentioned that light rays which are reflected off convex mirror assemblies do not intersect. In contrast to this, light rays that are reflected off a parabolic concave mirror do intersect exactly within its focal point. For concave mirrors that deviate from a parabolic shape (e.g., spherical mirrors) the light rays do not intersect within a single point, but bundle within an area of rejuvenation.

The rendering techniques introduced above and the ones discussed below, are described on the basis of convex mirrors. However, they can be applied for concave mirrors without or with only minor modifications to the algorithms.

Figure 4.18 illustrates how, given the viewpoint and the location of the mirror, a geometry is transformed by the GeometryBasedReflectedModelViewTransform algorithm (without projection onto the plane). The portion of the geometry that is located within area A is mapped to area A' (behind the mirror's focal point -as seen from the viewpoint). This mapping has a similar behavior as for convex mirrors. The portion of the geometry that is located within area B, is mapped to area B' (in front of the mirror's focal point -as seen from the viewpoint). In this case, the mapped geometry is flipped and the polygon order is changed. The geometry that is located on the intersecting surface of A and B is mapped to the focal point.

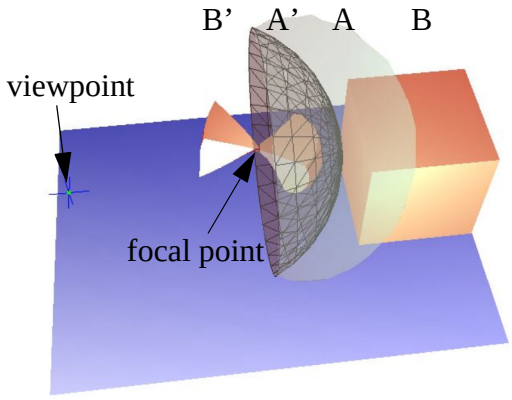


Figure 4.18: Reflected geometry at a concave mirror.

Figure 4.19 shows that concave mirrors can be treated just like convex mirrors. The left image indicates the application of the GeometryBasedReflectedModelViewTransform algorithm (again, without projection onto the plane). The transformed geometry and the mirror model have been exported to a ray-tracer and the image that is seen from the viewpoint has been ray-traced. The result is outlined at the lower right of the left image (red cube).

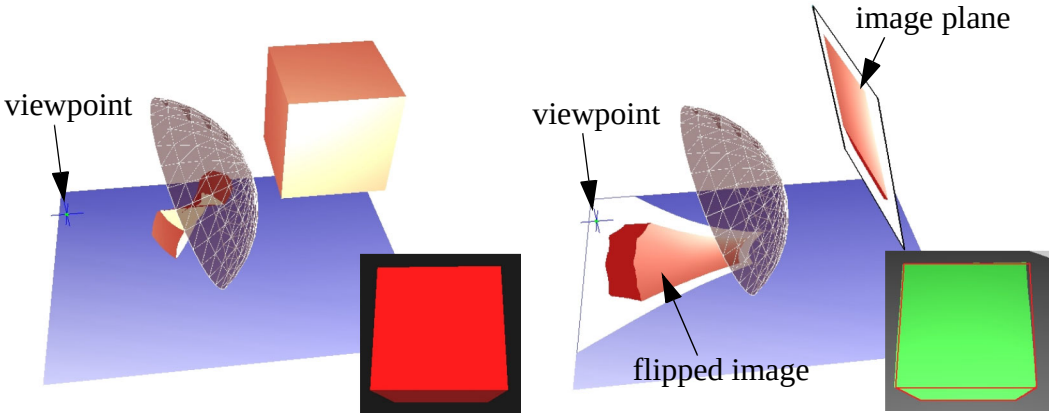


Figure 4.19: The geometry-based approach (left) and the image-based approach (right), applied with a concave mirror.

We can see, that the transformed geometry appears untransformed as reflection in the mirror. Note that due to the different orders of polygons that are located in front of and behind the focal point, polygon order related options offered by the applied rendering package should be disabled. Otherwise, additional actions have to be taken to determine the location of each transformed polygon and its order.
The same experiment has been carried out with the ImageBasedReflectedModelViewTransform algorithm (including the projection onto the plane). The image geometry was transformed and the image that has been generated during the first rendering pass was mapped onto the transformed grid. The right part of figure 4.19 illustrates that the projected image is a flipped version of the original image. If ray-traced from the given viewpoint, this deformed image appears as correct reflection in the mirror (see lower right of the right image). Note that the contours of the ray-traced result have been highlighted, since the planar projection of the deformed image does not provide sufficient normal or depth information to generate correct shading effects with the ray-tracer. Note also, that the geometry-based reflected model-view transform that includes the projection yields the same results as the image-based reflected model-view transform.
Mirrors with a mixed convexity (i.e., simultaneously convex and concave mirrors) can cause multiple images of the same object, or they can reflect multiple objects to the same location within the image space. In such cases, the transformations from object space to image space and vice versa are not definite and are represented by possible many-to-many mappings. Such types of mirrors should be decomposed into convex and concave parts (as done by Ofek [Ofe99]) to ensure a correct functioning of our algorithms. Although for many surfaces this can be done fully atomically [Spa92], PBAR configurations do not make use of mirrors with mixed convexities. This is because one of the initial aim of the PBAR concept -namely to unequivocally overlay real environments with computer graphics in an AR manner- is physically not supported by such mirrors. Consequently, we do not consider mirrors with mixed convexities for our concept or for the presented algorithms.

4.3 Acceleration Schemes

This section introduces selective refinement, progressive rendering and parallel processing as possible acceleration schemes for those components of our image-based two-pass rendering method that are not supported by hardware acceleration. The cone-like Virtual Showcase,

described in section 5.5.4 serves again as a proof-of-concept prototype for the following acceleration techniques.

4.3.1 Selective Refinement

Displays which require a non-linear pre-distortion to present undistorted images on non-planar surfaces (e.g., [Vanb00, Ras01, Ban01]), or to neutralize optical distortion (e.g., [Wat95, Yan01]) usually apply multi-pass rendering techniques. Projective textures [Seg92] or uniform grids are used to deform the image generated during the first pass before it is displayed as a texture map during the final pass. However, these approaches do not consider the error that is generated from a piecewise linear texture interpolation to adapt the underlying geometry. The algorithm that is presented in this section supports this.

If image warping is applied for a non-linear pre-distortion, the required grid resolution of the underlying image geometry depends on degree of curvilinearity that is introduced by the display (e.g., caused by the properties of a mirror, a lens, or a projection surface). Pixels within the triangles of the warped image mesh are linearly approximated during rasterization (i.e. after the second rendering pass). Thus, some image portions stretch the texture while others compress it. This results in different regional image resolutions.

If, on the one hand, the geometric resolution of the uniform grid is too coarse, texture artifacts are generated during rasterization. This happens in this case because a piecewise bi- or tri-linear interpolation within large triangles is only a crude approximation to neutralize the curvilinear optical transformations. If, on the other hand, the grid is oversampled to make the interpolation sections smaller, interactive frame rates cannot be achieved. In an extreme case, the grid resolution equals the image resolution (i.e., the size of the patches matches the size of a pixel). For this situation, the correct position of each pixel is computed individually (depending on the configuration of the optical elements), rather than being approximated. Note that this corresponds to a ray-tracing approach and is not yet realizable in real-time.

In addition to the speed of the first rendering pass, the performance of view-dependent multi-pass methods depends mainly on the complexity of the image geometry that influences geometric transformation times and on the image resolution that influences texture transfer and filter times.

This section focuses on the image geometry. We introduces a selective refinement method that generates image grids with appropriate regional grid resolutions on the fly. This method avoids oversampling and the occurrence of artifacts within the final image. Note that we do not consider a level-of-detail refinement of the scene geometry that is rendered during the first pass. Although this would also reveal an additional speedup for the overall rendering process, it is out of our scope.

The main difficulty for a selective refinement method that supports mirror displays is that in contrast to simpler screens, a displacement error which defines the refinement criterion has to be computed within the image space of their optics, rather than in the screen space of the display. For convexly curved mirrors, this requires fast and precise numerical methods. For displays that do not contain view-dependent optical components (e.g., curved screens), these computations are much simpler because analytical methods can be used to determine the error within the object space (e.g., the screen space).

The effectiveness of our selective refinement method, both -from a visual appearance and a performance speed-up point of view- is discussed in section 6.4.1.

