Fang, Yi

„Stereoskopische Visualisierung technischer Daten mit Java und JOGL“

eingereicht als

Bachelorarbeit

an der

HOCHSCHULE MITTWEIDA(FH)
UNIVERSITY OF APPLIED SCIENCES

Fachbereich: Mathematik-Naturwissenschaft-Informatik

Flöha, Juli 2011

Fang, Yi

„Stereoskopische Visualisierung technischer Daten mit Java und JOGL“

eingereicht als

Bachelorarbeit

an der

HOCHSCHULE MITTWEIDA(FH)
UNIVERSITY OF APPLIED SCIENCES

Fachbereich: Mathematik-Naturwissenschaft-Informatik

Flöha, Juli 2011

Erstprüfer: Prof. Dr. –Ing. Mario Geißler

Zweitprüfer: Dr. -Ing. Bernhard Sünder

Drittprüfer: Dipl. –Informatiker(FH) Daniel Stockmann

Bibliographische Beschreibung

Yi Fang:

„Stereoskopische Visualisierung technischer Daten mit Java und JOGL“, Gesellschaft für angewandte Mess- und Systemtechnik 2011

Mittweida, Hochschule Mittweida (FH), Fachbereich Mathematik – Naturwissenschaften – Informatik, Bachelorarbeit, 2011

Referat

Ziel der Bachelorarbeit ist es, für die Software jBEAM ein Modul „stereoskopische Visualisierung“ basierend auf den bestehen JOGL 3D Grafiken zu entwickeln. In den folgenden Kapiteln werden die Grundkenntnisse der Stereoskopie und ihre Realisierungsweise erläutert.

Danksagung

Besonderer Dank gilt Herrn Stefan Thämming und Herrn Daniel Stockmann, die mir diese Bachelorarbeit ermöglichen.

Für die hervorragende Betreuung danke ich Herrn Dr. Bernhard Sünder, der als Geschäftsführender Gesellschafter der Firma in diese Bachelorarbeit direkt involviert war.

Herrn Prof. Dr. Mario Geißler danke ich für die intensive Betreuung. Seine Konkretisierung der vorliegenden Ausarbeitungen hatte entscheidenden Anteil am Erfolg dieser Arbeit.

Allen anderen, nicht namentlich erwähnten Personen, danke ich für die effektive Zusammenarbeit und ihre Unterstützung.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Themenbedeutung

Das Thema „Stereoskopische Visualisierung technischer Daten mit Java und JOGL“ wird im Auftrag der Firma „Gesellschaft für angewandte Mess- und Systemtechnik“ ein Software-Modul entwickelt.

Die Stereoskopie ist seit langen bekannt. Sie fand zuerst in der Fototechnik und später mit der stereoskopische Visualisierung im 3D-Kino breite Anwendung. Seit eingen Jahren gibt es Displays, die eine 3D-Darstellung ermöglichen. Aber die Stereoskopie wird nur von wenige Software unterstützt, Bsp. Maya von AutoDesk, Dassault Systemes CATIA usw..

Die Bachelorarbeit untersucht eine Lösung für die stereoskopische Visualisierung mit JOGL und jBEAM.

Die Software jBEAM ist die neue Generation der Analyse- und Visualisierungssoftware. Sie hat einen Modul „Universal 3D-Graphic (OpenGL)“, mit dem man die „ 3D-Points Diagram, 3D-Surface Diagram, 3D-Waterfall Diagram und 3D-Bar Diagram“ zeichnen kann. Die Abbildung 1.1 zeigt der universelles 3D-Grafik mit einem oberfläche Diagramm.

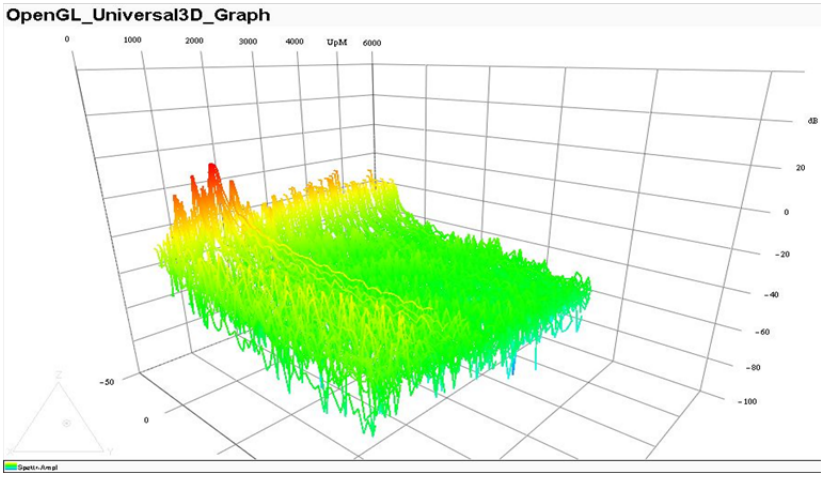


Abbildung 1. 1: Universelles 3D-Diagramm[1]

Der Modul „Universal 3D-Graphic (OpenGL)“ soll mittels Java und JOGL für eine stereoskopische Visualisierung erweitert werden.

1.2 Aufgabenbeschreibung

Aufgabe dieser Bachelorarbeit ist es, die stereoskopische Visualisierung meß-technischer Daten mit Java und JOGL zu realisieren.
Im Kapitel 2 werden die Grundlagen der Stereoskopie vorgestellt. Kapitel 3 beschäftigt sich mit einer Lösung zur Realisierung der stereoskopischen Visualisierung mit JOGL.

Im Kapitel 4 werden wichtige Aspekte im Kontext dargelegt.

2 Stereoskopische Visualisierung

2.1 Stereoskopie

Mit den Augen kann man ein Bild sehen, das ein Bild die eigenen Level und Depth enthält. Man spielt häufig ein 3D Computer Spiel oder sieht häufig 3D Filme zu Hause. Aber ist es leider kein „True 3D“. Weil die meisten Bildschirme 2D sind, werden die Bilder auf diesen Bildschirmen exportiert. Diese sind „3D-Ebene“ genannt.

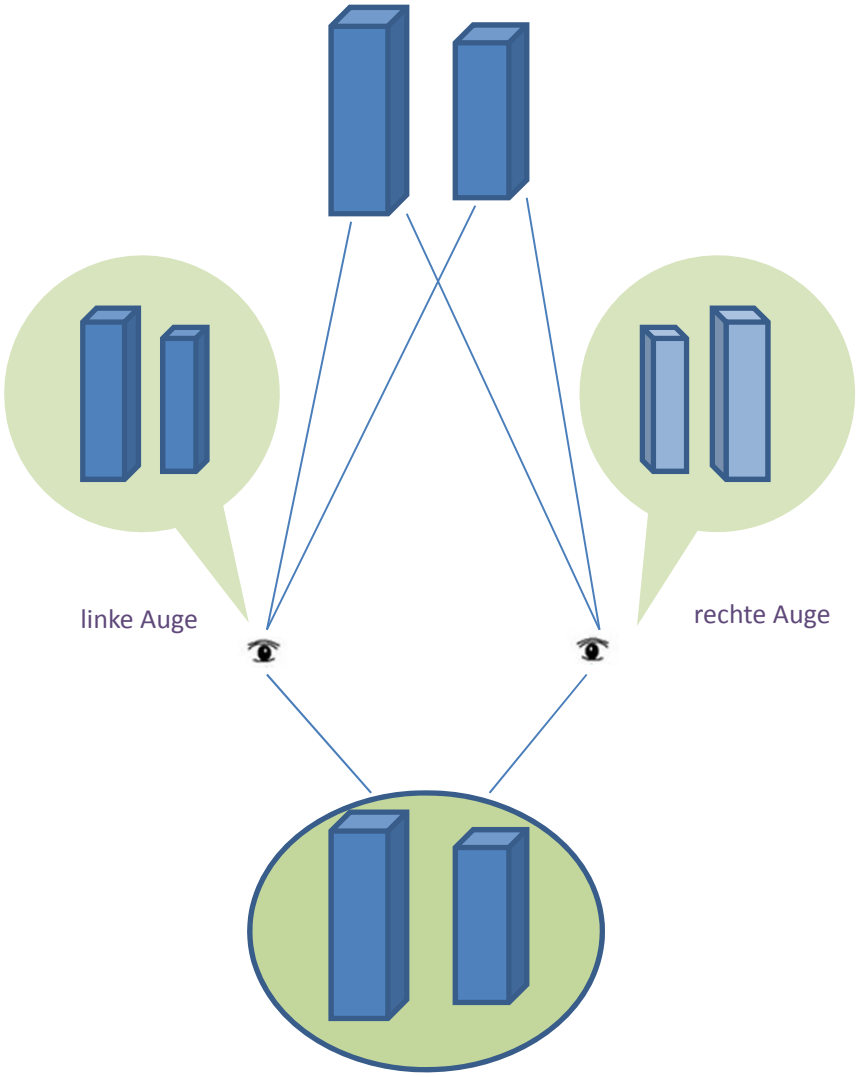


Abbildung 2. 1: Das Prinzip der True 3D (eigene Darstellung)

Wenn ein Bild mit zwei Augen zu sehen ist, werden zwei oder mehre Aussichtspunkte beobachtet. Dadurch werden die verschiedenen wahrgenommenen Bilder genommen. Durch das Triangulationsprinzip wird die Parallaxe berechnet und werden die dreidimensionalen Informationen der Bilder gekriegt. Der Vorgang ist das Prinzip der stereoskopische Visualisierung.

Durch die Stereoskopie Technologie wird die Parallaxe auf dem Bildschirm angezeigt. Damit können die Zuschauer ein „True 3D“ sehen. Dadurch bekommen unsere zwei Augen verschiedene Bilder. Das Prinzip ist in der Abbildung 2.1 dargestellt.

Wie sind die stereoskopischen Bilder zu sehen? Das wird eine spezifische 3D Brille und Umfang benötigt. In dem folgenden Abschnitt werden die 3D Brillen und ihre Technik erklärt.

2.2 3D Technik

Stereoskopische Visualisierung besteht aus zwei Weise, „mit Brille“ und „ohne Brille“. „Ohne Brille“ Technologie wird in der Handy oder dem Business verwendet. „Mit Brille“ Technologie wird in den Monitoren, der Fernsehen oder den Beamern in den Familien oder Kinos verwendet. Bisher werden zwei 3D Technik, aktiv Stereo und passiv Stereo, für die Verwendung einer Brille genutzt. D.h., es werden unterschiedliche 3D Brillen und Bildprojektionen benutzt.

Aktiv Stereo heißt auch SG-Shutter. Für die Realisierung werden die Shutterbrille und Bildschirm mit 120Hz oder höher benötigt. Die Hersteller sind z.B. Nvidia, Sony, Samsung, Hisense usw..

Passiv Stereo heißt auch PR-Polaroid. Für die Realisierung sind die Farbfilterbrille oder Polfilterbrille und Bildschirm zu benötigen. Die Hersteller sind z.B. LG, Konka, Chuangwei[1], usw..

[1] Konka, Chuangwei sind die chinesischen Hersteller China

2.3 Stereoskopie mit 3D-Brille

Bisher werden drei verschiedene 3D Brillen auf dem Markt angeboten. Die sind Farbfilterbrillen (Anaglyphenbrillen), Shutterbrillen und Polfilterbrillen. Ob die alle 3D Brillen gleich sind? Welche Unterschiede, Vorteile und Nachteile gibt es denn? Die sind in den folgenden Abschnitten zu erklären.

2.3.1 Farbfilterbrillen (Anaglyphenbrillen)

In den Farbfilterbrillen wird passiv Stereo verwendet. Die Gläserfarben sind rot-blau, rot-grün, magenta-grün oder rot-cyan. Die Sorten der Farbfilterbrillen sind in der Abbildung 2.3 dargestellt.

Die Brillen sind billig und sehr populär. Man kann einfach die Filme oder Foto von Internet herunterladen und zu Hause Stereoskopie genießen. Weil bis jetzt alle Monitor, Fernsehen oder Beamer die Farbfilterbrillen unterstützen.

Das Prinzip der Farbfilterbrille ist auch einfach, dass die Bilder für zwei verschiedene Farben gerendert werden. Das linke bzw. rechte Auge sieht jeweils nur eine Farbe. Aber es gibt ‘etwas Ghosting‘. In der Abbildung 2.2 ist das Prinzip der Farbfilterbrille dargestellt.

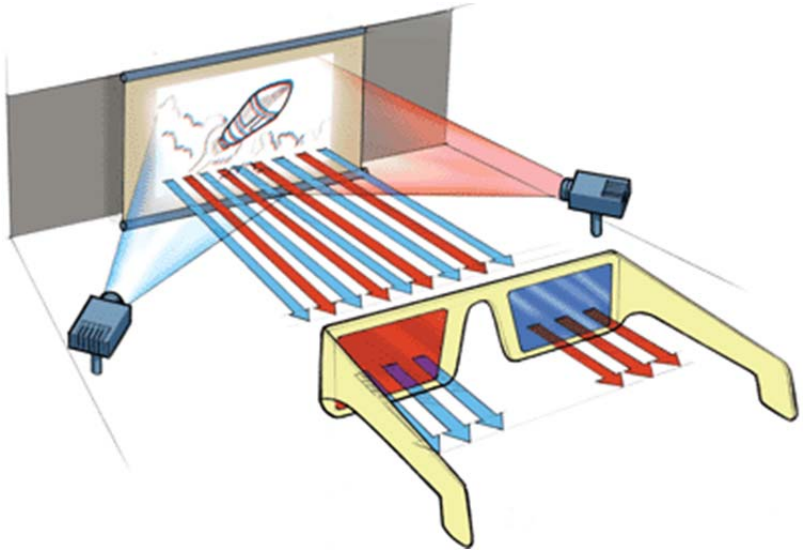


Abbildung 2. 2: Das Prinzip der Farbfilterbrille (eigene Darstellung)

Wenn man Filme oder Fotos mit den Farbfilterbrillen schauen möchte, muss man genau wissen, welche Farbfilterbrille zu benutzen ist. Weil einige Filme die rot-grün Farbfilterbrillen benötigen, oder andere Farbfilterbrillen. Natürlich sind ihre Nachteile sehr offenbar. Die Farbfilterbrillen können zwei verschiedene Farben filtern. Wenn man damit die Filme schaut, wird „colourcast“ erzeugt, davon wird Stereoskopie erzeugt.

Abbildung 2. 3: Die Sorten der Farbfilterbrillen (eigene Darstellung)

2.3.2 Polfilterbrillen

In den Polfilterbrillen wird auch passiv Stereo verwendet. Im Vergleich zu den Farbfilterbrillen sind die Gläserfarben von Polfilterbrillen gleich. Der Bildeffekte ist deutlich besser als Farbfilterbrillen. Bisher werden die Polfilterbrillen mit der sehr guten Helligkeit im Kino häufig verwendet.

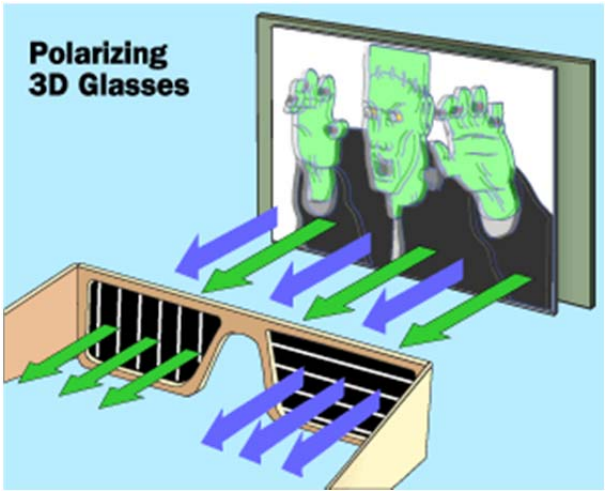


Abbildung 2. 4: Das Prinzip der Polfilterbrille (eigene Darstellung)

Der Lichtstrahl hat eine Polarisation. Die passive Stereo Technologie von Polfilterbrillen benutzt genau dieses Prinzip. Es werden zwei Bilder erzeugt, eines mit vertikaler eines mit horizontaler Polarisierung. Danach nehmen die Polfilterbrillen die zwei verschiedenen Bilder für zwei Augen an. Dadurch kann man die Stereoskopie erzeugen. Die Abbildung 2.4 zeigt das Prinzip.

Bisher sind Polfilterbrillen von RealD 3D in der Abbildung 2.5 dargestellt. Darunter ist MasterImage 3D sehr bekannt.

Abbildung 2. 5: Die Polfilterbrille von RealD [2]

2.3.3 Shutterbrillen

In den Shutterbrillen wird aktiv Stereo verwendet. Aktiv Stereo Technologie ist sehr bekannt und wird in Fernsehen, Monitor oder Beamer usw.. eingesetzt. Viele Hersteller produzieren mit aktiv Stereo Technologie Bildschirm, Computer Spiel oder Anwendungssoftware. Aber die Shutterbrillen sind teuer.

Die Realisierung von aktiv Stereo erfordert eine Bildwiederholfrequenz von 120Hz oder höher, d.h., für jedes Auge eine Bildfolge mit mindesten 60Hz.

Die Shutterbrille braucht zur Steuerung der Umschaltung in der Brille ein Signal. Diese wird durch einen Infrarot-Sender erzeugt. Wenn Bild-Signale gefangen werden, kann die Shutterbrille gleichzeitig eingeschaltet werden.

Aktiv Stereo Technologie kann die Bildschirmauflösung nicht verändern und die Helligkeit vom Bildschirm nicht verringern.

Genau wie für Polfilterbrillen ist ein spezifischer Bildschirm für Shutterbrillen nötig. Im Abschnitt 2.5 wird es genau erklärt.

2.3.4 Vergleich von drei verschiedenen 3D Brillen

In der Tabelle 2 stellt der Vergleich von drei verschiedenen Brillen zusammen.

Vergleich von drei verschiedenen Brillen		
Typ	Vorteile	Nachteile
Farbfilterbrillen	Einfach Prinzip	Schlechte 3D Wirkung
	billig	Farbabweichung
Polfilterbrillen	Bessere 3D Wirkung	niedrige Bildschirmauflösung (Halbierung)
	billig	geringere Helligkeit
Shutterbrillen	Mehre Anwendung	teuer
	Bildschirmauflösung	Bild flackernd
	Helligkeit	Ghosting
	3D Blu-ray	

Tabelle 1: Vergleich von drei verschiedenen 3D Brillen (eigene Darstellung)

2.4 3D Grakfikkarte

Stereoskopie benötigt nicht nur eine entsprechende 3D Brillen, sondern auch die passende Grafikkarte, die die 3D Technologie unterstützt.

Zur Zeit sind zwei hauptsächlichen Hersteller, die die Grafikkarte speziell produzieren, Nvidia und AMD(ATI). In den folgenden Abschnitten werden die Unterschiede

zwischen Nvidia und AMD vorgestellt.

2.4.1 Nvidia

Nvidia ist sehr bekannt und hat viele Erfahrungen für Grafiklösungen. Die Grafikprodukte GeForce und Quadro werden in den meisten Computern eingesetzt. Die beiden Grafikkarten unterstützen stereoskopische Visualisierung.

- 3D VISION

Die Technologie „3D VISION“ von Nvidia bietet eine Grafiklösung für stereoskopische Visualisierung an.

Für Stereoskopie hat Nvidia ein Produkt „3D Vision (3D Brille)“, das in der Abbildung 2.6 gezeigt wird, produziert.

Abbildung 2. 6: Die 3D Vision [3]

Die „3D Vision“ ist eine Shutterbrille und verwendet aktiv Stereo. Sie ist nicht billig. So ist Infrarot-Sender nötig. Das komplette Produkt heißt „Nvidia 3D Vision Kit“, die in der Abbildung 2.7 dargestellt ist. Aber die „Nvidia 3D Vision Kit“ enthaltet zwei verschiedene Versionen, GeForce 3D Vision und Quadro 3D Vision.

Shutterbrille benötigt noch einen speziellen Bildschirm mit mindestens 120Hz.

Abbildung 2. 7: Nvidia 3D Vision Kit [3]

- Nvidia GeForce

GeForce Grafikkarten können in allen PCs oder Laptops eingesetzt werden und ihr Preis ist geringer als der einer Quadro Grafikkarte.

Heute sind die Computer Produkte inkl. GeForce Grafikkarten und 3D Brillen sehr einfach zu kaufen, um 3D Filme, 3D Fotos oder 3D Coumputer Spiel zu genießen.

Aber nicht alle GeForce Grafikkarten die Technologie 3D Vision unterstützen. Vor dem Einkaufen müssen die technischen Daten der GeForce Grafikkarten auf der Webseite von Nividia angeschaut werden.

Zum Bsp. GeForce Grafikkarte GT550M. In der folgende Abbildung 2.8 ist die Technische Daten von GT550M, die die Funktion von „NIVIDIA 3D Vision“ beinhalt, dargestellt

Für GeForce Grakfikkarten gibt es keine spezifischen Anforderungen für CPU. Aber es wird eine CPU mit 2 Cores und 2/4 Threads empfohlen.

Feature Support:

NVIDIA 3D Vision Ready	✓
NVIDIA Optimus Technology	✓
Hardware Video Decode Acceleration	✓
NVIDIA Verde Drivers	✓
NVIDIA PhysX™-ready	✓
NVIDIA CUDA™ Technology	✓
Microsoft DirectX	11

Abbildung 2. 8: Die Technische Daten von GT550M [4]

- Nvidia Quadro

Die Quadro Grafikkarten können auch in der meisten PCs oder Laptops eingesetzen werden. Quadro Grafikkarte ist die professionelle Grafikkarte . Sie sind sehr teuer.

Quadro Grafikkarten mit sehr guten 3D Grafiklösungen sind besser als GeForce. Quadro Grafikkarten umfassen einige verschiedene Produkte, z.B. Quadro, Quadro Fermi, Quadro Plex, Quadro FX, Quadro CX und Quadro NVS.

Darunter unterstützen nur Quadro Fermi und Quadro FX die Technologie Quad Buffer.

Vierfach gepuffertes Stereo ist eine Technologie für die Realisierung des stereoskopischen Renderings. Für stereoskopisches Rendering muss jedes Auge ein einzelnes Bild erhalten. Vierfach gepuffertes Stereo bietet für jedes Auge die double buffering an.

Vierfach gepuffertes Stereo sendet an jedes Auge ein eigenes Bild aus einem geringfügig anderen Blickwinkel und verwendet dafür – im Unterschied zu den herkömmlichen zwei Puffern (vorne, hinten) – vier Puffer (vorne links, vorne rechts, hinten links, hinten rechts).

Nvidia Quadro Fermi und Quadro FX Grafikkarte unterstützen die Technologie vierfach gepuffertes Stereo und bietet zwei Weise, die in der Tabelle 3 dargestellt sind, um Grafikkarte in PC einzusetzen.

1.	über internen Stereoanschluss	gut
2.	über USB	Flackern

Tabelle 2: Zwei Varianten, um Grafikkarte in PC einzusetzen(eigene Darstellung)

Hier wird die 1. Variante empfohlen, weil die Performance deutlich besser ist. In der Tabelle 4 sind eine Bespiel der Unterschiede zwischen Quadro FX 5800 und Quadro FX 580 dargestellt. Darin wird deutlich gezeigt, dass die gesamte technische Performance von Quadro FX 5800 sehr höher als Quadro FX 580 sind.

	Quadro FX 5800	Quadro FX 580
	über internen Stereoanschluss	über USB
CUDA	240	32
Speichergröße	4GB	512MB
Speicherschnittstelle	512-bit	128-bit
Speicherbandbreite	102GB/s	25.6
OpenGL	3.1	3.0
Nvidia SLI	ja	nein
Leistungsaufnahme	189W	40W

Tabelle 3: Die Unterschiede zwischen Quadro FX 5800 und Quadro FX 580(eigene Darstellung)

In Deutschland sind zur Zeit nicht alle Quadro Grafikkarten verfügbar. In der Tabelle 5 werden die verfügbaren Quadro Grafikkarten für die Technologie Quad Buffer (Vierfach gepuffertes Stereo) in Deutschland angezeigt.

Quadro	6000
	5000
	4000
	FX 5800
	FX 4800
	FX 3800
	FX 5600
	FX 4600
	FX 4700 X2
	FX 3700

Tabelle 4: Die verfügbare Quadro Grafikkarten für Quad Buffer in Deutschland(eigene Darstellung)

Die Technologie von der Quadro FX und Quadro Fermi wird schnell entwickelt. Aber gibt es noch die wesentlichen Unterschiede in den beiden Produkten. In der Abbildung 2.9 sind die Quadro Fermi dargestellt.

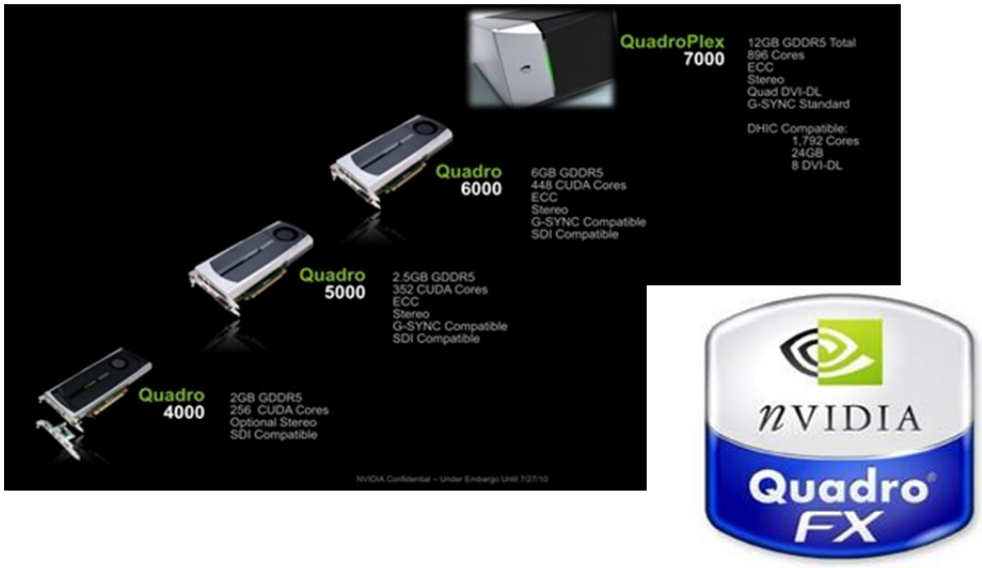


Abbildung 2. 9: Die Grafikkarte von Quadro Fermi [5]

In der folgenden Tabelle 6 wird der Vergleich von den Grafikkarten Quadro FX 3800, 4800, 5800 und Quadro 4000, 5000, 6000 angezeigt. Weil die 6 Grafikkarten die meiste Grafik-Technologie unterstützen.

.

GK-Typ	**Quadro 6000**	**Quadro 5000**	**Quadro 4000**	**Quadro FX 5800**	**Quadro FX 4800**	**Quadro FX 3800**
CUDA	448	352	256	240	192	192
Speichergröße	6 GB GDDR5	2.5 GB GDDR5	2 GB GDDR5	4 GB GDDR3	1.5 GB GDDR3	1 GB GDDR3
Speicherschnittstelle	384-bit	320-bit	256-bit	512-bit	384-bit	256-bit
Speicherbandbreite	144 GB/s	120 GB/s	89.6 GB/s	102 GB/s	76.8 GB/s	51.2 GB/s
Leistungsaufnahme	225W	152W	142W	189W	150W	108W
API	DirectX 11 OpenGL 4.1			DirectX 10 OpenGL 3.1		
Preis(Euro)	3500-4200	1800-2000	700-900	3200-3800	1600-1900	800-1000

Tabelle 5: Vergleich von 6 Quadro Grafikkarten(eigene Darstellung)

In der Tabelle 6 kann man einfach erkennen, dass Quadro 5000, Quadro FX 5800 und Quadro FX 4800 gute Auswahl sind. Nach dem Preis-Leistungsverhältnis ist Quadro FX 5800 in dieser Tabelle am besten. Danach sind Quadro 5000, Quadro FX 4800.
Quadro Grafikkarte hat hohe leistungsfähige professionelle 3D Grafiklösungen und benötigt hohe Anforderung für CPU, Speicher, Festplatte und Betriebssystem. Die Tabelle 7 zeigt das Minimum der Systemanforderung.

Betriebssystem	Vista 32/64, Win7 32/64
Speicher	mind. 2GB
Festplatte	mind. 100MB
CPU	>=Intel Core 2 Duo, >= AMD Athlon X2

Tabelle 6: Das Minimum der Systemanforderung(eigene Darstellung)

2.4.2 AMD

AMD als der Gegenspieler von Nvidia hat im Jahr 2010 die 3D Technologie „HD3D“ veröffentlicht, dass die meisten RADEON Grafikkarten die HD3D Technologie unterstützen.

- AMD HD3D und Radeon HD

Das Prinzip von AMD HD3D und Radeon HD ist ähnlich wie Nvidia. In der Abbildung 2.10 wird das Prinzip von AMD HD3D dargestellt. Eine Shutterbrille, ein Bildschirm mit mindestens 120Hz und ein Infrarot-Sender sind nötig.

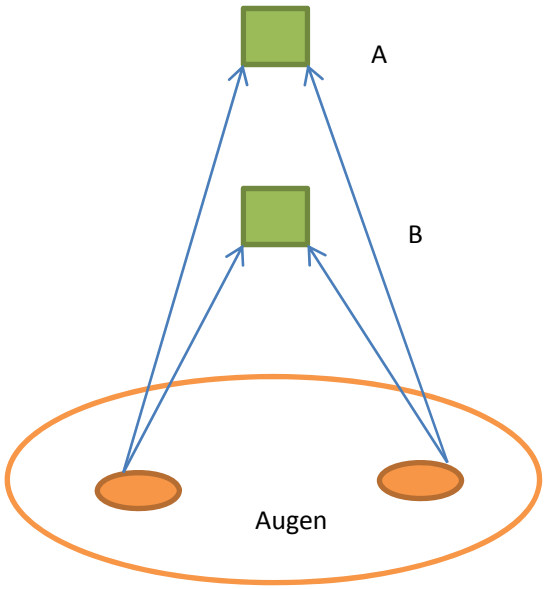


Abbildung 2. 10: Das Prinzip von AMD HD3D (eigene Darstellung)

AMD hat auch eine eigene Shutterbrille in der Abbildung 2.11 produziert.

Abbildung 2. 11: Die Shutterbrille von AMD [6]

Der Unterschied zwischen AMD HD3D und Nvidia 3D Vision ist die Benutzung von Brille. AMD HD3D unterstützt die 3D Brillen von anderen Herstellern. Aber unterstützt Nvidia 3D Vision nur die eigene Brille.

Für die Realisierung der AMD HD3D Technologie ist die Unterstützung der Software nötig. Die Firma IZ3D und DDD bieten die Softwarenunterstützung für AMD HD3D an.

In der Tabelle 8 zeigt eine Zusammenfassung für die AMD Radeon Grafikkarten, die die Technologie AMD HD3D unterstützen.

	3D Gaming	Blu-ray 3D	2D to 3D Video Conversion	3D Fotos
Radeon HD6990	**J**	**J**	**J**	**J**
Radeon HD6970	**J**	**J**	**J**	**J**
Radeon HD6950	**J**	**J**	**J**	**J**
Radeon HD6870	**J**	**J**	**J**	**J**
Radeon HD6850	**J**	**J**	**J**	**J**
Radeon HD6790	**J**	**J**	**J**	**J**
Radeon HD6770	**J**	**J**	**J**	**J**
Radeon HD6750	**J**	**J**	**J**	**J**
Radeon HD5970	**J**	**N**	**J**	**J**
Radeon HD5870	**J**	**N**	**J**	**J**

Radeon HD5850	**J**	**N**	**J**	**J**
Radeon HD5830	**J**	**N**	**J**	**J**
Radeon HD5770	**J**	**N**	**J**	**J**
Radeon HD5750	**J**	**N**	**J**	**J**
Radeon HD5670	**J**	**N**	**J**	**J**
Radeon HD5570	**J**	**N**	**J**	**J**
Radeon HD5550	**J**	**N**	**J**	**J**
Radeon HD5450	**J**	**N**	**J**	**J**

Tabelle 7: Die Radeon Grafikkarten, die die AMD HD3D unterstützen(eigene Darstellung)

Die Unterstützung für Blu-ray 3D benötigt die Technologie UVD 3.0. Ihre Technologie unterstützt bis jetzt nur AMD Radeon HD6000 Version.