4.3.1.1 Background

Recent advances in level-of-detail (LOD) rendering take advantage of temporal coherence to adaptively refine geometry between subsequent frames. Especially terrain-rendering applica-

tions consider viewing parameters to regionally enhance terrain models [Gro95, Linds96, Hop98].
Hoppe introduces progressive meshes [Hop96] and has later developed a view-dependent refinement algorithm for progressive meshes [Hop97, Hop98]. Given a complex triangle mesh, Hoppe first pre-generates a coarse representation (so-called base mesh) by applying a series of edge collapse operations. A sequence of pre-computed vertex split operations (that are inverse to the corresponding edge collapse operations) can then be applied to the base mesh's regions of interest to successively refine them. The selection of the appropriate vertex split operations is based on his refinement criteria. The main advantage of progressive meshes is that they support a progressive transmission of geometry data over a network. Thus, the coarse base-mesh is transmitted from a host to the local machine, first. Instead of sending more complex geometric LODs to provide refinements, a sequence of light-weight vertex split operations is transmitted.
Lindstrom [Linds96] describes a method that generates a view-dependent and continuous LOD of height fields dynamically in real-time, instead of pre-computing a coarse base mesh and a sequence of refinement steps. He hierarchically subdivides a regular height field into a quad-tree of discrete grid blocks with individual LODs. Beginning with the highest LOD, Lindstrom regionally applies a two-step surface simplification method: He first determines which discrete LOD is needed for a particular region by applying a coarse block-based simplification, and then performs a fine-grained re-triangulation of each LOD model in which vertices can be removed. To satisfy continuity among the different LODs, Lindstrom considers vertex dependencies at the block boundaries.
The main difference between both methods is that Lindstrom performs a dynamic simplification of high-resolution height fields for domains in $\Re^2$ during rendering. Lindstrom's mesh definition provides an implicit hierarchical LOD structure. Hoppe applies refinement steps to low-resolution LODs of arbitrary meshes during rendering. His mesh definition is more general and does not require an implicit hierarchical LOD structure. Consequently, the refinement steps and the low-resolution base mesh have to be pre-computed. In addition, he applies triangulated irregular networks (TINs) for triangulation, rather than regular grids. Note that these two types of refinement methods may be representative for related techniques.

Since our image grid can also be parameterized in $\Re^2$ and provides an implicit hierarchical LOD structure, multiple LODs or appropriate refinement steps do not need to be pre-computed, but can be efficiently determined on the fly. However, simplifying a high-resolution mesh instead of refining a low-resolution mesh would require to re-transform all grid vertices of the highest LOD after a change of the viewpoint occurred. This is very inefficient, since for the type of displays which we consider viewpoint changes normally happen at each frame.
In contrast to the static geometry that is assumed for the methods described above our image-grid geometry is not constant, but it dynamically deforms with a moving viewpoint. Consequently, the geometry within all LODs dynamically changes as well.
Therefore, we propose a method that dynamically deforms the image geometry within the required LODs while selectively refining the lowest-resolution base mesh during rendering. The method aims at minimizing the displacement error of the image portions, the complexity of the image geometry and -consequently- the number of vertex transformations and triangles to be rendered.
While subsection 4.3.1.2 first presents our image triangulation approach, subsection 4.3.1.3 outlines the general recursive grid generation and refinement algorithm. Subsection 4.3.1.4 describes our main generation and refinement criteria, and how to efficiently compute them to achieve interactive framerates. Since the computations of the refinement criteria are partially display specific, some components are explained using the example of our curved mirror dis-

play. Finally, subsection 4.3.1.5 outlines the display specific components of the algorithm and describes how they can be adapted for other displays.

4.3.1.2 Image Triangulation

Instead of transforming and rendering a uniform high-resolution mesh, we start from the coarsest geometry representation and successively refine it regionally until certain refinement criteria are satisfied. Due to the well-defined structure of our image grid, all possible global or discrete LODs can be computed at runtime – including the highest, which is the uniform high-resolution representation of the mesh itself.
Figure 4.20 illustrates our image triangulation approach, which is similar to Lindstrom's triangulation method for height fields [Linds96]. While figure 4.20-left shows an unrefined patch at LOD n, figure 4.20-right shows the same patch at LOD $n+1$, with lower LOD neighbors.

Given a highest LOD of m, we chose an indexed $(2^m + 1) \times (2^m + 1)$ matrix structure to store the grid vertices.
We want to differentiate between the following types of patch vertices:

- *L-vertices* are vertices at the corners of a patch (e.g., at indices [*i*,*j*],[*i*,*l*],[*k*,*l*] and [*k*,*j*] in figure 4.20-left);
- *X-vertices* are vertices at the center of a patch (e.g., at index [(*k*-*i*)/2,(*l*-*j*)/2] in figure 4.20-left);
- *T-vertices* are vertices that split the patch edges after refinement (e.g., at indices [*i*,(*l*-*j*)/2], [*k*,(*l*-*j*)/2], [(*k*-*i*)/2,*l*], and [(*k*-*i*)/2,*j*] in figure 4.20-right)

To refine a patch, it is divided into four sub-patches by computing the corresponding four T-vertices, as well as the four X-Vertices that lay inside the sub-patches. Note that the matrix structure is in our particular case equivalent to a quad-tree data structure. To ensure consistency during rasterization, the T-vertices have to be connected to their neighboring X-vertices wherever a LOD transition occurs (e.g., at all four neighbors of the refined patch, as shown in the example illustrated in figure 4.20-right).

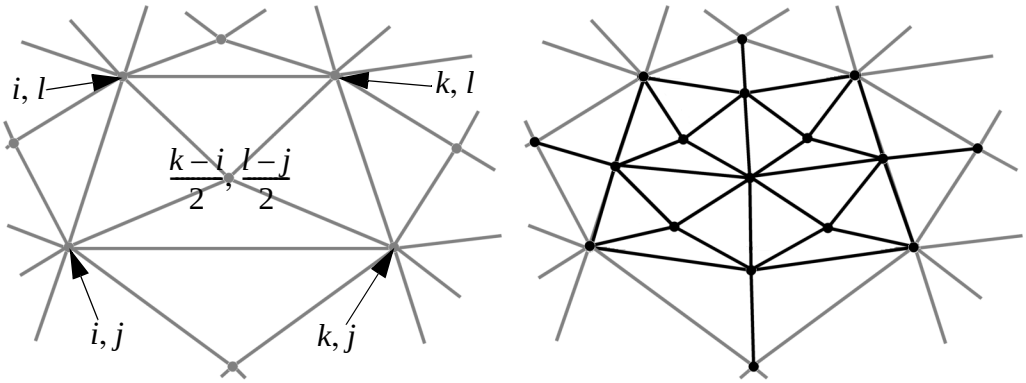


Figure 4.20: Triangulation of unrefined patch at LOD n (left), and triangulation of refined patch at LOD $n+1$ with resolution transitions (right).

Due to the well-defined matrix structure that contains the image grid, the following conditions are given:

(i) A clear relationship between the X-vertices and T-vertices exists: X-vertices can never be T-vertices, and vice versa.
(ii) Each patch has definite L-vertices, T-vertices and X-vertices, whose indices can always be computed.
(iii) Each X-vertex can be explicitly assigned to a single patch at a specific LOD.
(iv) Each T-Vertex can be explicitly assigned to exactly one or two adjacent patches at the same LOD.

The triangulation methods described above require *continuous* level of detail transitions [Tay96]. This implies that neighboring patches do not differ by more than one LOD.