- AMD FirePro und FireGL

Abbildung 2. 12: AMD FirePro und FireGL [6]

AMD Grafikkarten FirePro und FireGL unterstützen die OpenGL 4.0 und 3.3. Sie haben auch gute Kompatibilität zu CAD- und DCC-Anwendung. Die offizielle

Information wird gezeigt, dass AMD Grafikkarten FirePro und FireGL über den spezifischen Treiber auch das Quad Buffer Stereo (Vierfach gepuffertes Stereo) unterstützen können.
Aber die Technologie von AMD ist noch sehr jung und weiter zu entwickeln.

2.4.3 IZ3D

IZ3D ist eine Firma und bietet die Grafiklösung für die stereoskopische Visualisierung.
Bisher hat IZ3D die Produkte 3D Monitor und Software. 3D Monitor wird im Abschnitt 2.5 erklärt.
Die Software von IZ3D kann das normale 3D Spiel auf True 3D Spiel wechseln, damit man das Spiel mit der 3D Brille genießen kann. Die Software unterstützt alle Betriebssystemen und Grafikkarten. Die spezifische Grafikkarte ist nicht zu kaufen.
Die Technologie von IZ3D unterstützt auch 3D Filme und Fotos. IZ3D und DDD bieten auch die Grafiktreiber für AMD Grafikkarten mit AMD HD3D Technologie, weil AMD HD3D einen offenen Standard ist.

2.5 3D Monitor, 3D Fernsehen, 3D Beamer

Für stereoskopische Visualisierung sind ein 3D Monitor, 3D Fernsehen oder 3D Beamer nötig. In den oberen Abschnitten werden schon 3D Brillen und 3D Grafikkarte dargestellt. Man kann die Stereoskopie mit der Farbfilterbrille auf allen Bildschirme schauen. Aber wird mit Shutterbrille und Polfilterbrille ein genau 3D Bildschirm benötigt.

2.5.1 Für Polfilterbrille

In dem Kino trägt man eine Polfilterbrille oder Farbfilterbrille, um den Film anzuschauen. Zwei Beamer werden für Polfilterbrille eingestellt. Zwei Beamer erzeugen die Bilder für die Zuschauer. Das linke Auge erhält die Bilder von linkem Beamer und rechtes Auge erhält die Bilder vom rechten Beamer. Dann kann man die Bilder wie stereoskopische Visualisierung schauen. Diese Loesung ist für den Heimgebrauch zu aufwendig und zu teuer.
Deshalb werden andere Technologien verwendet .

- Zwei hintereinander angeordnete LCD Panels

In dem Bildschirm werden zwei „LCD Panel“ eingestellt. Auf den Panels werden zwei senkrecht zueinanderstehende Polarisationen verwendet (Bildwiederholfrequenz 120Hz). Durch die Polfilterbrille erreicht man wieder einen stereoskopische Ansicht
Bisher werden schon von einigen Herstellern 3D Bildschirme produziert, z.B. der 3D Monitor von IZ3D in der Abbildung 2.13 usw..

3D Monitor: IZ3D H220Z1

Abbildung 2. 13: 3D Monitor IZ3D H220Z1 [7]

Man muss über zwei Grafikkarten verbinden.

Abbildung 2. 14: 3D Monitor IZ3D H220Z1 [7]

Abbildung 2. 15: 3D Monitor IZ3D H220Z1 [7]

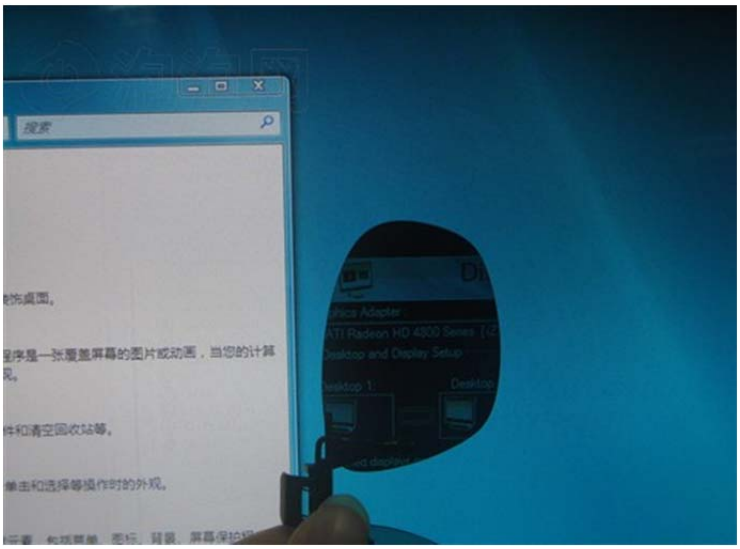


Abbildung 2. 16: 3D Monitor IZ3D H220Z1 [7]

Man kann die Bilder mit der Polfilterbrille auf hinterem LCD Panel sehen.

Aufgrund der Kosten wird der Monitor von den Herstellern sehr wenig produziert.

- interlaced

1080i ist eine alte Technologie für Bildschirmauflösung. Das Prinzip der 1080i Technologie ist interlaced. Die Abbildung 2.17 sind das Prinzip dargestellt.

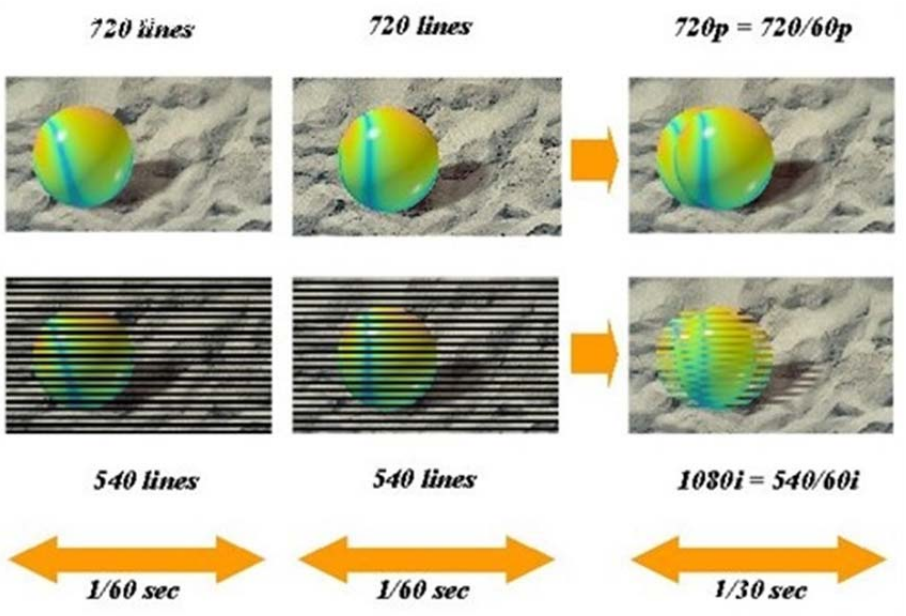


Abbildung 2. 17: Das Prinzip der 3D Technologie interlaced [12]

Die Bilder werden von ungeraden Zeile und geraden Zeile verteilen, Es gibt auch zwei senkrechte zueinandere Polarisationen auf den beiden Zielen. Wenn man die Polfilterbrille trägt, kann von linkem Auge die ungerade Zeile gesehen und von rechtem Auge die gerade Zeile gesehen werden.

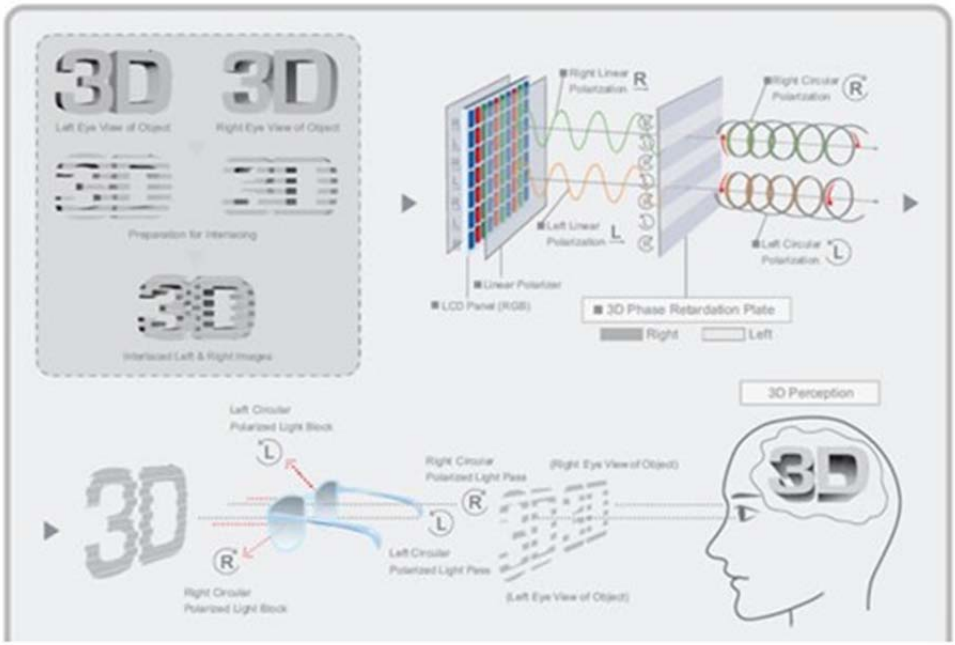


Abbildung 2. 18: Das Prinzip dieser 3D Technologie [12]

Aufgrund der Differenz der Verschiebung wird die Stereoskopie im Hirn erzeugt. Die Bildschirmauflösung wird reduziert.

Z.B. der 3D Monitor LG D2341. In seinem Bildschirm wird die wichtige Technologie eingesetzt. Wenn man mit der Polfilterbrille die Bilder oder den Film anschaut, gibt es keine besondere Anforderung für Grafikkarten.

In der Abbildung 2.19 sind der 3D Monitor LG D2341 ohne/mit Brille dargestellt.

Ohne 3D Brille

Mit 3D Brille

Abbildung 2. 19: Der 3D Monitor LG D2341 [9]

2.5.2 Für Shutterbrille

Der 3D Bildschirm für Shutterbrille ist in der Welt sehr bekannt. Bisher gibt's viele Produkte aus den verschiedenen Herstellern.

Das aktive Stereo benötigt die spezifische Grafikkarte und Shutterbrillen und einen spezifischen 3D Bildschirm.

Aufgrund der spezifischen Grafikkarte müssen die Hersteller die eigenen 3D Bildschirme mit der guten Kompatibilität produzieren und testen.

- AMD HD3D

In der Webseite von AMD hat schon die Liste für 3D Fernsehen und 3D Beamer empfohlen. In der Tabelle 9 zeigt einige 3D Monitore, die die Technologie AMD HD3D unterstützen. Die folgenden Abbildungen 2.20 und 2.21 zeigen die 3D Beamer und 3D Fernsehen.

a) 3D Monitor

Zalman line interleaved
Viewsonic V3D241WM – LED
Acer HS244HQ

Tabelle 8: Die 3D Monitoren, die AMD HD3D unterstützen(eigene Darstellung)

b) 3D Fernsehen

Mitsubishi	Samsung	LG	Sharp	Sony	Panasonic
WD-65738	LN46C750	55LX6500	LC60LE925UN	XBR-46HX909	TC-P50VT25
WD-82738	UN46C7000WF	47LX9500	LC52LE925UN	XBR60LX900	TC-P54VT25
WD-73738	UN46C9000	55LX9500		XBR52LX900	PS63C7705
WD-82838	UN55C9000			KDL-55HX800	PS50C7705
WD-65838	PN50C7000YF			KDL-46HX800	
WD-73838	PN63C8000YF				
	PN50C8000YF				
	PN58C8000YF				
	UN55C7000XF				
	UN40C7000WF				
	UN55C7000WF				
	PN58C680				
	PN50C680G5F				
	PN50C490B3D				
	LN55C750				

Abbildung 2. 20: Die 3D Fernsehen, die AMD HD3D unterstützen [13]

c) 3D Beamer

Viewsonic	Optoma	BenQ	Infocus	Mitisubishi	Dell	Sharp
PJD6531W	PRO350W	MP777	IN3116	EW270U	S300	PG-D45X3D
PDJ6251	HD67	MP776	IN2116	XD600U	M410HD	PG-D3010X
PDJ6241	HD66	MP626	DepthQ-WXGA-HD	XD280U	M210X	PG-D2500X
PDJ6221		MP782 ST	IN104	EX240U		
PDJ6381		MP772 ST	IN102	XD221U		
PDJ6211						
PDJ6220-3D						
PDJ6210-3D						
PDJ5111-3D						

Abbildung 2. 21: Die 3D Beamer, die AMD HD3D unterstützen [13]

AMD HD3D hat einen offenen Standard. So basieren ein paar Produkte von anderen Herstellern auf AMD HD3D. Sie produzieren auch die eigene Shutterbrillen und Monitor, Fernsehen oder Beamer, die auch gute Kompatibilität zu der 3D Grafikkarte von AMD haben und unterstützen. Die Abbildung 2.22 sind ein paar 3D Produkte von anderen Herstellern dargestellt.

Shutter Glass Models	Works with
XPand X102	Projectors with DLP Link support[2]
XPand X103	3D LCD TV, 3D DLP TV, 3D Plasma TV
Xpand X103x	HP Envy 17 3D Notebook
Viewsonic Shutter glasses	Included with Viewsonic V3D241wm-LED monitor[1]
LG 3D Glasses (AG-S100)	LG 3D TVs
Sharp AN3DG10S	Sharp 3D TVs
Sony 3D Glasses (TDGBR100)	Sony 3D TVs
Panasonic 3D Glasses (TY-EW3D2MU, TY-EW3D2SU)	Panasonic 3D TVs
Samsung 3D Glasses (SSG-2200AR/ZA)	Samsung 3D TVs
Acer Shutter Glasses	Included with Acer HS244HQ

Abbildung 2. 22: Die 3D Produkte von andere Herstellers [13]

- Nividia 3D VISION

Nvidia hat keinen offenen Standard, aber es gibt viele Produkte, die Ndivia 3D VISION unterstützen. In der Tabelle 10 zeigt die aktuellen 3D Monitore. Mehre Informationen zu 3D Fernseher oder 3D Beamer können in der Webseite von Nvidia gefunden werden.

Marke	Auflösung	Anschlüsse
Acer GD245HQ	1920x1080	VGA,DVI,HDMI
Acer GN245HQ	1920x1080	VGA,DVI,HDMI
Acer HN274H	1920x1080	VGA,DVI,HDMI(3x)
Acer HS244HQ	1920x1080	VGA,HDMI(2x)
Alienware OptX AW2310	1920x1080	DVI,HDMI
Asus VG236H 120HZ	1920x1080	DVI,HDMI
Asus VG236HE	1920x1080	DVI,HDMI

BenQ XL2410T	1920x1080	VGA,DVI,HDMI
Hannstar HS233H3B	1920x1080	VGA,DVI,HDMI
Lenovo L2363dwA	1920x1080	
LG W2363D	1920x1080	VGA,DVI-D,HDMI(2x)
LG W2363DB-PF	1920x1080	VGA,DVI-D,HDMI(2x)
Samsung® SyncMaster 2233RZ 120Hz	1680x1050	DVI
ViewSonic FuHzion VX2268wm	1680x1050	VGA,DVI-D

Tabelle 9: Die 3D Monitore, die Nvidia 3D Vision unterstützen(eigene Darstellung)

2.5.3 Vergleich von den 3D Bildschirmen

Bei 3D Bildschirmen können die verschiedenen Technologien verwendet werden. Tabelle 11 zeigt einen Vergleich zwischen aktiv Stereo und passiv Stereo für 3D Bildschirme.