4.3.1.3 Recursive Grid Refinement

The objective of this step is to generate an image grid that provides a sufficient regional grid resolution (i.e. appropriate discrete LODs) to avoid artifacts within the rasterized texture that would result from undersampling, as well as oversampling.
The following pseudo code illustrates our approach to recursively refine a grid patch, which initially is equivalent to the lowest LOD (i.e. the patch at the lowest LOD is outlined by the L-vertices at the four corners of the image geometry):

```
RecursiveGridRefinement(i,j,k,l )
1:  begin
2:    a=(k-i)/2 , b=(l-i)/2
3:    if GeneratePatch([i,j],[i,l],[k,l],[k,j],[a,b])
4:    begin
5:      TransformPatchVertices([i,j],[i,l],[k,l],[k,j],[a,b])
6:      P = P ∪ {[i, j, k, l]}
7:      if RefineFurther([i,j],[i,l],[k,l],[k,j],[a,b])
8:      begin
9:        RecursiveGridRefinement(i,j,a,b)
10:       RecursiveGridRefinement(a,j,k,b)
11:       RecursiveGridRefinement(i,b,a,l)
12:       RecursiveGridRefinement(a,b,k,l)
13:       if j < 2^m + 1   TC[a,j]+=1
14:       if k < 2^m + 1   TC[k,b]+=1
15:       if l > 1   TC[a,l]+=1
16:       if i > 1   TC[i,b]+=1
17:     end
18:   end
19: end
```

The patch that has to be refined is stored at indices *i,j,k,l* within our matrix structure. Condition (ii) allows us to locate the position of the patch-individual X-vertex at indices *a,b* (line 2). First, we evaluate whether or not a patch has to be generated at all (line 3). The conditions that are implemented within the GeneratePatch function will be discussed in subsection 4.3.1.4. The four L-vertices and the X-vertex are transformed from the image plane to the display surface (line 5) –as outlined in subsection 4.2.2. Note that the TransformPatchVertices function is representative for a composition of the in section 4.2.2 discussed image vertex transformations. For our mirror display the image plane is located within the image space of the optics. In

this case, these mappings composite the vertex individual model-view transformations to neutralize reflection and refraction, as well as the projection transformation that maps a vertex onto the display surface. Note that vertices are only transformed once – even if the recursive refinement function addresses them multiple times. This is realized by attaching a marker flag to each vertex. A reference to the transformed patch is stored in the patch set *P* by adding the patch's indices to *P* (line 6). In line 7, we call a function that evaluates the transformed patch based on pre-defined refinement criteria and decides whether or not this patch has to be further refined. Our main refinement criterion is described in subsections 4.3.1.4.2 through 4.3.1.4.4. If this decision is positive, the patch is divided into four equal sub-patches and the refinement function is recursively called for all of these sub-patches (lines 9-12). Note that condition (ii) also allows us to determine the indices of the patch's four T-vertices, which become L-vertices of the sub-patches in the next LOD. Consequently, the GeneratePatch and the RefineFurther functions represent the exit conditions for the recursion.
In subsection 4.3.1.2 we said that T-vertices have to be connected to their neighboring X-vertices whenever an LOD transition occurs to ensure consistency during rasterization. To detect LOD transitions, we attach a counter (TC) to each T-vertex. This counter is incremented by 1, each time the corresponding T-vertex is addressed during the recursive refinement (lines 13-16). Note that the if-conditions ensure a correct behavior of the counter at the image geometry's boundaries. Due to condition (iv) each counter can have one of the following three values:

- *0*: indicates that the T-vertex is located at a boundary edge of the image geometry or it is contained by a discrete LOD that is higher than the required one for the corresponding region.
- *1*: indicates a LOD transition between the two neighboring patches that –with respect to condition (iv)– belong to the T-vertex.
- *2*: indicates no resolution transition between the two neighboring patches that belong to the T-vertex.

After the image grid has been completely generated, all patches that are referred to in *P* are rendered with appropriate texture coordinates during the second rendering pass. Thereby, the counters of the patch's four T-vertices are evaluated. Depending on their values, either one or two triangles are rendered for each counter. These triangles form the final patch. Counter values of 0 or 2 indicate no LOD transition between adjacent patches. Consequently, a single triangle can be rendered which is spanned by the T-vertex's neighboring two L-vertices and the patches X-vertex (this is illustrated in figure 4.20-right). A counter value of 1, however, indicates a LOD transition. According to subsection 4.3.1.2, two triangles have to be rendered that are spanned by the T-vertex itself, the two neighboring L-vertices and the X-vertex of the adjacent patch (this is illustrated in figure 4.20-left).

4.3.1.4 Generation and Refinement Criteria

For large terrain models and similar geometry, view-dependent criteria (such as viewing-frustum content [Fun93, Linds96, Hop96, Hop97, Hop98]), appearance criteria (such as the screen-space error [Hop97] or smallest possible projections [Linds96]) and geometric criteria (such as the slopes of terrain meshes [Linds96] or surface orientations [Hop97]) are applied to make decisions about further refinements.
This subsection discusses the patch generation and refinement criteria that are implemented within the GeneratePatch and RefineFurther functions. The input for these functions is the four L-vertices, as well as the X-vertex of a patch. They deliver the Boolean value *true* if the patch has to be generated, transformed, rendered or further refined, or *false* if this is not the case.

The GeneratePatch function evaluates whether or not a patch has to be generated at all. An evaluation criterion that considers the scene's convex hull is described in subsection 4.3.1.4.1. In general, the RefineFurther function can represent a Boolean concatenation of multiple refinement criteria (such as maximal patch size, angular patch deformation, etc.). An important refinement criterion for LOD methods is the screen space error. Since the computations of this displacement error are display specific, we describe an important variation of the screen space error that can be applied for convex mirror displays -the *image space error*. This error is explained in greater detail in subsections 4.3.1.4.2 through 4.3.1.4.4. Another refinement criterion -the *projected patch size*- is described in subsection 4.3.1.4.5.

4.3.1.4.1 Spatial Limits

The multi-pass rendering method that is described in section 4.2.2 uses the scene's bounding sphere to determine the parameters of the symmetric viewing frustum and the image size (as illustrated in figure 4.14). Since all image generation methods assume a rectangular image shape that is adapted to today's screen shapes, the bounding sphere provides enough information to determine the rectangular image size.

Bounding spheres, however, are only rough approximations of the scene's extensions and consequently cover a fairly large amount of void space. This void space results in grid patches on the image plane whose texture does not contain visible color information.

To speed up our method, we aim at avoiding these patches while creating the image grid. This implies that these patches are not transformed and refined during the RecursiveGridRefinement algorithm and that they are not rendered during the second rendering pass. A condition that causes the recursion to exit in these cases is implemented within the GeneratePatch function:

We evaluate a tighter convex container of the scene that is generated either in a pre-process for static objects, or at runtime for animated scenes. For each untransformed patch that is passed recursively into the RecursiveGridRefinement algorithm, we have to determine whether the container is visible on that patch – partially or as a whole. Our approximation is twofold: First, we intersect the geometric lines of sight from è to all four L-vertices of the patch with the front-facing portion of the container. Second, we intersect the geometric lines of sight from è to front-facing container vertices with the patch. If at least one of the resulting rays causes an intersection, the patch might contain visible color information and it will be further processed. If, however, all rays don't cause intersections, the patch is not treated further (i.e. it won't be transformed nor refined or rendered). If the convex container is represented as a triangle mesh, a fast ray-triangle intersection method [Moe97] is applied together with the front-facing container triangles. Note that as for vertex transformations, the intersection information are buffered and looked up in the memory, rather than re-computing them multiple times while evaluating adjacent patches.

We use oriented convex hulls as containers. It is obvious that the complexity of the container influences the performance of this method. Although precise container can eliminate a maximum number of patches, the number of intersection tests increases with the container's number of vertices and polygons. Our experiments have shown that the highest speedups are reached if the container is as simple as an oriented bounding box or a very coarse but tighter convex hull. However, the complexity of the convex hull that maximizes the speedup and balances intersection tests with patch generations depends on the scene and the required rendering precision.

4.3.1.4.2 Image Space Error

The *image space error* (δ_{is}) is a variation of a screen space error that can be computed for convex mirror displays which present the image plane within the image space of the optics, rather than on a display surface. We want to define the image space error as the geometric distance between the desired position ($\grave{\nu}_d$) and the actual appearance ($\grave{\nu}_a$) of a point on the image plane. Consequently the image space error is given by $\delta_{is} = |\grave{\nu}_d - \grave{\nu}_a|$ and delivers results in image space coordinates (e.g., *mm* in our case).

The consideration of the screen space error that is determined relative to a display surface is a common measure for many computer graphics methods (e.g., [Linds96, Hop97, Hop98]). However, our image space error has to be determined on the image plane which is located inside the image space (as illustrated in figure 4.14) rather than on the display surface. The image space is the virtual space, generated by an optical element. In case of our Virtual Showcase mirror optics, the image space is the reflection of the object space (i.e. the physical space in front of the mirror optics) that optically overlays the physical space inside the Virtual Showcase. In addition, the optically deformed pixels do not maintain a uniform size within the image space. Consequently, we chose an Euclidean distance between geometric points as an error metric, rather than expressing the image space error in pixels.

For any given pixel on the transformed patch with texture coordinates *u,v*, we can compute δ_{is} as follows (cf. figure 4.21):

First we determine the pixel's world coordinate $\grave{\nu}''$ at *u,v* within the object space (i.e. on the display surface). Note that the pixels, which are mapped onto the patch's transformed vertices, optically appear at their correct locations on the image plane inside the image space. This is because their exact mappings have been determined during the patch's transformation. This transformation considers the laws of geometric optics (i.e., reflection and refraction laws). Note also that the pixels that are displayed anywhere else (i.e. inside one of a patch's triangle) do not necessarily appear at their correct locations on the image plane. This is because their positions on the display surface are approximated by a linear texture interpolation, rather than by optical laws.

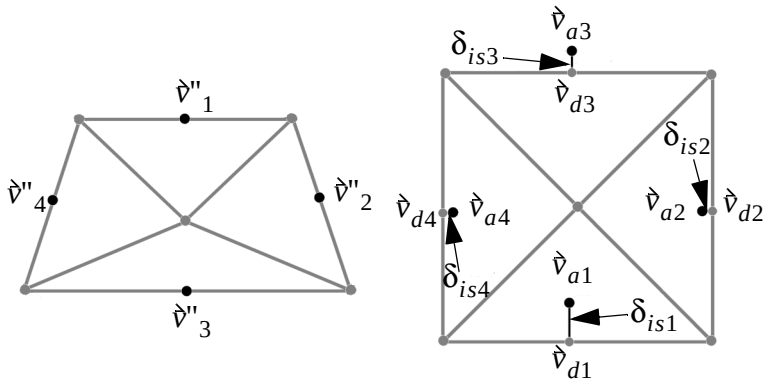


Figure 4.21: Samples on transformed patch (left). The distance between desired and actual appearance of samples near the untransformed patch results in the image space error (right).

The second step is to determine the position of the optical image ($\grave{v}'$) of $\grave{v}''$ within the image space of the mirror optics. The projection of $\grave{v}'$ onto the image plane results in $\grave{v}_a$. The transformation from $\grave{v}''$ to $\grave{v}_a$ will be discussed in detail in subsection 4.3.1.4.3.

In an ideal case, $\grave{v}_a$ is located at the position that also maps to the texture coordinates *u*,*v* within the untransformed patch. We can identify the location which does this as our desired image position $\grave{v}_d$. However, if $\grave{v}_a \neq \grave{v}_d$, the image space error δ_{is} for this pixel is non-zero.
We chose to compute the image space errors for the four points on the transformed patch (4.21-left) that should map to the patch's T-vertices on the untransformed patch (4.21-right) as if the image space errors were zero for these positions. Obviously this is not the case in our example, shown in figure 4.21-right.
Since the untransformed patch is a rectangular quad, small image space errors suggest that the optical mapping between the transformed and untransformed patch is linear. Furthermore, we can then conclude that a linear texture interpolation within the displayed transformed patch produces approximately correct results while being mapped (i.e., reflected and refracted) into image space. Consequently, we can say that the resulting image space errors describe a patch's curvilinearity at the representative pixel locations.
To decide whether or not a patch has to be further refined, we determine the largest of the four image space errors. If it is above a pre-defined threshold value $\bar{\delta}_{is}$ the patch has to be further refined and the RefineFurther returns *true*.

4.3.1.4.3 Computing Object-Image Reflections

To compute $\grave{v}_a$ from a given viewpoint $\grave{e}$, the object point $\grave{v}''$ and the optic's geometry is equivalent to finding the extremal Fermat path from $\grave{v}''$ to $\grave{e}$ via the optics. In general, this would be a difficult problem of variational calculus.
Beside ray- and beam-tracing approaches, several methods have been proposed that approximate reflection on curved mirrors to simulate global illumination phenomena within rendered 3D scenes. All of these methods face the above mentioned problem in one or the other way.
Mitchell and Hanrahan [Mit92], for instance, solve a multidimensional non-linear optimization problem for explicitly defined mirror objects ($g(\grave{x}) = 0$) with interval techniques. To compute reflected illumination from curved mirror surfaces, they seek the osculation ellipsoid that is tangent to the mirror surface, whereby its two focal points match the light source and the object point.

For a given viewpoint $\grave{e}$, Ofek and Rappoport [Ofe98] spatially subdivide the object space into truncated tri-pyramid shaped cells. In contrast to solving an optimization problem, they apply accelerated search techniques to find the corresponding cell that contains the object $\grave{v}''$ at interactive rates.
While Mitchell's multidimensional optimization approach is far from being applied at interactive rates, Ofek's search method does not provide the precision, required by an optical display.
In the following, we present a numerical minimization method to compute the object-image reflection for specific mirror surfaces (such as cones and cylinders). For such surfaces, we can reduce the optimization problem to only one variable. Consequently, our method provides an appropriate precision at interactive rates.
For the subsequent example we chose a cone-shaped mirror surface, since this surface type has also been used for our Virtual Showcase display (see subsection 5.5.4).

Cones and similar rotation bodies have the property that multiple surface points lay on a common plane. For example, all points on the straight line spanned by a cone's peak and an arbitrary point on its bottom circle lay on the same plane. Consequently, individual plane parameters can be determined for all angles around the cone's principle axis.

To determine an object's ($\vec{v}_d''$) image for a given viewpoint $\vec{e}$ and mirror surface $g(\vec{x}) = 0$, we first assume an arbitrary angle α around the cone's principle axis. We then determine the surface's tangent plane TP_1 at α by computing a surface point and the surface normal at α. Since $g(\vec{x})$ is an explicit function, we can compute the surface normal by using its first-order derivatives $g/(\partial\vec{x})$. Next, we reflect $\vec{v}_d''$ over TP_1 to its corresponding position within the image space and project this image onto the image plane, as described in subsection 4.3.1.4.2. In figure 4.22, the projected image point is outlined by $\vec{v}$.

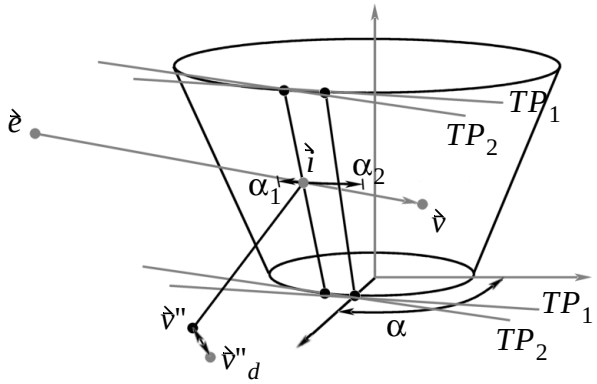


Figure 4.22: Object-image reflection via numerical minimization.

To verify the quality of our assumption, we reflect $\vec{v}$ back into the object space and project it onto the display surface. For a given $\vec{v}$, $g(\vec{x})$ and $\vec{e}$, a simple analytical solution exists to determine this image-object transformation: The ray spanned by $\vec{e}$ and $\vec{v}$ is intersected with $g(\vec{x})$ by solving a simple quadratic equation. The surface intersection $\vec{i}$, together with the normal vector at $\vec{i}$ (again determined using the surface's first-order derivatives) gives the tangent plane TP_2 at $\vec{i}$. Reflecting $\vec{v}$ and $\vec{e}$ over TP_2 and projecting the reflection of $\vec{v}$ onto the display surface using the reflection of $\vec{e}$ as center of projection, results in the point $\vec{v}''$. The image-object transformation is illustrated in figure 4.14. Note that for simplicity, the image-object transformation and the object-image transformation have been described as simple reflection/projection transformations. Normally, they incorporate refraction as well.

If the tangent plane at α produces the extremal Fermat path between $\vec{v}_d''$ and $\vec{e}$, the geometric distance Δ between $\vec{v}_d''$ and $\vec{v}''$ is zero, $TP_1 = TP_2$, and $v_a = \vec{v}$. Otherwise Δ is non-zero.

To approximate the correct α, we minimize the above-described function for Δ. Since this function depends only on α, we can apply fast numerical optimizers for one dimension. However, because we cannot easily derive its first order derivatives but it appears to be nicely para-

bolic near its minimum, we apply Brent's inverse parabolic interpolation [Bren73] with bracketing.
To speed up the minimization process (i.e. to reduce the number of function iterations), we can constrain the function range for a particular $\grave{\nu}_d"$. As illustrated in figure 4.22, the minimization is restricted to find α between α_1 and α_2. These boundaries can be determined as follows: Given $\grave{e}$ and the untransformed patch that belongs to $\grave{\nu}_d"$, we evaluate the projections on the mirror surface of the untransformed patch at the next lower LOD. The two angles α_1 and α_2 at the horizontal extrema of these projections are the corresponding boundary angles. Note that these projections are determined while transforming the patches (i.e., within TransformPatch-Vertices). Thus, for efficiency reasons, they are stored and looked up in the memory, rather than re-computing them again.
Our experiments showed that sufficiently precise solutions can be found after a small number of iterations. Typically we achieve average errors of $\Delta = 0{,}002mm$ with an average of 3 iterations. This value is still two orders of magnitude smaller than the smallest image space error we experimented with ($\delta_{is} = 0{,}1mm$).