	Aktiv Stereo	Passiv Stereo
Flackernd	Ja	Nein
3D Bildschirmauflösung	gut	normal
3D Brille	schwer, teuer	leicht, billig
„Ohnmacht“	Ja	Nein
Helligkeit	normal	gut
Frequenz	120Hz	60Hz/120Hz/240Hz

Tabelle 10: Vergleich zwischen aktiv- und passiv Stereo für 3D Bildschirme(eigene Darstellung)

In der Tabelle wird deutlich, dass jede Stereo-Technologie Vorteile und Nachteile hat. Der wesentliche Unterschied besteht aber im 3D Realisierungsweg: aktiv Stereo

mittels Brille und passiv Stereo durch den Bildschirm.

2.6 Stereoskopie ohne 3D Brille

Normalweise braucht man eine 3D Brille, um 3D Film zu betrachten. Es gibt aber auch die Möglichkeit, Stereoskopie ohne Brille zu realisieren (Abbildung 2.23).

Abbildung 2. 23: Die Wirkung der Stereoskopie ohne Brille [8]

Zu der Realisierung der Stereoskopie ohne Brille werden drei Technologien verwendet.

- Parallax Barrier

Das Prinzip ist wie passiv Stereo mit Polfilterbrille. Zu der Realisierung der Technologie werden ein LCD-Schalter, Polarisation-Coating und ein Makromolekül-Flüssigkristall benötigt. Es werden viele vertikale Streifen durch das Flüssigkristall und das Polarisation-Coating erzeugt. Davor wird eine Barriere erzeugt. Diese ist nicht durchsichtig. In der Abbildung 2.24 ist das Prinzip der Parallax Barrier dargestellt.

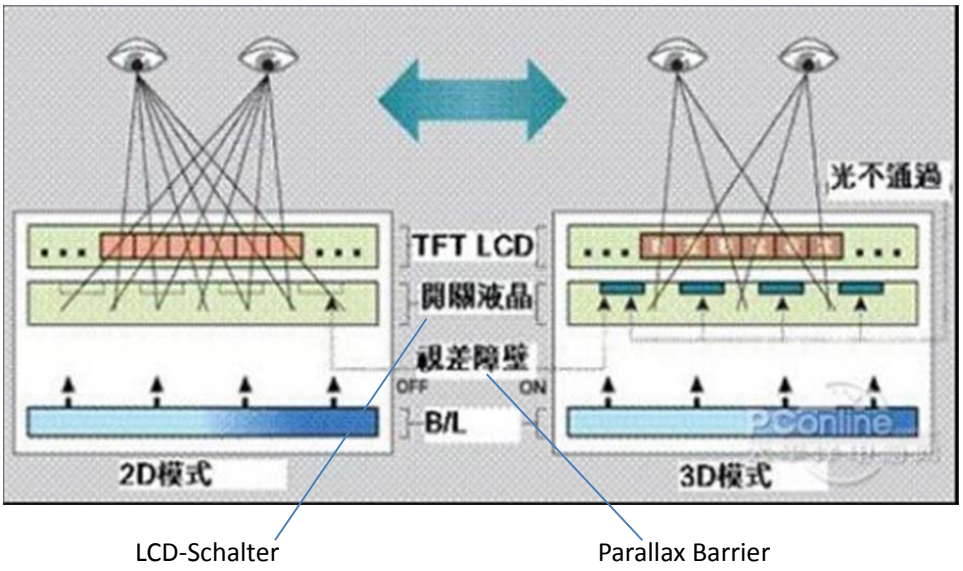


Abbildung 2. 24: Das Prinzip der Parallax Barrier [8]

Durch LCD Schalter sehen die zwei Augen verschiedene Bilder. Der Nachteil ist niedrige Helligkeit.

- Lenticular Lens

Das Prinzip der Technologie ist, dass ein Lenticular Lens vor dem LCD liegt. Die Pixel der Bilder werden auf viele Subpixel aufgeteilt und durch die Linsen können die Subpixel aus verschiedenen Richtungen gesehen werden. Die Helligkeit bleibt, weil die Lenticular Lens durchsichtig ist. In der Abbildung 2.25 ist das Prinzip der Lenticular Lens dargestellt.

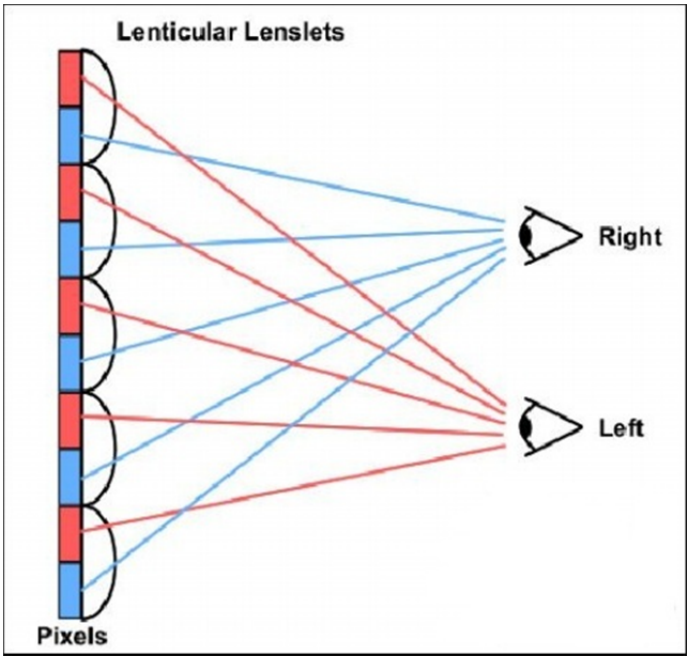


Abbildung 2. 25: Das Prinzip der Lenticular Lens [8]

- Directional Backlight

Diese Technologie kommt aus der Firma 3M. Es gibt zwei LED, die durch LCD-Panel und eigene Software den 3D Bilder mit der Sequenz(sequentiell) für Zuschauer zeigen. Aber ist die Technologie sehr jung. Sie wird sich weiter entwickeln.

Bisher haben einige Hersteller schon die Technologie Stereoskopie ohne Brille auf der Handy, Mp4 Player, Tablets verwendet. Die wichtigen Hersteller sind z.B. LG, HTC, Apple usw.. In den weiteren Abbildungen sind die Produkte dargestellt.

LG Handy

Abbildung 2. 26: LG Handy kann die Stereoskopie ohne Brille [14]

Ipad 3D

Abbildung 2. 27: Ipad mit Displayfolie kann die Stereoskopie ohne Brille [15]

Wenn eine Displayfolie auf Ipad- oder Iphone-Bildschirm aufgelegt wird, kann man die stereoskopischen Bilder ohne Brille schauen. Das Prinzip ist Lenticular Lens genannt.

Gadmei P83 PMP Tablet

Abbildung 2.28: Newsmy Mp4-Player und Gadmei Tablet realisieren die Stereoskopie ohne Brille [10] [11]

3 Stereoskopie mit JOGL

3.1 OpenGL Programmierung mit Java:JOGL

OpenGL (Open Graphics Library) ist eine Spezifikation für eine plattform- und programmiersprachenunabhängige Programmierschnittstelle zur Entwicklung von 2D- und 3D-Computergrafik. OpenGL ist eine professionelle Grafik-Programm-Schnittstelle.

JOGL (Java Bindings for OpenGL) ist ist ein Open-Source Projekt, welches 2003 von der Game Technology Group von Sun Microsystems ins Leben gerufen wurde. JOGL wurde mit dem Ziel entwickelt hardwarebeschleunigte 3D-Grafiken mit Hilfe einer low-level Grafikbibliothek in Java verwirklichen zu können. Da heutzutage OpenGL das Standard low-level 3D Application Programming Interface (API) auf dem Markt ist, hat man sich für ein Binding mit dieser Grafikbibliothek entschieden. Die Entwicklung wird von Sun Microsystems (Java) und Silicon Graphics Incorporated (OpenGL) unterstützt, wodurch die ständige Weiterentwicklung und Pflege des Projektes gesichert ist.

Mithilfe von JOGL kann ein Programmierer in Java auf die Funktionen von OpenGL zugreifen. Zu diesem Zweck besitzt JOGL spezielle Java-Wrapperklassen, welche als Schnittstellen zu den nativen Funktionen von OpenGL dienen.

JOGL stellt die meisten Features von OpenGL, der GLU und der GLUT Bibliotheken zur Verfügung. Funktionen der OpenGL Utility Library (GLU) sind z.B. die Unterstützung beim Rendern von Kugeln, Zylindern und anderen grafischen 3D-Objekten aber auch die Kamerapositionierung usw. JOGL implementiert aber nur einen Teil der Funktionen des OpenGL Utility Toolkit (GLUT) wie beispielsweise die Realisierung von grafischen Grundprimitiven. Die Fenstersystemfunktionen der GLUT wurden nicht mit übernommen, da Java sein eigenes high-level Fenstersystem mitbringt. Das heißt Java-Programmierer müssen sich nicht mit neuen Funktionen aus der GLUT befassen sondern kömnen Fenster mit den AWT- und Swing-Komponenten realisieren, was den Einstieg in JOGL wesentlich erleichtert.

3.2 Geschichte

- 07/1992 wurde SGI OpenGL 1.0 veröffentlicht.
- 1995: SGI OpenGL 1.1. Es gab mehre neue Funktionen.
- 1997 wurden viele 3D Spiel für Windows 95 veröffentlicht. In diese Zeit war DirectX 3.0 sehr schlecht, so unterstützte Windows 95 OpenGL.
- Am 28.07.2003: SGI OpenGL 1.5 veröffentlicht. In OpenGL 1.5 sind Buffer Object, Shadow usw. hinzugefügt geworden.
- 08/2004 wurde OpenGL 2.0 veröffentlicht. OpenGL 2.0 aus 3Dlabs unterstützt OpenGL Shading Language.
- 08/2008 hat Khronos OpenGL 3.0 mit GLSL 1.30 shader Language in Siggraph 2008 veröffentlicht.
- 03/2009 wurde OpenGL 3.1 mit GLSL 1.40 veröffentlicht. OpenGL 3.1 hat API vereinfacht. Damit wird die Effizienz der Software-Entwicklung erhöht.
- 08/2009 wurde OpenGL 3.2 veröffentlicht. Die Version erhöhte die Leistung und die Geschwindigkeit der Grafikbearbeitung und verbesserte die Qualität des Sehvermögens.
- Am 26.07.2010 wurden OpenGL 4.1 und OpenGL OpenGL Shading Language 4.10 veröffentlicht. Die Version enthält gute Kompatibilität zu OpenGL 3.2.

3.3 Merkmale

- Modellieren

Die Bibliothek von OpenGL bietet die Funktionen für Punkte, Line, Polygon, Kurven, Flächen und einige Dreidimensionale Objekte, z.B. Ball, Kegel, Polyeder, Teekanne usw. an.

- Transformation

Die Grafikbibliothek von OpenGL bietet die Funktionen für Transformation, z.B. Translation, Drehung, orthografischen Projektion und perspektivischen Projektion usw. an.

- Farbe

OpenGL hat zwei Farbe-Modi . Die sind RGBA und Color Index.

- Licht und Material

Light: Emitted Light, Ambient Light, Diffuse Light, Specular Light usw.

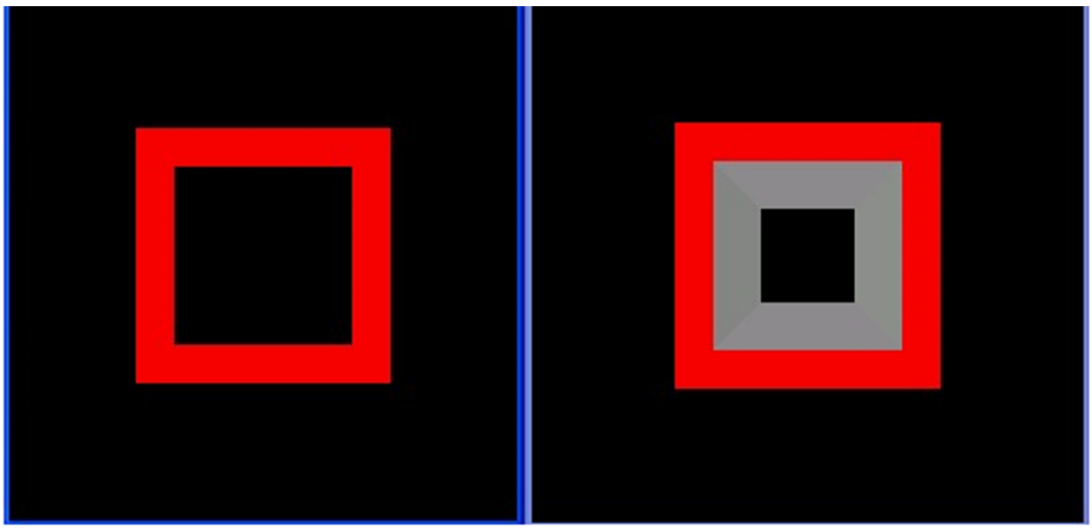


Abbildung 3. 1: Orthografische- und Perspektivische Projektion (eigene Darstellung)

Das Material wird als Reflexion des Lichtes zum Ausdruck. Z.B.:

- Texture Mapping
- Blending, Antialiasing, fog
- Double Buffering

Die Klassenhierarchie und Interfacehierarchie sind im Anhang.

3.4 Erweiterungen

Einige Bibliotheken bieten einige Funktionen für OpenGL an. Z.B.:

- GLU
- GLUT

GLUT(OpenGL Utility Toolkit) ist ein Programmpaket. Es ist unab- hängig von dem Fenster-System und teilt viele wichtige Klasse ein. Die mehren Informationen werden in dem folgenden Links gefunden: http://www.opengl.org/resources/libraries/glut

- GLUI
- GLEW
- GLEE

3.5 Wichtige Interface und Klassen

Interface	Beschreibung
GLAutoDrawable	Eine high-level Abstraktion, die einen eventbasierenden Mechanismus für OpenGL Rendering anbietet.
GLEventListener	Bietet Events an, welche im Code benutzt werden können um das OpenGL Rendering in GLAutoDrawable zu managen. OpenGLCallbackmethoden wie init(), display(), reshape() und displayChanged() sind im Interface zu implementieren.
GL	GL ist das Hauptinterface für OpenGL. Es bietet Zugang zur gesamten OpenGL-Funktionalität an.
GLDrawable	Eine Abstraktion für ein OpenGL Rendering

	Target. Kann einen OpenGL-Kontext erstellen, welcher benutzt werden kann um das Rendering auszuführen.
Animator	Ruft die display()-Methode einer oder mehrer GLAutoDrawable Insatzen in einer Schleife auf. Kreiert einen Hintergrund-Thread um diese Aufgabe zu erfüllen.
FPSAnimator	Direkte Unterklasse des Animator. Hier kann man eine Frames-Per-Second-Rate angeben und dadurch bestimmen wie oft die display()-Methode aufgerufen wird. Dabei wird das Erfüllen der Rate nicht garantiert aber angestrebt.
Component	Component ist eine abstrakte Klasse. Ein Component repräsentiert ein Objekt, welches auf dem Ausgabefeld angezeigt werden kann und mit dem Benutzer interagieren kann. Beispiele für Components sind Buttons, Checkboxes oder Scrollbars.
Canvas	Canvas repräsentiert einen leeren rechteckigen Bereich auf dem Display. In diesen Bereich kann eine Applikation zeichnen oder von dort Benutzereingaben entgegen nehmen. Für eine sinnvolle Verwendung von Canvas muss eine Unterklasse erstellt werden, in der die paint()-Methode überschrieben wird, um entsprechende Grafiken auf der Canvas auszugeben.
GLCanvas	GLCanvas ist eine AWT-Komponente, welche

	OpenGL Renderingunterstützung anbietet. GLCanvas ist die Hauptimplementation des GLDrawable-Interfaces.
Container	Ein Container ist eine AWT-Komponente die andere AWTKomponenten enthalten kann.
JPanel	JPanel ist ein generischer leichtgewichtiger Container.
GLJPanel	Eine leichtgewichtige Swing Komponente, welche OpenGL Renderingunterstützung anbietet. Kompatibel mit Swing User-Interfaces.
Panel	Panel stellt einen einfachen Container dar. In diesen kann eine Applikation alle anderen möglichen Komponenten einfügen.
GLU	über GLU hat man Zugriff auf die OpenGL Utility Library (GLU). Die Methoden entsprechen den Methoden der orginal C-Implementation.
GLUT	GLUT stellt eine Teilmenge des OpenGL Utility Toolkit dar. Nicht alle Methoden des GLUT wurden implementiert. Die Methoden weichen ein wenig von der orginal C-Implementation ab.