4.3.1.4.4 Error Direction Propagation
Although the number of iterations is relatively small, the computational expenses of four minimization steps per patch result in a noticeable loss of performance. Especially while evaluating the large number of higher LOD patches, such an approach might not produce a speedup.
We also noticed that the direction in which the highest image space error (i.e. the one of the four patch sides where δ_{is} is maximal) propagates is the same for higher LOD patches that are derived from the same parent patch.
However, from which level of detail on this is the case depends on the display's properties. If, for instance, the optics produce well-behaved image deformations, the propagation direction of the highest error is consistent for relatively high LODs. If, on the other hand, the optics produces noisy images, the error direction alternates as randomly.
To benefit from this situation, we specify a LOD Λ depending on the optic's and the display surface's properties.
For patches that are below Λ, we determine the image space error as described above: We compute δ_{is} at all four edges and find the highest value. In addition, we record the error direction (i.e. the edge where δ_{is} is maximal) for each patch.

For patches that are above Λ we reuse the error direction of the corresponding parent patch and compute δ_{is} only for the edge in this direction, rather than at all four edges. By doing this, we assume that the largest error will occur in the same direction as for the parent patch. Consequently, we reduce the number of minimization steps from four to one for all patches that are above Λ -i.e., for the majority of all relevant gird patches.
Our experiments have shown that this heuristic leads to a significant speedup while producing the same final output.

4.3.1.4.5 Projected Ptach Size
In addition to the image space error, the size of projected patches can be used to derive a further refinement criterion: Patches do not have to be refined further if their size falls below a

certain *projected patch size threshold* $\bar{\Sigma}_{pp}$. To compute the area of a patch we can apply heron's formula for projected triangle areas twice:

$$\Sigma_{pt} = \sqrt{(p-a)(p-b)(p-c)} \tag{4.4}$$

where a, b, c are the lengths of the three triangle edges and $p = (a+b+c)/2$. Note that the time consuming square-root is not required, if compared with the squared pixel size.
In contrast to the image space threshold, the projected patch size threshold is constant and will not be varied from frame to frame. We rather use this refinement criterion to eliminate patches that are too small to cause a visual improvement of the rendered image, but require the same computational cost as any other patch.

4.3.1.5 Display Specific Components

While our algorithm is valid for other displays that require non-linear pre-distortion, its display specific components have been explained based on a particular mirror display. The nature of the additional mirror optics makes the transformation of the grid patches and the computation of the resulting displacement error fairly complex. In fact, the implementation of the TransformPatchVertices and the RefineFurther functions have been explained with an emphasize on our example display. These two functions represent the display specific components of our algorithm. While for a mirror display, TransformPatchVertices and RefineFurther have to consider laws of geometric optics (such as reflection and refraction transformations) to map grid vertices from the image plane onto the projection surface and vice versa, they can be generalized to do the same for other displays without modifying the general algorithm.
If, for instance, the algorithm is used to project images onto curved screens (e.g., a cylindrical or spherical projection device), TransformPatchVertices would incorporate only projection transformations (i.e., it would only determine intersections of vertex-individual geometric lines of sight with the display surface). The resulting displacement error that is computed by RefineFurther can then be determined within the object space (e.g., the screen space), rather than within an image space. Compared to our numerical approach for convex mirror displays, this would be less complex since it involves only simple intersection computations for which analytical solutions exist. If a view-dependence is not required, TransformPatchVertices and RefineFurther could also retrieve pre-computed values from a look-up table.

4.3.2 Progressive Rendering

In this section we focus on the image resolution. We introduce a progressive rendering method that determines the highest image resolution possible within a desired frame rate based on the techniques described in section 4.2.2 and in conjunction with the selective refinement algorithm introduced in section 4.3.1.
Considering Funkhouser's classification [Fun93], our progressive method is *reactive* (i.e., based only on the time required to render the previous frame) rather than *predictive* (i.e., based on complexity of the scene to be rendered in the current frame). Note that a reactive approach that is based on a feed-back loop is sufficient in our case, since rendering-time estimations for single scene objects (as suggested by Funkhouser [Fun93] and others) would require access to the application's scene geometry. However, our image-based approach is independent of the scene geometry to address the problems discussed in section 4.2.1.2.

4.3.2.1 Progressive Refinement

Assuming that the complexity of the scene that is rendered during the first pass cannot be influenced by our approach, the performance of the image-based two-pass method mainly depends on two factors: The time needed to transform the grid (this is affected by the grid size gw, gh [11]) and the time needed to copy the image from the frame-buffer into the texture memory and to finally map it onto the grid (this is affected by the image resolution tw, th). Consequently, the method's order of growth can be estimated by $O(gw \times gh) + O(tw \times th)$ (see section 6.2 for an overview of the technique's computational cost and order-of-growth).

The RecursiveGridRefinement algorithm described in section 4.3.1.3 generates the minimal image geometry that prevents a predefined image space error, and consequently approaches to minimize the grid size factor.

The ProgressiveRefinement algorithm that is introduced below progressively modifies the image space error threshold $\bar{\delta}_{is}$ (to adapt to the grid complexity) and the image resolution tw, th depending on the desired rendering time Δ_T.

ProgressiveRefinement($\Delta_T, \varepsilon_{\bar{\delta}_{is}}, min_{\bar{\delta}_{is}}, max_{\bar{\delta}_{is}}, min_{tw}, max_{tw}, min_{th}, max_{th}$ ***):***

1: **begin**
2: GetTime(T_1)
3: GenerateImage($\grave{e}, \grave{p}, \theta, tw, th$)
4: GenerateImageGeometry($\grave{e}, \grave{p}, \theta, gw, gh$)
5: RecursiveGridRefinement(0, 0 , gw, gh)
6: RenderImage($\grave{e}, dw, dh, ww, wh$)
7: GetTime(T_2)
8: **begin** update grid refinement parameters
9: **if** $(T_2 - T_1) < \Delta_T$ | viewpoint and scenery have not changed
10: **if** $\frac{\bar{\delta}_{is}}{R_g} \geq \varepsilon_{\bar{\delta}_{is}}$: decrease image space error threshold: $\bar{\delta}_{is} = \frac{\bar{\delta}_{is}}{R_g}$
11: **else**
12: **if** $\bar{\delta}_{is} \cdot R_g \leq \frac{1}{\varepsilon_{\bar{\delta}_{is}}}$:increase image space error threshld: $\bar{\delta}_{is} = \bar{\delta}_{is} \cdot R_g$
13: **endif**
14: **end**
15: **begin** update image refinement parameters
16: **if** $\bar{\delta}_{is} \leq min_{\bar{\delta}_{is}}, tw \cdot R_i \leq max_{tw}, th \cdot R_i \leq max_{th}$
17: increase image resolution: $tw = tw \cdot R_i, th = tw \cdot R_i$
18: initialize $\bar{\delta}_{is}$
19: **endif**
20: **if** $\bar{\delta}_{is} \geq max_{\bar{\delta}_{is}}, tw/R_i \geq min_{tw}, th/R_i \geq min_{th}$
21: decrease image resolution: $tw = tw/R_i, th = th/R_i$

11.This assumes the worst case in which the entire grid is used, rather than a regionally refined version.

```
22:          initialize δ̄is
23:       endif
24:   end
25:end
```

Two time stamps are defined in lines 2 and 7 - measuring the start and the end time of each frame. In between these time stamps, the first rendering pass is carried out (line 3), the grid geometry is generated and recursively deformed (lines 4 and 5), and finally, the second rendering pass is performed (line 6).

The part of the algorithm that includes lines 8 through 14 modifies the image space error threshold $\bar{\delta}_{is}$ (which, in turn, affects the grid complexity as described in subsection 4.3.1) as follows:

If the time that was required to render the previous frame is below the desired rendering time Δ_T, or the viewpoint and the virtual scenery have not changed[12], then $\bar{\delta}_{is}$ is decreased[13] (line 10). This leads to an earlier refinement of patches with a lower curvature and consequently to a higher density of patches drawn. Note that if the viewpoint and the scene do not change, the image quality is successively improved from frame to frame. Thus, the rendering time increases and the frame rate might fall below the desired frame rate. However, since no perspective changes occur, this will not be noticed by the observer. After moving the viewpoint or modifying the scene in this situation, both -the image space error threshold and the image resolution are set back to "interactive" values to ensure a fast response. These values can be found, for instance, by determining the average image resolution and image space error threshold that are continuously tracked during viewpoint movements and scene modifications. Note that this is not outlined in the above algorithm.