Tabelle 11: Wichtige Interface und Klassen von JOGL

3.6 Grundgerüst eines JOGL-Programm

JOGL besitzt die zwei Haupt-GUI-Klassen GLCanvas und GLJPanel. Diese implementieren das Interface GLAutoDrawable. Die beiden Klassen werden als

„Zeichenfläche“ für die OpenGL Kommandos benutzt.

GLCanvas ist der Canvas von AWT sehr änlich. Es ist eine sehr kompexe Komponente. Deswegen muss man aufpassen wenn man diese mit Swing Komponenten kombiniert. OpenGL Operationen werden durch die Hardwarebeschleunigung sehr schnell ausgeführt.

GLJPanel ist eine „light-Komponente“, welche ohne Probleme mit Swing Komponenten klar kommt. Im Gegensatz zu GLCanvas ist GLJPanel etwas langsamer in der Ausführung der OpenGL Operationen.

3.6.1 Verwendung von GLCanvas

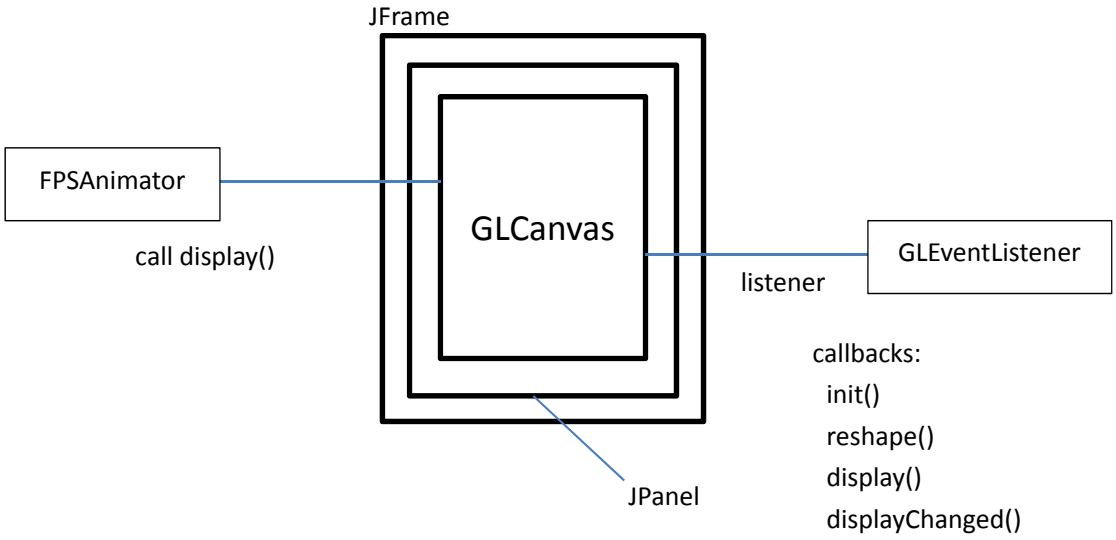


Abbildung 3. 2 Verwendung von GLCanvas (eigene Darstellung)

Die GLCanvas wird meist direkt ein JFrame eingefügt. Es besteht aber auch die Möglichkeit die GLCanvas in ein JPanel zu verpacken, dadurch ist es möglich mehrere lightweight GUI Komponenten in ein JFrame einzufügen. Die GLCanvas wird immer in Verbindung mit einem GLEventListener verwendet, welcher die Änderungen in der Canvas bearbeitet. Bei einem OpenGL-Programm übernimmt normalerweise OpenGL zur Laufzeit die Kontrolle über das Programm und ruft die

entsprechenden Callback-Methoden auf. Diese Funktion wird bei JOGL durch den GLEventListener übernommen. Wenn die Canvas nun das erste Mal erstellt wird, wird die init()-Methode des GLEventListeners aufgerufen. Die init()-Methode wird deshalb im eigenen Programm überschrieben um den OpenGL Status zu initialisieren. Jedes Mal wenn die Größe der Canvas oder deren Position verändert wird, wird die reshape()-Methode des Eventlisteners ausgeführt. Hier kann man den entsprechenden Code platzieren, um z.B. den Viewport zu initialisieren. Immer wenn die display()-Methode der Canvas aufgerufen wird, wird auch die Methode display() des Eventlisteners ausgeführt. In dieser Methode kann der Code für das Rendern der 3D Szene, die in der Canvas dargestellt werden soll platziert werden. Der Eventlistener enthält auch noch die Methode displayChanged(). Diese wird ausgeführt wenn sich beispielsweise die Einstellungen des Monitors ändern oder die Canvas auf einen anderen Monitor verschoben wird. Für einige grafische Anwendungen wie z.B. Animationen oder Spiele ist es nötig die Canvas regelmäßig zu aktualisieren. Mit einem FPSAnimator kann der Benutzer eine feste Frequenz einstellen mit der die display()-Methode der Canvas, und somit auch die des Eventlisteners, aufgerufen werden soll. Nach diesem Grundgerüst sind die meisten einfachen JOGL-Programme aufgebaut.

3.6.2 Verwendung von GLJpanel

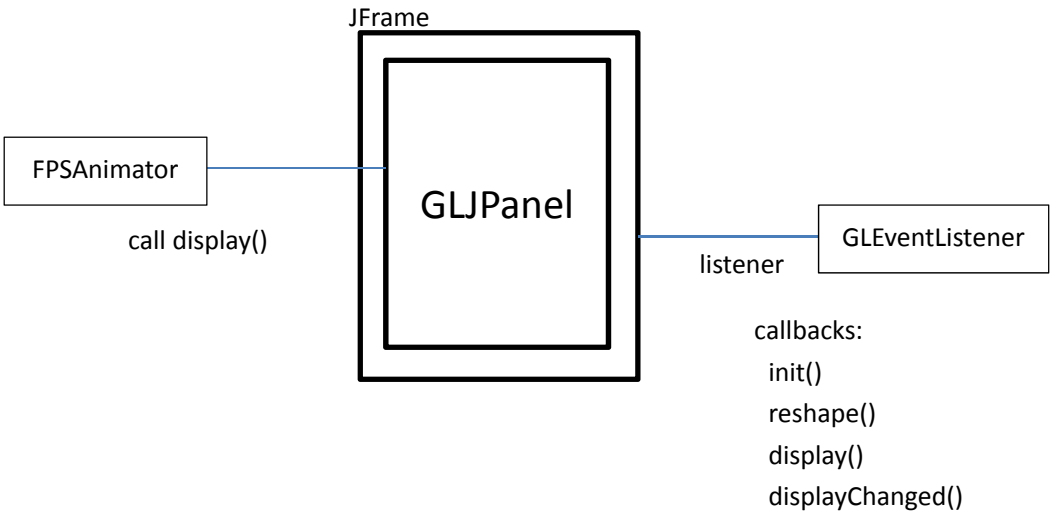


Abbildung 3. 3 Verwendung von GLJPanel (eigene Darstellung)

Da das GLJPanel eine „light-Swing Komponente“ ist kann sie direkt in das JFrame eingebettet werden, und es können auch noch weitere lightweight Komponenten hinzu gefügt werden. Der Rest dieses Aufbaus verhält sich ebenso wie der Aufbau mit einer GLCanvas. Das heißt man kombiniert das GLJPanel ebenfalls mit einem GLEventListener, welcher die Änderungen auf der Zeichenfläche bearbeitet. Mit einem FPSAnimator kann man wieder eine feste Rate einstellen mit der das GLJPanel aktualisiert werden soll.

3.7 Einstellung für Arbeitsumgebung

Zum Installieren und Ausführen von JOGL werden Java Development Kit, eine Entwicklungssoftware und JOGL benötigt.

3.7.1 Java und JOGL

a) Installieren von JDK

Hier wird die aktuelle Version empfohlen. Die aktuelle Version ist JDK 1.6.0 26.

b) JOGL 1.1 und JOGL 2.0

JOGL umfasst zwei Versionen, JOGL 1.1 und JOGL 2.0. JOGL 2.0 ist die aktuelle Version.

JOGL 1.1 enthält zwei jar-Dateien, gluegen-rt.jar und jogl.jar.

JOGL 2.0 ist ein etwas kompliziert und enthält viele jar-Dateien und dll-Dateien. Wenn ein JOGL Programm auszuführen ist, sind vier wichtigen jar-Dateien nötig, z.B. jogl.all.jar; gluegen-rt.jar; nativewindows.all.jar; new.all.jar, und noch die dll-Dateien.

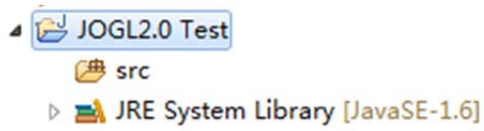
JOGL 2.0 unterstützt mehrere OpenGL-Profiles.

3.7.2 Entwicklungssoftware: Eclipse

c) Einstellung für Eclipse

Hier wird ein Bsp. mit JOGL 2.0 dargestellt. Die Einstellung von JOGL 1.1 ist gleich wie die Einstellung von JOGL 2.0.

- File--->New--->Java Project



- Bulid Path--->Configure Build Path--->Add External JARs

  Die dll-Dateien müssen noch in Projekt Ordner kopiert werden.

  Achtung: es gibt unterschiedliche dll-Dateien (32 Bit bzw. 64 Bit - Systeme). Die Abbildung 3.4 zeigt die jar- und dll-Dateien für OpenGL Projekt.

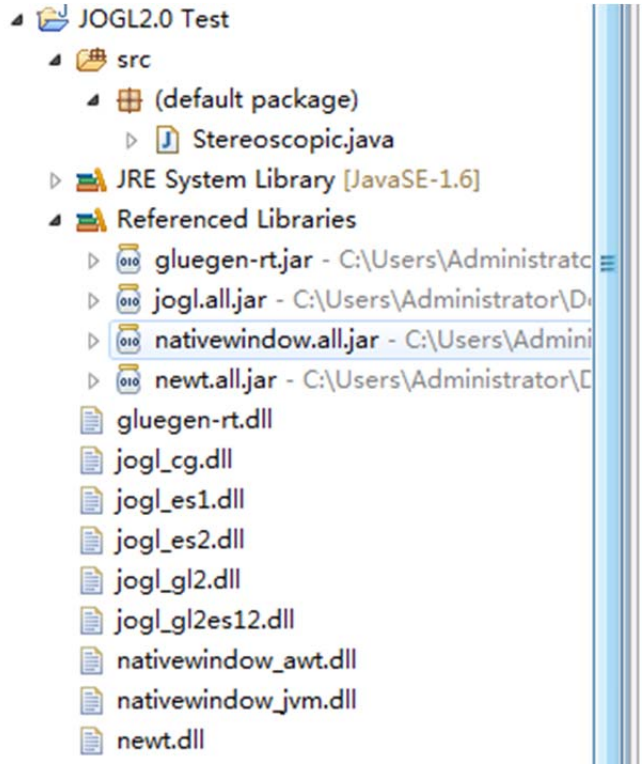


Abbildung 3. 4: Die jar- und dll-Dateien für OpenGL Projekt (eigene Darstellung)

3.7.3 Entwicklungssoftware:Netbeans

d) Einstellung für Netbeans

Das Software Netbeans ist ein etwas kompliziert. Die aktuelle Version ist 7.0. Sie unterstützt JOGL leider nicht.

- JOGL 1.1 unterstützt Netbeans von 6.5 bis 6.8.
- JOGL 2.0 unterstützt Netbeans 6.8.
- Es wird vorgeschlagen Netbeans 6.8 zu benutzen.

Die beide JOGL Versionen können in der Webseite von Netbeans gefunden.

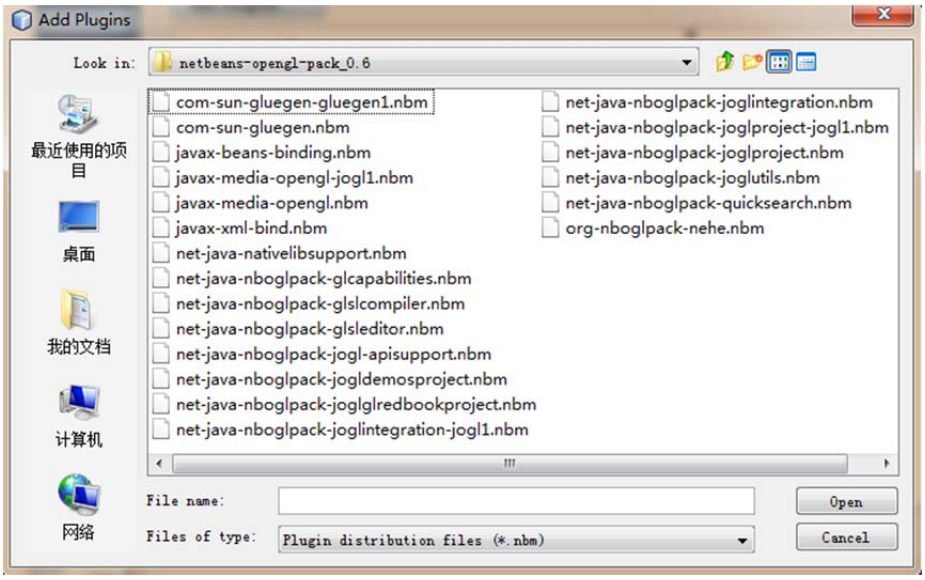
- JOGL 1.1: netbeans-opengl-pack_0.5.5
- JOGL 2.0: netbeans-opengl-pack_0.6

Hier wird ein Bsp. mit JOGL 2.0 und Netbeans 6.8 angezeigt. Die Einstellung von

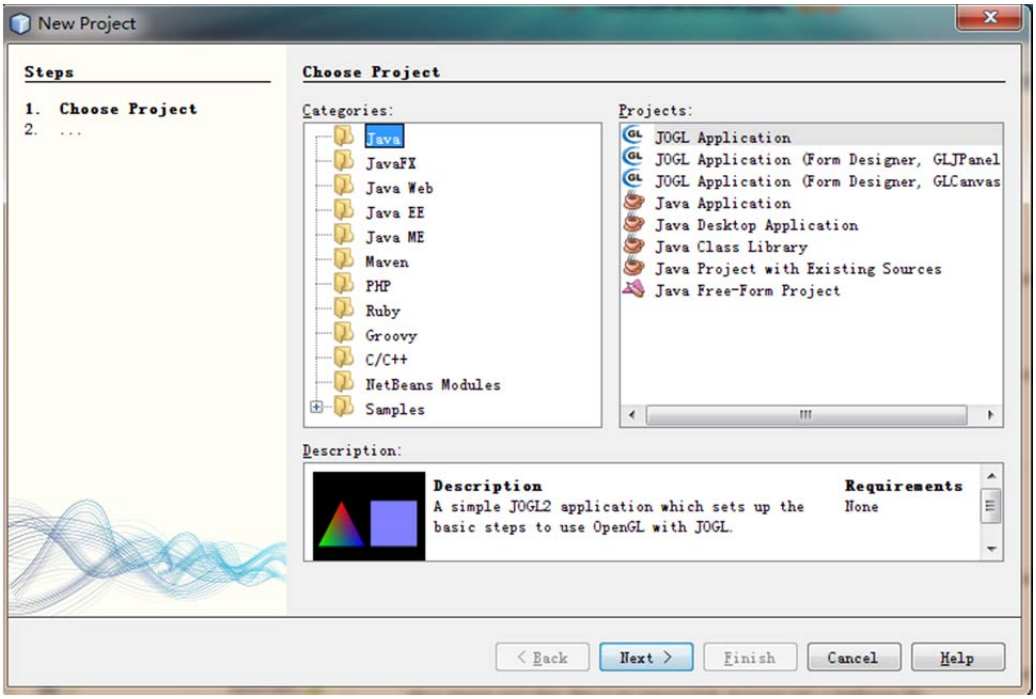
JOGL 1.1 ist so gleich wie JOGL 2.0.

- Tools--->Plugins--->Downloaded--->Add Plugins...

Alle Daten sollen hinzugefügt werden. Dann ist Netbeans automatisch zu installieren.

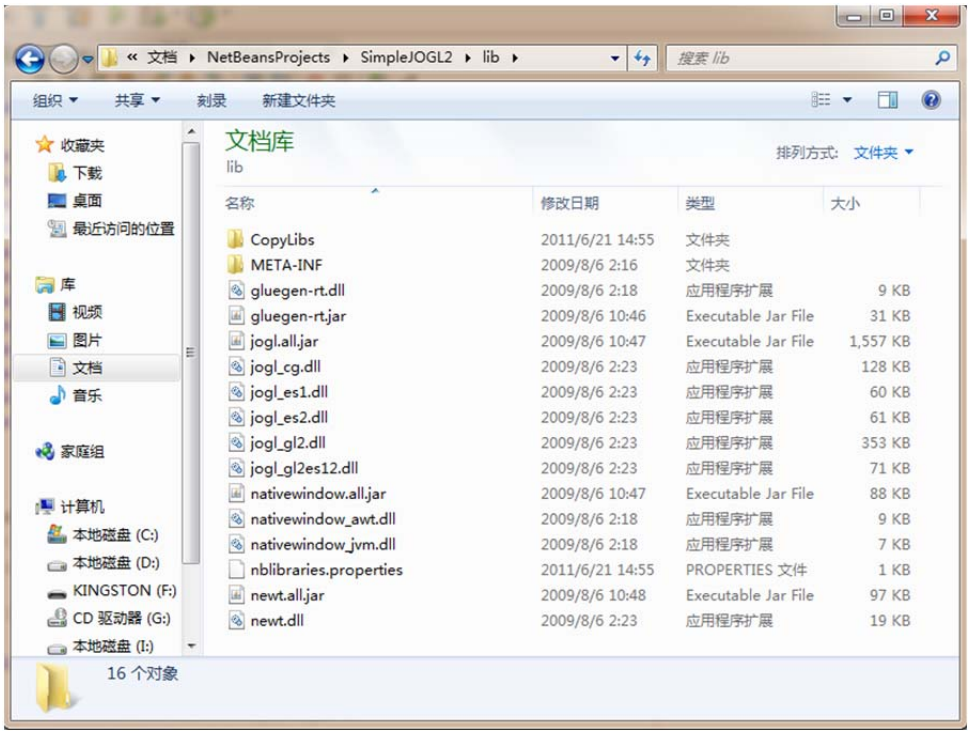


- File--->New Project



Nach der Erstellung des Projekts erstellt Netbeans auch automatisch die jar-Dateien

und dll-Dateien in dem Projekt-Ordner.



3.7.4 Vergleich zwischen Eclipse und Netbeans

In der Tabelle 13 zeigt der Vergleich zwischen Eclipse und Netbeans.

	Vorteile	Nachteile
Eclipse	Gute Kompatibilität	Manuell importieren
Netbeans	Automatisch importieren Visualisierung Jframe Exemplar erstellen Internationalisierung	Schlechte Kompatibilität

Tabelle 12: Vergleich zwischen Eclipse und Netbeans(eigene Darstellung)

3.8 Aktiv Stereo mit JOGL

JOGL hat die gleiche Konzeption wie OpenGL, Rendering auf 2D und 3D. GLCanvas

und GLJpanel sind die GUI-Komponenten. GLCanvas hat gute Kompatibilität zu AWT. GLJPanel hat gute Kompatibilität zu Swing.

Das aktive Rendering-Framework verwendet JOGL-Eigenschaften, um Anwendung und Kontext auf die Zeichnungsoberfläche zuzugreifen. Die Oberfläche ist eine typische Unterklasse von AWT's Canvas und bedient über „rendering thread", wie aus Abbildung 3.5 ersichtlich ist.

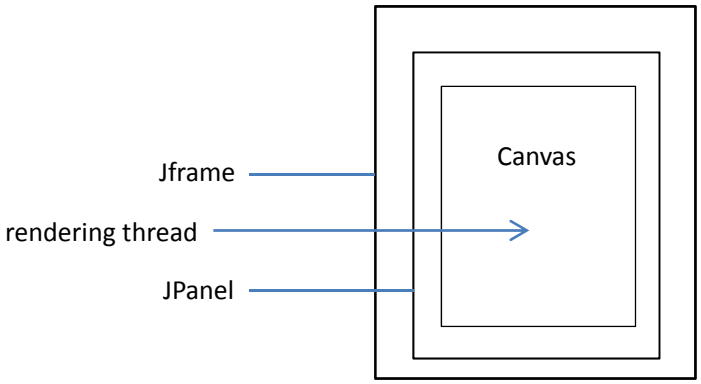


Abbildung 3. 5: Aktiv Stereo mit JOGL (eigene Darstellung)

JOGL kann die stereoskopische Visualisierung bzw. aktiv Stereo realisieren. Die Technologie von JOGL wird Quad Buffer Stereo (Vierfach gepuffertes Stereo) genannt.

3.8.1 Quad Buffer

Quad Buffer (Vierfach gepuffertes Stereo) ist eine Technologie für die Realisierung der stereoskopischen Rendering. Dazu muss jedes Auge ein einzelnes Bild erhalten. Vierfach gepuffertes Stereo bietet pro Auge ein double buffering an.

Vierfach gepuffertes Stereo sendet an jedes Auge ein eigenes Bild aus einem geringfügig anderen Blickwinkel und verwendet dafür – im Unterschied zu den herkömmlichen zwei Puffern (vorne, hinten) – vier Puffer (vorne links, vorne rechts, hinten links, hinten rechts).

Zu der Realisierung werden eine Shutterbrille und eine Nvidia Quadro Grafikkarte

benötigt. Die Information sind im Abschnitt 2.3.3 und 2.4.1 formuliert geworden. Die Grundlage von OpenGL wird in den folgenden Abschnitt dargestellt.

3.8.2 Grundlage

- Wichtige Interface:
  - GLDrawable
  - GLCanvas
  - GLJPanel
  - GLCapabilities
  - GLDrawableFactory

- Exemplar
  - Wenn ein JOGL Projekt in Netbeans erstellt wird, erstellt Netbeans ein Exemplar automatisch. Das spart Zeit
  - Natürlich muss man kein Exemplar benutzen. Man kann es auch löschen.
  - Exemplar mit UML Diagramm in der Abbildung 3.6 ist gezeigt:

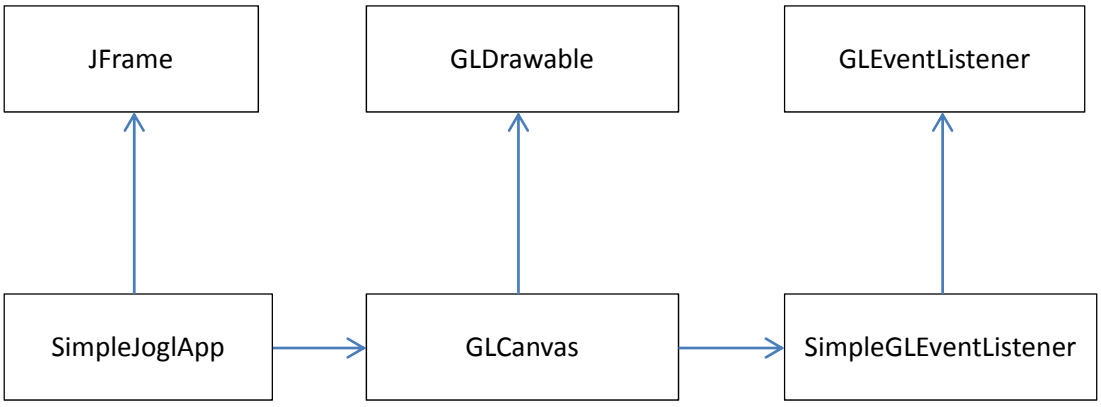


Abbildung 3. 6: Exemplar einer JOGL-Anwendung mit UML Diagramm (eigene Darstellung)

- SimpleJoglApp ist ein Jframe. Es enthält GLDrawable bzw. ein GLCanvas. SimpleGLEventListener realisiert GLEventListener von GLCanvas. GLDrawable kann automatisch ausgeführt werden, ist die Voraussetzung, dass GLEventListener optimal sein muss. In der Abbildung 3.7 ist es ersichtlich gezeigt.

```
Frame frame = new Frame("Simple JOGL Application");

// use GL2 profile since we only use the old OpenGL 2.x fixed function pipeline
GLCapabilities capabilities = new GLCapabilities(GLProfile.get(GLProfile.GL2));

// try to enable 2x anti aliasing - should be supported on most hardware
//   capabilities.setNumSamples(2);
//   capabilities.setSampleBuffers(true);

GLCanvas canvas = new GLCanvas(capabilities);

canvas.addGLEventListener(new SimpleJOGL1());
frame.add(canvas);
```

Abbildung 3. 7: Exemplar einer JOGL-Anwendung mit Code (eigene Darstellung)

- Die Methode init() wird beim Start der Anwendung aufgerufen, wenn der OpenGL-Kontext initialisiert wird. Hier sollten alle Anweisungen stehen, die zur Einrichtung der Zeichenumgebung dienen, z.B. Die Festlegung des Koordinatensystems. In der Abbildung 3.8 ist ersichtlich gezeigt.[18]

```
public void init(GLAutoDrawable drawable) {

    // Use debug pipeline, all OpenGL error codes will be automatically
    // converted to GLExceptions as soon as they appear
    drawable.setGL(new DebugGL2(drawable.getGL().getGL2()));

    GL2 gl = drawable.getGL().getGL2();
    System.out.println("INIT GL2 IS: " + gl.getClass().getName());

    // Enable VSync - this limits the rendering FPS to screen refresh rate
    gl.setSwapInterval(1);

    // Setup the drawing area and shading mode
    gl.glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
    gl.glShadeModel(GL2.GL_SMOOTH); // try setting this to GL_FLAT and see what happens.
}
```

Abbildung 3. 8: Die Methode init() (eigene Darstellung)

- Die Methode reshape() wird eingesetzt, um auf eine Größenänderung der Zeichenfläche zu reagieren. Sie wird auch beim Programmstart nach der init() – Methode aufgerufen. Der Aufruf von reshape() zieht einen Aufruf von display() nach sich, wie aus Abbildung 3.9 ersichtlich ist.[18]

```
public void reshape(GLAutoDrawable drawable, int x, int y, int width, int height) {

    GL2 gl = drawable.getGL().getGL2();
    GLU glu = new GLU();

    // avoid a divide by zero error!
    if (height <= 0)
        height = 1;

    final float h = (float) width / (float) height;
    gl.glViewport(0, 0, width, height);
    gl.glMatrixMode(GL2.GL_PROJECTION);
    gl.glLoadIdentity();

    glu.gluPerspective(45.0f, h, 1.0, 20.0);
    gl.glMatrixMode(GL2.GL_MODELVIEW);
    gl.glLoadIdentity();
}
```

Abbildung 3. 9: Die Methode reshape() (eigene Darstellung)

- Die Methode Display() wird bei jedem Zeichenvorgang aufgerufen, vergleichbar mit der paint() –Methode bei Komponenten. Außerdem erfolgt der Aufruf von display() für jeden Frame bei animierten Anwendungen. In dieser Methode müssen daher die Anweisungen zum Aufbau des Bildes stehen, wie aus

Abbildung 3.10 ersichtlich ist.[18]

- Die Methode dispose() wird beim Beenden des Programmes aufgerufen. Hier sollten alle „Aufräumarbeiten“ durchgeführt werden, z.B. die Freigabe von Textur-Speichern.[18]
- GLCapabilities ist eine Hilfsklasse, welche die Rendering-Fähigkeiten ein Rendering Context unterstützt.

```
public void display(GLAutoDrawable drawable) {

    GL2 gl = drawable.getGL().getGL2();

    // Clear the drawing area
    gl.glClear(GL2.GL_COLOR_BUFFER_BIT | GL2.GL_DEPTH_BUFFER_BIT);
    // Reset the current matrix to the "identity"
    gl.glLoadIdentity();

    // Move the "drawing cursor" around
    gl.glTranslatef(-1.5f, 0.0f, -6.0f);

    // Drawing Using Triangles
    gl.glBegin(GL2.GL_TRIANGLES);
        gl.glColor3f(1.0f, 0.0f, 0.0f);     // Set the current drawing color to red
        gl.glVertex3f(0.0f, 1.0f, 0.0f);    // Top
        gl.glColor3f(0.0f, 1.0f, 0.0f);     // Set the current drawing color to green
        gl.glVertex3f(-1.0f, -1.0f, 0.0f); // Bottom Left
        gl.glColor3f(0.0f, 0.0f, 1.0f);     // Set the current drawing color to blue
        gl.glVertex3f(1.0f, -1.0f, 0.0f);  // Bottom Right
    // Finished Drawing The Triangle
    gl.glEnd();

    // Move the "drawing cursor" to another position
    gl.glTranslatef(3.0f, 0.0f, 0.0f);
    // Draw A Quad
    gl.glBegin(GL2.GL_QUADS);
        gl.glColor3f(0.5f, 0.5f, 1.0f);     // Set the current drawing color to light blue
        gl.glVertex3f(-1.0f, 1.0f, 0.0f);  // Top Left
        gl.glVertex3f(1.0f, 1.0f, 0.0f);   // Top Right
        gl.glVertex3f(1.0f, -1.0f, 0.0f);  // Bottom Right
        gl.glVertex3f(-1.0f, -1.0f, 0.0f); // Bottom Left
    // Done Drawing The Quad
    gl.glEnd();

}
```

Abbildung 3. 10: Die Methode display() (eigene Darstellung)

- GLProfile
  - JOGL 2.0 stellt eine Abstraktion des ursprünglichen OpenGL Version, die Profile heißt. Profile ermöglichen Java-Anwendungen in einer Weise. Mit der Kompatibilität können sie mit mehreren OpenGL-Versionen gleichzeitig geschrieben werden.

- Die aktuelle Profile sind in der Abbildung 3.11 dargestellt:

```
public static final String GL4   = "GL4";
public static final String GL3   = "GL3";
public static final String GL2   = "GL2";
public static final String GLES1 = "GLES1";
public static final String GLES2 = "GLES2";
public static final String GL2ES1 = "GL2ES1";
public static final String GL2ES1 = "GL2ES2";
public static final String GL2GL3 = "GL2GL3";
```

Abbildung 3. 11: Die aktuelle Profile von OpenGL

- GL2 Profile unterstützt OpenGL nur bis 3.0
- Hier wird ein Bsp. in der Abbildung 3.12 angezeigt, wie man OpenGL mit JOGL 2 einrichten kann.

```
Context:
GLProfile profile = GLProfile.get(GLProfile.GL3);
...
GLCapabilities capabilities = new GLCapabilities(profile);
...
GLCanvas canvas = new GLCanvas(capabilities);
...
canvas.addGLEventListener(...);
...

Rendering:
public void display(GLAutoDrawable drawable) {
GL3 gl = drawable.getGL().getGL3();
gl.glClear(GL.GL_COLOR_BUFFER_BIT | GL.GL_DEPTH_BUFFER_BIT);
...
}
```

Abbildung 3. 12: OpenGL mit JOGL 2 einrichten (eigene Darstellung)

- GLCanvas ist ein komplexe AWT-Komponente. Sie bietet OpenGL-Rendering-Unterstützung, welche das Interface GLDrawable einbindet.