If, however, the previous render time is below Δ_T and the viewpoint or the scene have changed, then $\bar{\delta}_{is}$ in increased[14] (line 12) to cause the reverse effect. Increasing and decreasing $\bar{\delta}_{is}$ is achieved by multiplying or dividing $\bar{\delta}_{is}$ by the grid refinement function R_g. Note that the outcome of this function may depend on different parameters (such as $\bar{\delta}_{is}$ itself and/or $(T_2 - T_1)$ and Δ_T).

However, if the algorithm has reached a stage where the grid becomes too coarse, the edges between the patches can be recognized and texture errors such as the ones discussed in subsection 4.3.1 are produced. If in such a case the desired rendering time has still not been reached, the image resolution is decreased[15] to increase the grid resolution as the result of a side-effect. If, on the other hand, the grid geometry is complex but the desired rendering time has not been exceeded, the image quality is improved by increasing its resolution[16]. This image resolution

12. If head-tracking is applied, this can be determined by evaluating whether the current viewpoint has been moved out of a spherical area that is centered around the position of the previous viewpoint (this allows to cope with small tracking noise).

13. But no further than a defined smallest value $\varepsilon_{\bar{\delta}is} \leq 1$.

14. But no further than the maximum value $\frac{1}{\varepsilon_{\bar{\delta}is}}$.

15. But no further than a smallest image resolution $min_{tw} \times min_{th}$.

16. But no further than a largest image resolution $max_{tw} \times max_{th}$.

refinement criterion is outlined in lines 15-24 of the ProgressiveRefinement algorithm: If $\bar{\delta}_{is}$ is below a minimal threshold value $min_{\bar{\delta}is}$ (i.e., the grid is complex because low-curvature patches are refined early), then the image resolution is increased and $\bar{\delta}_{is}$ is set back to its initial value (lines 16-18). If $\bar{\delta}_{is}$ is above a maximal threshold value $max_{\bar{\delta}is}$ (i.e., the grid is coarse because high-curvature patches are refined late), then the resolution is decreased and $\bar{\delta}_{is}$ is initialized. Note that image resolution refinement function R_i may also depend on different parameters (such as $\bar{\delta}_{is}$ and/or $(T_2 - T_1)$ and Δ_T). In the case that $min_{\bar{\delta}is} = max_{\bar{\delta}is}$ then only the image resolution is refined while the image grid geometry remains constant (depending on the initial value of $\bar{\delta}_{is}$).
The ProgressiveRefinement algorithm prevents the image geometry from becoming too coarse and from producing visual artefacts between patches during rasterization that are caused by the linear texture mapping. Consequently, the image space error threshold is varied within a range ($\bar{\delta}_{is} = [min_{\bar{\delta}is}, max_{\bar{\delta}is}]$) that does not affect the image appearance. Varying the image resolution, however, does always affect the appearance of the image (but does not produce artefacts) and can be visually detected by the observer. This is the reason for modifying the image resolution only if a further simplification of the image geometry would cause visual artefacts.
Examples of refinement functions that are used for grid geometry refinement and image resolution refinement are described in subsection 4.3.2.2.

4.3.2.2 Refinement Functions

As mentioned above, a refinement function that modifies a specific value x (such as $\bar{\delta}_{is}$ or tw, th in our case) depends on different parameters. The choice of this function influences the dynamic behavior of the progressive refinement over time (e.g., how fast a desired frame rate can be approached and how stable it can be kept).
Rusinkiewicz's QSplat rendering method [Rus00], for instance, applies concatenated square-roots ($x' = x\sqrt{\sqrt{x}}$ and $x' = x/(\sqrt{\sqrt{x}})$) as a refinement function for progressively displaying meshes as projected points (so-called splats) based on a bounding sphere hierarchy of the mesh. This function approaches a pre-defined frame rate quickly and reacts to changes of the frame rate fast. For a continuously changing viewpoint or scenery (e.g., in case of animated virtual scenes), however, it is difficult to keep a stable image quality once the desired frame rate has been reached.
To overcome this problem, we evaluated several other potential grid refinement functions and found that linear functions (such as $x' = ax$ and $x' = x/a$, where $a \geq 1$ is a constant) are better suited for our approach than higher order functions. Linear functions are not as responsive to frame rate changes, but provide a higher stability once the frame rate has been reached. This is especially useful, when viewpoint changes occur more frequently (eventually continuously, as in the case of a head-tracked viewer) rather than sporadically (as in the case of the mouse-controlled desktop viewer, that is used for QSplat [Rus00]).
However, to support both -a fast response to frame rate changes and an appropriate stability of the image quality, once the desired frame rate has been reached- we can combine two (or more) different grid refinement functions by taking the deviation of the current rendering time from the desired time into account:

In the case the viewpoint and the scene do not change, we can always apply a fast but unstable refinement function R_{gf} (e.g., $R_{gf} = 1/(\sqrt{\sqrt{x}})$, or $R_{gf} = a/x$, whereby a is the constant that causes the highest image/grid resolution within a single step -such as $a = \varepsilon_{\bar{\delta}is}$ for the image space error threshold or $a = max_{tw} = max_{th}$ for the image resolution) to quickly improve the image quality.
Note that because of the image space error threshold boundaries $[min_{\bar{\delta}is}, max_{\bar{\delta}is}]$ that are discussed above, the image space error threshold is always less than one in our case.
In the case the viewpoint or the scene do change and the previous render time $T_2 - T_1$ is below Δ_T, we can concatenate the fast refinement function R_{gf} with a stable refinement function R_{gs} (e.g., $R_{gs} = a$) as follows:

$$R_g = R_{gf}\left(1 - \frac{(T_2 - T_1)}{\Delta_T}\right) + R_{gs}\left(\frac{(T_2 - T_1)}{\Delta_T}\right) \tag{4.5}$$

If $T_2 - T_1$ differs highly from Δ_T, R_{gf} is the dominant function. Consequently, the refinements are coarse. The closer $T_2 - T_1$ approaches Δ_T, the more dominant R_{gs} becomes (i.e., the refinements become finer).
In the case that the viewpoint or the scene do change and the previous render time $T_2 - T_1$ is above Δ_T, we need to compute the weights slightly different to achieve the same effect:

$$R_g = R_{gf}\left(1 - \frac{\Delta_T}{(T_2 - T_1)}\right) + R_{gs}\left(\frac{\Delta_T}{(T_2 - T_1)}\right) \tag{4.6}$$

We can apply the concatenated refinement function to refine the image resolution tw, th as well. However, since tw, th are always greater than one, our exemplary fast refinement function has to be reversed to $R_{if} = \sqrt{\sqrt{x}}$, while our exemplary stable refinement function remains the same[17]: $R_{is} = R_{gs} = a$.
Note that concatenation of multiple refinement functions is not directly outlined in the ProgressiveRefinement algorithm (lines 10, 12 and 17, 20 would be affected by this)
The presented refinement functions are evaluated in section 6.4.2.

4.3.3 Parallel Processing

Since image generation and image warping (i.e., the generation, transformation and refinement of the image geometry) are completely independent processes, an overall performance speed-up can be achieved by carrying them out in parallel.
In contrast to Hoppe's progressive meshes[Hop96, Hop97, Hop98], we do not aim to progressively transmit refinement steps or different LODs of the image geometry over the network, but rather to support an efficient selective refinement on a local machine. On the one hand, our internal LOD structure and grid representation (in form of look-up tables, etc.) are too large to

17.A different constant a may be used for image resolution and grid refinement.

be transmitted over a network in real-time. On the other hand, they can be efficiently accessed and modified locally.
Therefore, we decided to outsource the image generation step to another processor (*host*) and to keep the image warping steps (represented by the GenerateImageGeometry and RecursiveGridRefinement algorithms) and the final rendering step on the initial processor (*client*). This is illustrated in figure 4.23 for one frame.

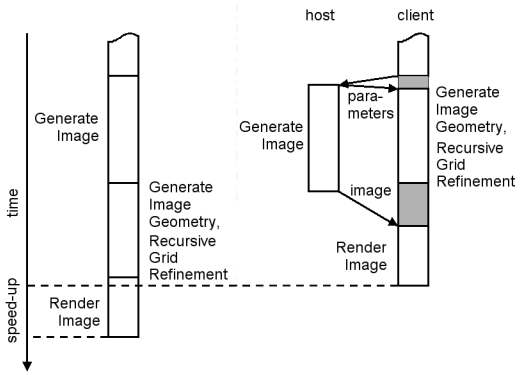


Figure 4.23: Sequential processing (left) and parallel processing (right).