- void setStereo(bool enable)

- TRUE: Falls Stereo buffering freigegeben ist.
- FALSE: Falls Stereo buffering nicht freigegeben ist, ist das vorgegeben.

```
GLCapabilities capa = new GLCapabilities();
capa.setStereo(true);
```

- Quad Buffer für zwei Augen
  - linkes Auge

```
gl.glDrawBuffer(gl.GL_BACK_LEFT);
gl.glClear(gl.GL_COLOR_BUFFER_BIT | gl.GL_DEPTH_BUFFER_BIT);
```

  - rechtes Auge

```
gl.glDrawBuffer(gl.GL_BACK_RIGHT);
gl.glClear(gl.GL_COLOR_BUFFER_BIT | gl.GL_DEPTH_BUFFER_BIT);
```

  - glDrawBuffer: Puffer schreiben.
  - glClear löscht das Puffer und hält den zuvor definierten Konfigurationswert.
    - GL_COLOR_BUFFER_BIT: Farbe Puffer einrichten
    - GL_DEPTH_BUFFER_BIT: Tiefe Puffer einrichten
- glTranslated, glRotatef, glScalef
  - OpenGL benutzt homogene Koordinaten d.h. (x,y,z) = (x‘,y‘,z‘,h), x = x’/h, y = y‘/h, z = z‘/h, h = 1. Bsp. (2, 3, 4) = (2, 3, 4, 1).
  - OpenGL benutzt lineare Transformation und bezeichnet mit der Hilfe der Matrix-Multiplikation. Z.B. ein Punkt (3, 3, 3) verschiebt zu neuer Koordinaten (5, 6, 7), d.h. (3+2, 3+3, 4+4) = (5, 6, 7) bzw. glTranslated (2, 3, 4).
  - glTranslated: Die Multipliakation zwischen aktuelle Matrix und Translation-Matrix.
  - glRotatef kann auch mit Matrix bezeichnet werden, d.h. glRotatef (angle,x,y,z).

- glRotatef: Die Multipliakation zwischen aktuelle Matrix und Rotation-Matrix
- Automatisches Drehen mit x-, y- und z-Achse.
  - _gl.glRotatef (angle, 1.0f, 0.0f, 0.0f); mit x Achse drehen
  - _gl.glRotatef (angle, 0.0f, 1.0f, 0.0f); mit y Achse drehen
  - _gl.glRotatef (angle, 0.0f, 0.0f, 1.0f); mit z Achse drehen
  - Die Drehensrichtung und -geschwindigkeit werden durch angle +(-)= 1.0f definiert werden;
  - += und -= bedeuten die Drehensrichtung. 1.0f bedeutet die Drehensgeschwindigkeit.
- glScalef (float x, float y, float z): Die Multipliakation zwischen aktuelle Matrix und Scaling-Matrix.
- OpenGL hat eine Transformationsmatrix Stack

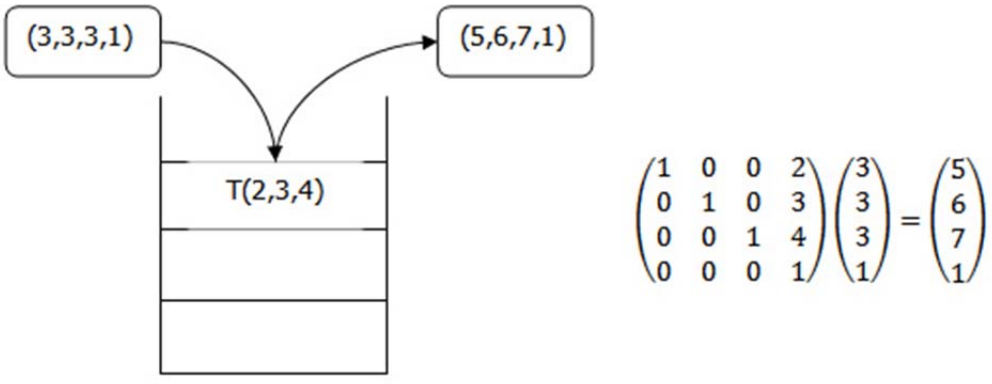


Abbildung 3. 13: Transformationsmatrix Stack (eigene Darstellung)

- glPushMatrix: es kopiert die Transformationsmatrix und Push nach Stack.
- glTranslated: T (2, 3, 4, 1) * (3, 3, 3, 1)=(5, 6, 7, 1).
- glPopMatrix: Pop die Transformationsmatrix aus dem Stack.

- glMatrixMode, glLoadIdentity

- glMatrixMode (Glenum mode)
- Mode bestimmt eine aktuelle Matrix. Die enthält GL_MODELVIEW, GL_PROJECTION, GL_TEXTURE.
  - GL_MODELVIEW: Anzeige Matrix
  - GL_PROJECTION: Projektion Matrix
  - GL_TEXTURE: Texturen Matrix
- glLoadIdentity und MatrixMode werden zusammenbenutzt.
- glLoadIdentity wird die aktuelle Matrix auf Einheitsmatrix wechselt.

- Animator

Die Klasse Animator legt einen neuen Thread an, der die repaint()-Methode eines GLAutoDrawable-Objektes (z.B. GLJPanel) regelmäßig aufruft. Auf diese Art und Weise lässt sich eine einfache Animation realisieren. [18]

Der Animator versucht, so oft wie möglich ein neues Bild zeichnen zu lassen. Die Bildgenerierungsrate ist daher stark von der Komplexität der Szene und der verfügbaren Hardware abhängig! [18]

- Idealerweise sollte der Animator als statische Variable deklariert werden, z.B.:

```
public static Animator animator;
```

- Starten der Animation:

```
animator.start();
```

- Beenden der Animation:

```
public void windowClosing(WindowEvent e) {
    animator.stop();
```

- FPSAnimator

FPSAnimator ist eine Unterklasse von Animator. Sie realisiert eine gewünschte

FPS(frames-per-second), um CPU-Zeit zu sparen.

```
final FPSAnimator animator = new FPSAnimator(canvas, 60);

final FPSAnimator animator = new FPSAnimator(canvas, 60, true);
```

- FPS ist gleich 60.

- Orthografische Projektion und Perspektivische Projektion sind zwei wichtige Funktionen von OpenGL.

  Die logischen Breiten der Orthografischen Projektion für vorne, hinten, oben, unten, links und rechts sind gleich. Deswegen wird die Orthografische Projektion auf CAD-Anwendung, Architektur, Text oder 2D Spiel angewendet.

  Perspektivische Projektion kann die weite Ansicht verkürzen und schrumpfen. Die logischen Breiten sind unterschiedlich. Der Zuschauer kann auch die nahe oder weite Ansicht von den verschiedenen Richtungen sehen. So wird Perspektivische Projektion auf 3D Spiel angewendet, wie aus Abbildung 3.14 ersichtlich ist.

  In dem folgenden Text wird die Unterschiede zwischen Perspektivische Projektion und Orthografische Projektion erklärt.

- gluPerspective, gluLookAt

  - gluPerspective (double fovy, double aspect, double zNear, double zFar);

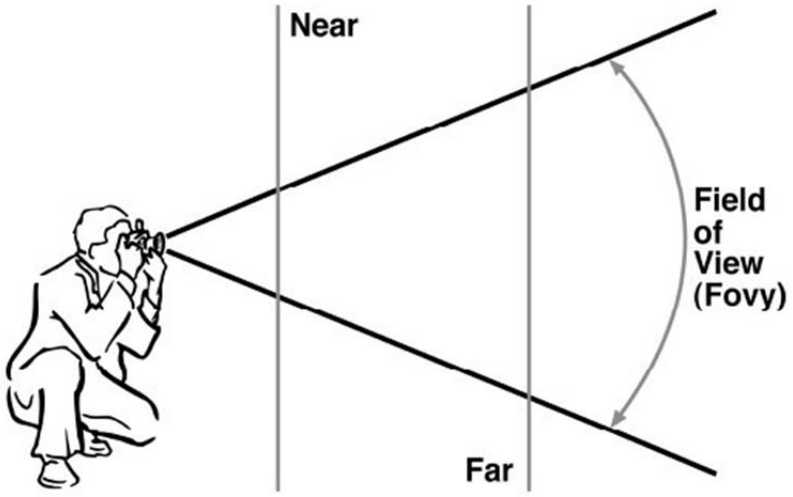


Abbildung 3. 14: Das Prinzip der Perspektivischen Projektion [22]

- fovy bezeichnet den vertikalen Blickwinkel in einem Intervall von 0° bis 180°.[19]
- aspect bezeichnet das Verhältnis zwischen horizontalen und vertikalen Bildschirmmaßen bzw. x/y.[19]

- Das Prinzip der Perspektivischen Projektion.

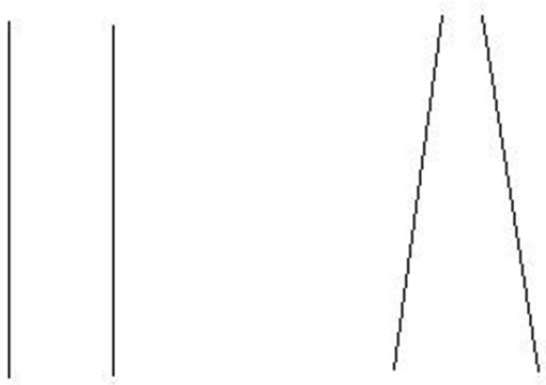


Abbildung 3. 15: Das Prinzip der Perspektivischen Projektion (eigene Darstellung)

- gluLookAt (double eyeX, double eyeY, double eyeZ, double centerX, double centerY, double centerZ, double upX, double upY, double upZ);
  - (double eyeX, double eyeY, double eyeZ) bezeichnet die Koordinate der

Augen.

- (double centerX, double centerY, double centerZ) bezeichnet die Koordinate des Referenzpunkts. Es stellt eine Sehenrichtung für die Augen.

- (double upX, double upY, double upZ) bezeichnet Angle-Breite. Die vorgegebene Koordinate ist (0, 1, 0).
- Wenn man kein gluLookAt benutzt, ist vorgegebene Koordinate (0, 0, 0). D.h. gluPerspective definiert die Koordinate der Augen. gluLookAt definiert die Koordinate und stellt Sehenrichtung der Augen.

In der Abbildung 3.16 ist ein Programm für Perspektivische Projektion dargestellt.

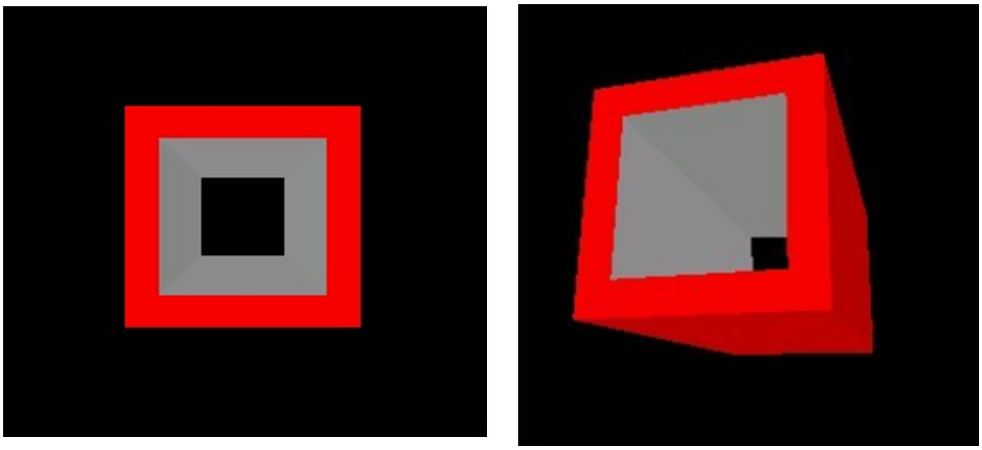


Abbildung 3. 16: Ein Programm für Perspektivischen Projektion (eigene Darstellung)

- glOrtho
  - glOrtho (double left, double right, double bottom, double top, double near_val, double far_val);
  - glOrtho richtet das Plane-Koordinatensystem ein.
    - left: linke Koordinate
    - right: rechte Koordinate
    - bottom: untere Koordinate
    - top: obene Koordinate

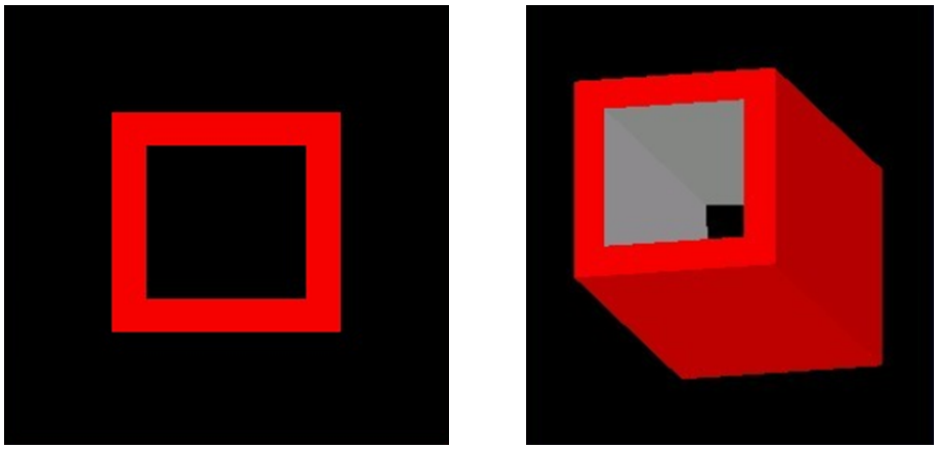


Abbildung 3. 17: Ein Programm für Orthografischen Projektion (eigene Darstellung)

3.8.3 Einstellung der Grafikkarte

- Im Kapitel 2.4.1 wurde festgestellt, dass bisher nur Nvidia Quadro Grafikkarten die Technologie Quad Buffer unterstützen. Aber muss man auch die Grafikkarte entsprechend einstellen.
- Hier wird die Einstellung der Grafikkarten Quadro FX 4800 und Quadro 5000 angezeigt.
- Zunächst ist das Fenster der Einstellung von Grafikkarte zu öffnen. Dann wird der Befehl „3D-Einstellungen verwalten“, die in der Abbildung 3.18 dargestellt ist, ausgewählt.

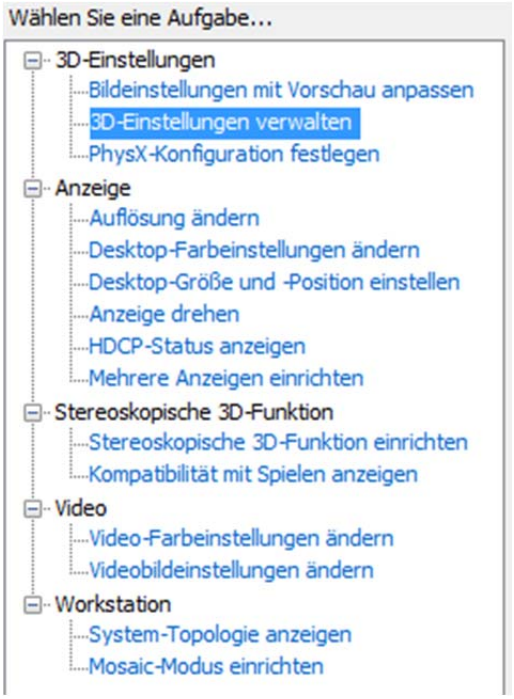


Abbildung 3. 18: 3D-Einstellung verwalten (eigene Darstellung)

- Jetzt kann man im rechten Fenster sehen, dass viele Möglichkeiten für „Globale Voreinstellungen“ angeboten werden. Darunter wird „3D OpenGL Stereo“, die in der Abbildung 3.19 gezeigt wird, ausgewählt.

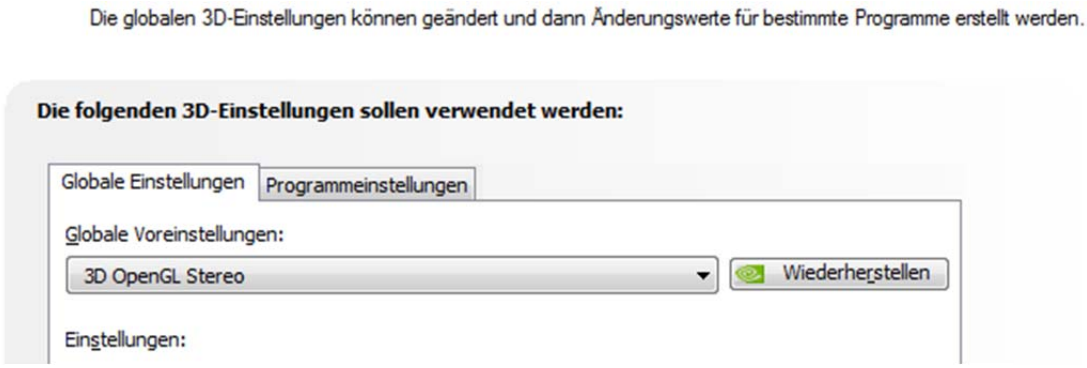


Abbildung 3. 19: 3D OpenGL Stereo (eigene Darstellung)

- In der Einstellungslist gibt es einige Optionen. Die sind Dreifach-Puffer, Overlay aktivieren, Stereo – Aktivieren, Stereo – Anzeigemodus, Stereo – Augen umkehren und Vertikale Synchronisierung. (siehe Abbildung 3.20)
- Empfohlen wird die folgende Einstellung.