With respect to the right image of figure 4.23: The client (that has information about the optics, the projection screen and the viewer) first sends the parameters that are required to carry out the image generation step to the host (i.e., $\grave{e}, tw, th$). The host (that has information about the scene which has to be rendered) acknowledges this by returning the parameters that are required to carry out the image transformation steps (i.e., $\grave{p}$, θ). After this, both processes (i.e., ImageGeneration on the host, and GenerateImageGeometry and RecursiveGridRefinement on the client) are carried out in parallel. Finally, the generated image is transferred to the client and RenderImage is carried out.
If the two processors are located on different machines that are connected by a network, the parameters and the generated images have to be transmitted from one node to the other. Consequently, additional communication time is required. If, however, both processors are part of the same machine (e.g., a two-processor Pentium or a multi-processor workstation), a shared memory block (or other inter-process communication mechanisms) can be utilized for the data transmission. In this case, the transmission time can be reduce.
Note that the gray parts on the client site represent idle-times. Compared to the initial sequential processing approach (left image of figure 4.23), we can see that a certain performance speed-up can be achieved if the data transmission time (the time to send the parameters and the time to send the image between the two processor nodes) is smaller than the time that can be saved by running both processes in parallel instead of sequentially. However, since the client still applies our progressive rendering method (this is not illustrated in figure 4.23), the transmission time is added to the actual rendering and transformation times. Consequently, the communication time (e.g., the network traffic, if communicating over a network) automatically influences the image quality in terms of being able to keep the desired rendering time on the client site.
The advantage of this approach is not only a speed-up that can be gained by running two processes in parallel, but also the possibility that a remote rendering is supported. In the case of remote rendering, high-performance computers can act as remote hosts to render complex

scenes while low-cost personal computers can be used as local clients to transform and display the generated image. However, a high-speed network is required to provide an efficient image transmission.
The transmission time can be reduced by efficiently encoding the image before it is transmitted. However, the encoding/decoding process should not require more time than the time that can be saved by transmitting the encoded image instead of the original one. In addition, the encoding/decoding method should not transmit a large amount of additional data (such as probability tables) that might be needed to decode the image. A more problem specific coding method that is optimized for optical see-through presentations and does not reduce the image quality is introduced below: Instead of encoding the entire image with three bytes per pixel (according to our RGB image-representation), we can encode an image row/column-wise (cf. figure 4.23): We use a byte stream that indicates the first appearance of a non-black pixel (using four bytes to store the x/y image-position of that pixel), then storing all following colored pixels (using three bytes for the RGB color values of each pixel) until the a black pixel is reached. In this case, we store three zero-bytes to indicate the termination of a color sequence and search for the launching pixel of the next color sequence that is attached to our byte stream in the same manner.

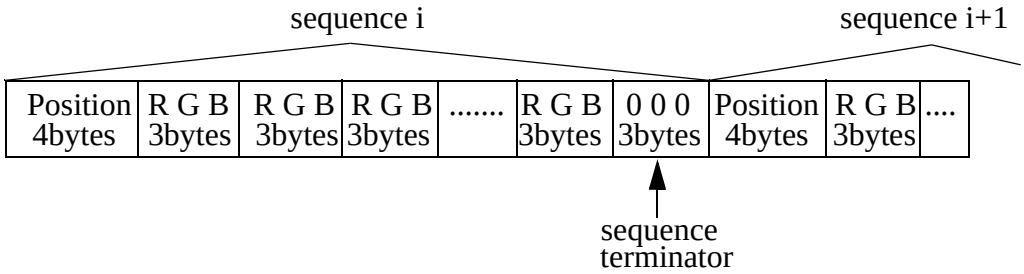


Figure 4.24: Color-sequence-based image coding method that does not reduce the image quality.

This representation allows us to transmit only non-black pixels. As discussed in section 4.2.2.3, we chose black as background color to cause transparency within the reflected black areas of the image.
In addition, we have implemented a run-length method (cf. figure 4.25) that stores three bytes for an RGB pixel and one byte that contains the length of a run of pixels with the same or similar color.

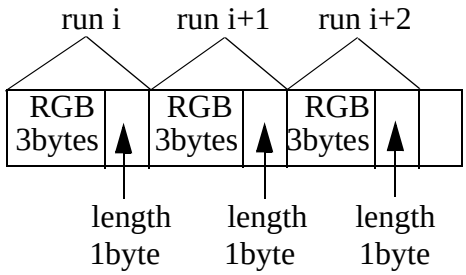


Figure 4.25: Run-length-based image coding method that reduces the image quality.

The next pixel that differs from the previously stored pixel by a certain amount creates a new run (i.e., another four bytes to store the RGB and the length values). Since the RGB values are

defined within the $\Re^3$, the color-deviation between two pixels can be expressed with the distant function $\left|[r_1, g_1, b_1] - [r_2, g_2, b_2]\right|$. Note that for performance reasons, a square root should not be computed for determining the distance between two RGB vectors.
A consistent black background is also fully eliminated by this method. The degree of compression of the image content can be controlled with the color-deviation threshold $\overline{cd}$. However, if $\overline{cd}$ is too large the image quality is visibly reduced. The reason for this is the color homogenization that is applied by this method.
Section 6.5 presents an evaluation of a parallel processing experiment with two PCs that were connected by a 100MBits/s LAN. In addition, we analyze our image coding approaches and compare it with the adaptive Huffman coding algorithm [Knu85].

4.4 Non-planar Projection Surfaces and Multiple Projections

So far, we have assumed that a single projector beams the generated image onto a single planar display surface. For all introduced transformation methods, an affine perspective off-axis projection has been applied. However, these methods can also be used for non-planar display surfaces and multiple projectors, by simply exchanging the final way of projection.
In the following paragraphs, we give examples of how our rendering approaches can be extended towards non-planar projection surfaces and multiple projections. We have to differentiate between the affine methods for absolute image forming systems, and the curvilinear methods for non-absolute optical systems.
As discussed in section 3.1.2.2, Raskar, et al apply projective texture-mapping [Seg92] to support single or multiple front projections onto a multi-plane or curved display surface as one component of a three-pass rendering method [Ras98a, Ras98b]. Projective textures utilize a perspective texture matrix to map projection-surface vertices into texture coordinates of those pixels that project onto these vertices. Raskar's first rendering pass generates an image of the virtual scene that will look perspectively correct to the user. During the second pass, this image is projected out from the user's current point of view onto a registered virtual model of the display surface -using projective texture-mapping. Finally, the textured model is rendered from the projector's point of view and the result is beamed onto the real display surface. If multiple projectors are used, the second pass have to be repeated for each projector individually. The generated images have to be geometrically aligned and color and edge blended appropriately to realize a seamless transition between them [Ras98a]. This is done within Raskar's third pass.
To support absolute optical elements (such as planar mirrors) that require affine model and view transformations, Raskar's method can be slightly modified: Instead of rendering the original scene from the observer's actual viewpoint, the transformed scene has to be rendered from the transformed viewpoint. The transformations that are applied to the scene and the viewpoint depend on the optics (e.g., the reflected model view transform discussed in section 4.1.1, or the reflected model-view transform discussed in section 4.1.2). Raskar's other rendering passes remain unchanged. Note that if multiple planar display surfaces are used, and if one projector is assigned to one projection plane, projective textures and multi-pass rendering are unnessecary. Instead, regular multi-pipeline rendering can be applied (as it is done for surround-screen projection systems, such as CAVEs [Cru93] or multi-sided workbenches [Tan01c, Bar01b]).
Considering non-absolute image forming systems, we can see that Raskar's first rendering pass is similar to our first pass (outlined by the GenerateImage algorithm). In our case, however, the generated image is additionally transformed with respect to the applied optics (i.e., by the refracted image transform, image-based reflected model-view transform and implicit projected image transform). If we want to project the transformed image onto a non-planar sur-

face, the projection matrix *P* and the perspective division that are used by the ImageBasedReflectedModelViewTransform algorithm can not be applied. However, projective texture-mapping cannot be applied either, because multiple centers of projection exist (one for each transformed vertex). Instead, the intersections of the geometric lines of sight, that are spanned by the transformed image-grid vertices and their corresponding transformed viewpoints with the display-surface model, have to be computed. Transforming the image-grid vertices to their intersection points equals a projection from their individual centers of projection (i.e. their individual viewpoints). The projected image can then be rendered from the perspective of the projector(s), as proposed by Raskar. However, plane-dependent 2D computations (such as the ones needed by the implicit projected image transform, the recursive grid refinement, and the progressive refinement) have to be carried out on the projector's image plane (i.e. normalized to the coordinate system of an arbitrary plane, perpendicular to the projector's optical axis), or transformed image plane (i.e. within the optic's image space) respectively.

4.5 Optical Chains

In general, we have introduced our transformation components in the following order: refraction transformations, reflection transformations, projection transformations and projector predistortion (i.e. projected image transformations).
In section 4.1.5, we have already seen, that the transformations have to be repeated for every single optical element individually. However, the optics described in section 4.1.5 was assumed to be convex, thus the optical path was folded only once by each element before it reached the projection plane.
If multiple optical elements form an optical chain (i.e., the optical path is folded multiple times before it reaches a projection surface), the refraction transformations (i.e., refracted model transform and refracted image transform) and/or the reflection transformations (i.e., reflected view transform, reflected model-view transform, geometry-based reflected model-view transform, and image-based reflected model-view transform) have to be applied more than once to the scene or image vertices. The transformations' properties and their application sequence are defined by the properties of optical elements and their constellation within the optical chain.
For curvilinear transformations (i.e., the ones required for curved mirrors and for lenses in general), new intersections of the geometric lines of sight (or the light rays) with the corresponding surfaces have to be computed for each surface of each optical element. This is necessary to determine the element's individual properties (such as normal vectors, tangent planes, etc.) that are required for the corresponding transformation. The refracted image transform itself represents a good example of such a case. Here, two surface intersections have to be computed to determine the in-refractor and the out-refractor.
For affine transformations (i.e., the ones used by planar mirrors), no intersection computations are required. Note that affine and curvilinear transformations can be mixed (as it is the case for reflection and refraction of planar mirror beam-splitters). After all necessary transformations have been applied, the geometry is finally projected onto the display surface.
We use a structure, similar to ray-trees (used for ray-tracing) as an internal representation for optical chains. The ray-tree's root represents the current view-point, its intermediate nodes represent optical elements, and its leaf nodes represent projection surfaces. In contrast to ray-tracing, our ray-tree does not have to be built explicitly for each ray during rendering but rather can be pre-defined. The reason for this is that the optical path from the viewpoint, over the intermediate optics to the projection surfaces is defined by the constellation of the optical elements. At each intermediate node, we allow a finite number of alternatives that are represented by multiple branches of our tree. This allows, that the optical path can be splitted by an optical

element and that different optical paths (i.e., multiple sub-chains) are possible. This is for example the case, if two subsequent mirrors can be seen as reflection in a third mirror.
By recursively traversing the ray-tree for each scene or image vertex, surface intersections are computed if necessary and individual transformations are applied to the current vertex at each intermediate node. If an intermediate node offers multiple branches, each alternative is tested by determining the subsequent element which is intersected by the current geometric line of sight. To speed up the intersection tests, we apply a bounding volume hierarchy as an approximation of the optical elements. Note that bounding volume hierarchies are also a common method for ray-tracing [Gla84]. If a leaf node is reached, the projection transformations and the projector pre-distortion are finally applied.

4.6 Summary

In this chapter, we have introduced several interactive rendering techniques that can be applied for PBAR configurations built from planar or curved optics and displays. An overview of the described techniques' computational cost and order-of-growth is presented in section 6.2.
While absolute optical systems (i.e., planar mirrors) provide an affine mapping from object space to image space, affine geometry transformations have been integrated into traditional transformation pipelines to benefit from off-the-shelf acceleration hardware. Affine geometry transformations do not cause additional computational cost. Consequently, the rendering time for the introduced techniques which support such optics only scales up with the number of applied rendering passes (e.g., in the case of multi-section optics and/or multiple observers, as illustrated in figure 4.10, for example).
For optical elements that require curvilinear transformations, however, an image-based approach is more efficient and more flexible than a geometry-based approach.

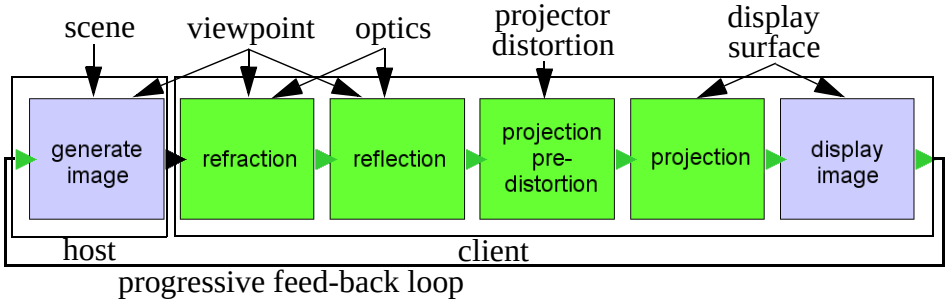


Figure 4.26: The image-based rendering pipeline.

Our final image-based approach (cf. figure 4.26) avoids a direct access to the scene geometry, and consequently prevents the time-consuming transformations of many scene vertices. In addition, it is not restricted to a geometry-based first rendering pass, but rather supports any image generating rendering method[18]. The introduced approach applies a sequence of optional non-affine image transformations (outlined in green), which we currently consider most efficient for non-stigmatic PBAR displays. It can be seen as a mixture between the extended camera concept [Loe96] and projective textures [Seg92]. While projective textures utilize a perspective texture matrix to map projection-surface vertices into texture coordinates of those pixels that project onto these vertices, our method projects image vertices directly on the projection surface while remaining their texture coordinates[19]. This is necessary because curved mirrors require a different center of projection for each pixel. Using individual projection

18.Note that arbitrary image generation methods are also supported if affine transformations are applied in combination with absolute optical systems.

parameters for each pixel, however, is the fundamental idea of the extended camera concept - although originally applied for ray-tracing. Here, the origin of primary rays passing through a pixel of the image plane depends on the pixel location itself. Thus the primary rays are not required to emerge from a single point (perspective projection) or to lie in a plane (orthogonal projection). The modified rays are traced through the scene in the usual way and result in color values for each pixel. The final image presents a distorted projection, according to the ray modification function. The main difference to our approach is that the extended camera concept generates a deformed image via ray-tracing, i.e., each pixel is generated from a modified primary ray. Our method deforms an existing image by projecting it individually for each pixel.

For displays that correct non-linear distortion by applying multi-pass rendering, generating appropriate regional levels of detail instead of applying uniform grids or constant image geometries allows to consider the error that is caused from a piecewise linear texture interpolation and to minimize it by adapting the underlying geometry. For this purpose, we have presented an adaptive grid refinement algorithm for real-time view-dependent image warping. It can be applied to neutralize optical distortion or to display undistorted images onto non-planar surfaces. We have used the example of cone-like Virtual Showcase displays to explain our algorithm, and to give concrete examples for the implementation of display specific components that are required by the algorithm. For other displays, these components have to be adapted. In particular, we have presented a method for object-image reflections (subsection 4.3.1.4.3) that was optimized for particular second order mirror surfaces, such as cones or cylinders. For other mirror surfaces, this method has to be replaced. On the one hand, the refinement algorithm prevents oversampling and texture artifacts. On the other hand, it speeds up rendering for such displays significantly while guaranteeing a maximal error on the image plane. Beside the displacement error, other criteria (such as the projected patch size, described in subsection 4.3.1.4.5) are evaluated and concatenated to define the final exit condition of the recursive refinement procedure.

The image-based approach is flexible enough to be smoothly integrated into existing software frameworks, and it is general enough to support different hardware setups (i.e., projection devices and optics). Additionally, it takes as much advantage of hardware-implemented rendering pipelines as currently possible and supports selective refinement, progressive rendering, and parallel processing. While the rendering passes and per-primitive transformations (outlined in blue) can be completely executed by graphics accelerators of today's graphics adapters, intermediate per-vertex transformations are not supported by prevalent rendering pipelines (such as the one implemented in OpenGL). Consequently they cannot benefit from current graphics acceleration hardware. If realized in software, these transformations tax the main CPU and memory bandwidth. Next generation graphics engines, e.g., [Lin01, Nvid01a], support programmable per-vertex operations, which will allow hardware acceleration of the required per-vertex transformations.

19.Texture coordinates have to be recomputed only if the texture size changes so that it does not fill out the assigned texture memory completely.