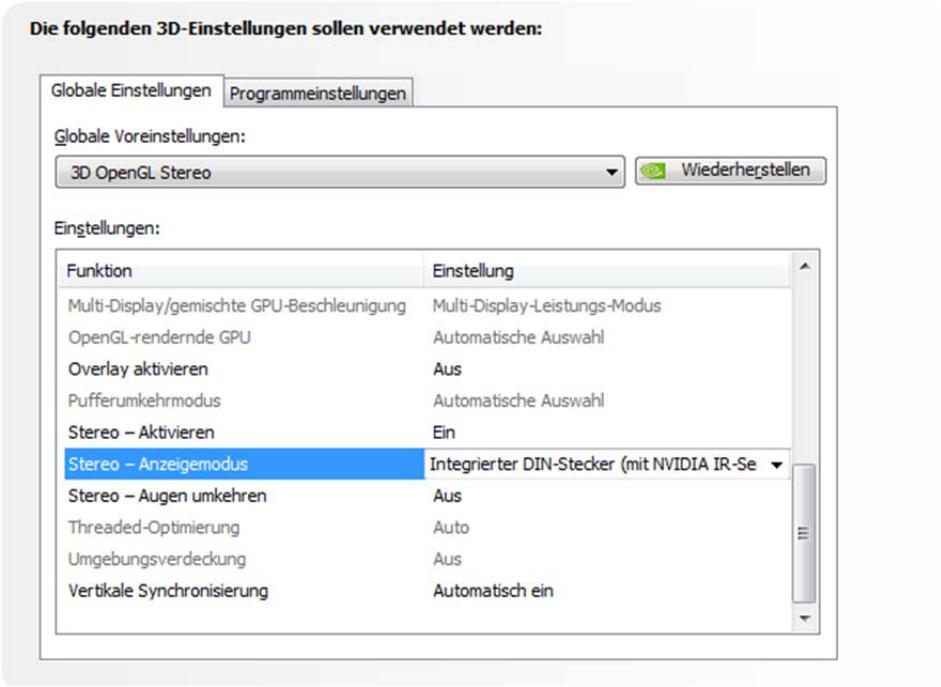


Abbildung 3. 20: Die Einstellung im 3D OpenGL Stereo (eigene Darstellung)

3.9 Passiv Stereo mit JOGL

JOGL kann auch passiv Stereo (Anaglyphe) realisieren. Aber muss man dann eine Farbfilterbrille verwenden. Die Farben können wie die Farben der Farbfilterbrille auf den Bildern definiert werden. Man muss auch einige wichtige Farbe-Methode von OpenGL kennen.

3.9.1 Grundlage

- glShadeModel
  - glShadeModel (Glenum mode) bezeichnet Flat-Shading-Modus und

Smooth-Shading-Modus und bestimmt ein Modus für die anderen Farben der Punkte zwischen zwei Punkte.

- mode: GL_FLAT und GL_SMOOTH, Default ist GL_SMOOTH
- Die Abbildung 3.21 zeigt den Unterschied.

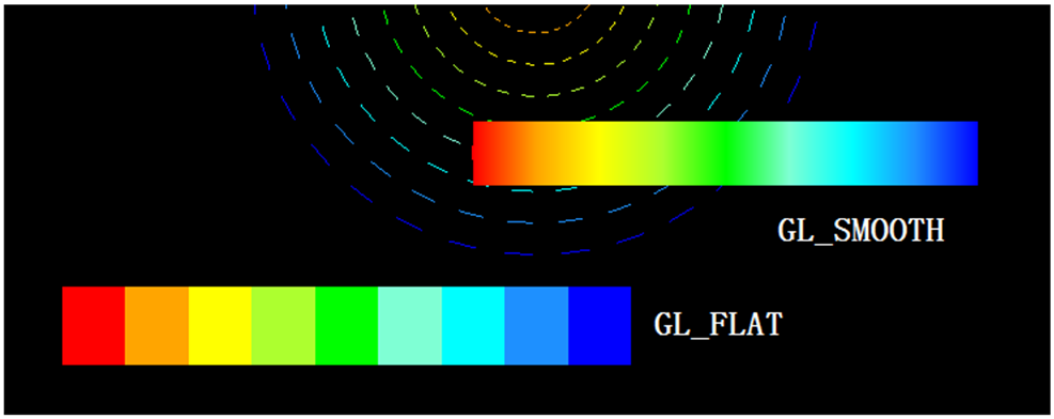


Abbildung 3. 21: Der Unterschied zwischen GL_SMOOTH und GL_FLAT [20]

- glColor
  - Einrichtung der Farbe
  - glColor (r, g, b)
    - r: rot-Intensität, g: grün-Intensität, b: blau-Intensität
    - Wertintervall ist von 0.0f bis 1.0f.
    - 0.0f bezeichnet schwarz.

- glClearColor
  - Lösung der Farbe
  - glClearColor (r, g, b, Alpha)
    - Alpha: Bildschirm zu löschen

- glColorMask

- Aktivieren oder Deaktivieren der Frame-Puffer RGB Farbe-Intensität.
- glColorMask (GLbool red, Glbool green, Glbool blue, Glbool alpha);
- Die Funktion von glColorMask bedeutet, ob die einzelne Farbe-Intensität in Frame-Puffer geschrieben wird.
- Bsp.: red ist GL_FALSE, d.h. red-Intensität kann nicht verändern.

In der Tabelle 14 ist die Definition der Funktion glColorMask für drei verschiedene Farbfilterbrillen.

linkes Auge	glColorMask
Rot-Cyan	glColorMask(true, false, false, true);
Cyan-Rot	glColorMask(false, true, true, true);
Rot-Grün	glColorMask(true, false, false, true);
Grün-Rot	glColorMask(false, true, false, true);
Rot-Blau	glColorMask(true, false, false, true);
Blau-Rot	glColorMask(false, false, true, true);
rechtes Auge	glColorMask
Rot-Cyan	glColorMask(false, true, true, true);
Cyan-Rot	glColorMask(true, false, false, true);
Rot-Grün	glColorMask(false, true, false, true);
Grün-Rot	glColorMask(true, false, false, true);
Rot-Blau	glColorMask(false, false, true, true);
Blau-Rot	glColorMask(true, false, false, true);

Tabelle 13: Die Definition der Funktion glColorMask für Farbfilterbrillen(eigene Darstellung)

In der Abbildung 3.22 zeigt ein Programm für passiv Stereo. Dadurch kann man mit

der rot-grün Farbfilterbrille schauen.

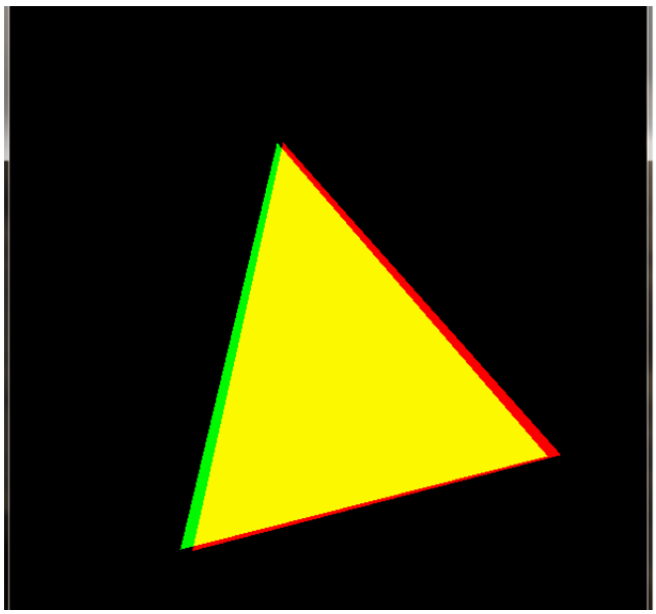


Abbildung 3. 22: Ein Programm für passiv Stereo (eigene Darstellung)

- glViewport
  - Setzen des Anzeigebereiches
  - glViewport (GLint x, GLint y, GLsizei width, GLsizei height);
  - x und y sind die Koordinaten für Links-Unter. (Einheit ist Pixel)

3.10 Processing

Processing ist eine Open-Source-Programmiersprache und eine Umgebung. Man kann über Processing die Grafik, Animation und Interaktionen erstellen. Nach der langen Entwicklung lernen und benutzen viele Leute Processing. Processing hat die folgenden Merkmale.

- frei unterladen und open-Source
- Export der interaktiven Programme auf 2D, 3D oder PDF
- OpenGL integrieren

- Unterstützung von Windows, Linux, Mac usw.
- Ausführung des Projekts online oder Export auf Application
- Über 100 Bibliotheken

Processing bietet auch die eigene Entwicklungsumgebung. Die ist „Processing Development Environment (PDE)“ genannt. PDE sieht wie Text-Editor aus, wie in der Abbildung 3.23 ersichtlich ist. Die aktuelle Version ist 1.5.1.

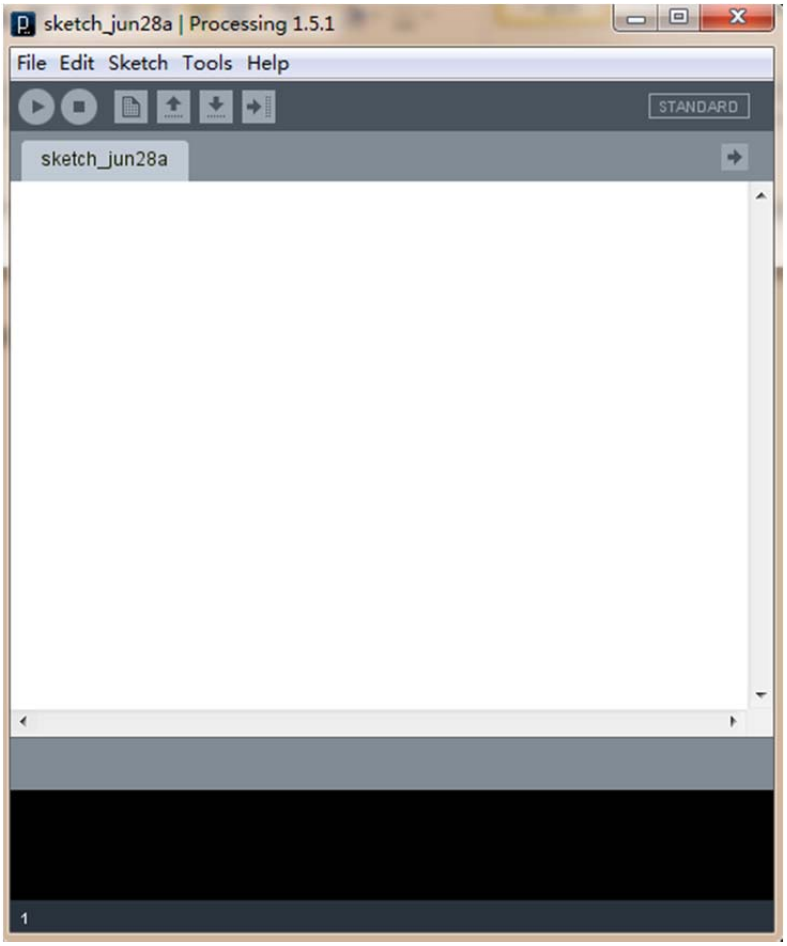


Abbildung 3. 23: Processing Development Environment (PDE) (eigene Darstellung)

Processing ist eine vereinfachte Java Sprache und integriert OpenGL, so kann die Stereoskopische Visualisierung theoretisch über Processing und OpenGL realisiert werden. Natürlich muss man die Hilfe der PDE benutzen. Die Dateityp wird mit .pde definiert.

- aktiv Stereo und passiv Stereo

In der Abbildung 3.24 zeigt ein Programm mit Processing für passiv Stereo.

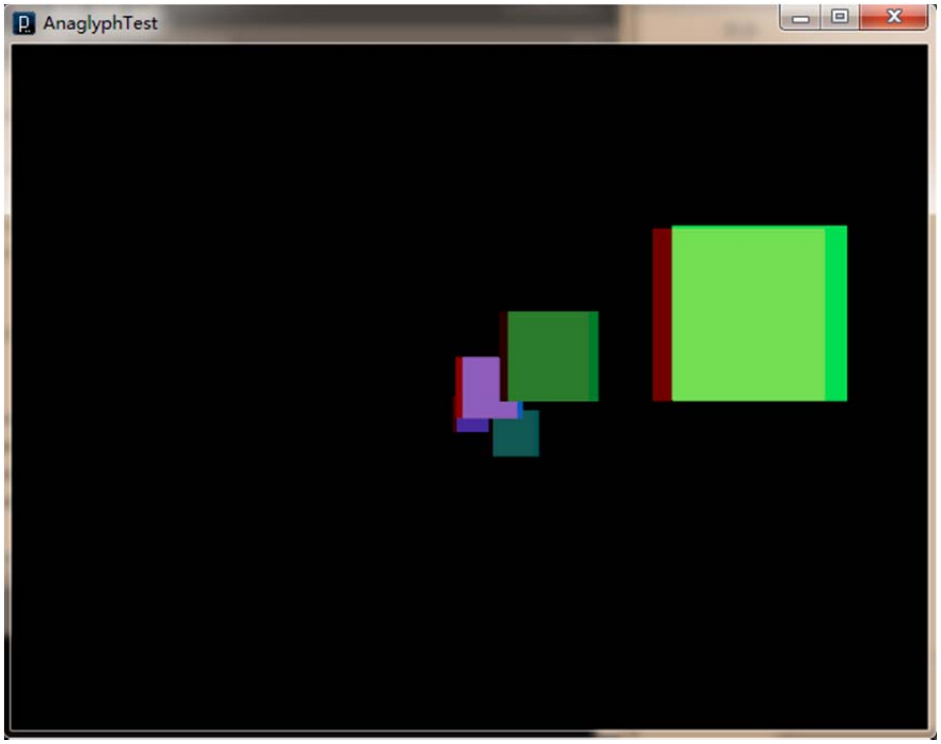


Abbildung 3. 24: Ein Programm mit Processing für passiv Stereo (eigene Darstellung)

Im Abschnitt 3.7 wird deutlich vorgestellt, dass die Stereoskopische Visualisierung über Farben realisiert kann werden. Das Prinzip des passiv Stereos mit Processing ist genau wie Kapitel 3.7. Die Animation und Grafik werden nur über Processing und OpenGL erstellt. Mehr Informationen können auf der Webseite http://processing.org/ gefunden werden.

4 Stereoskopische Visualisierung in jBEAM

Bis jetzt wird der Modul „Universal 3D-Graphic (OpenGL)“ der Software jBEAM mit der stereoskopischen Visualisierung realisiert. Dazu sind die Shutterbrille von Nvidia, 3D Monitor von Acer und 3D Nvidia Quadro Grafikkarte nötig. Natürlich kann man auch andere 3D Bildschirme bzw. 3D Beamer oder 3D TV verwenden.

Für die Benutzung müssen ein 3D-Points Diagram, 3D-Surface Diagram, 3D-Waterfall Diagram oder 3D-Bar Diagram zuerst erstellt werden. Dann soll man die rechte Maustaste auf dem Diagramm klicken und „stereoskopische Visualisierung“ ausgewählt. wie aus Abbildung 4.1 ersichtlich ist. In der Abbildung 4.1 zeigt ein 3D-Points Diagramm.

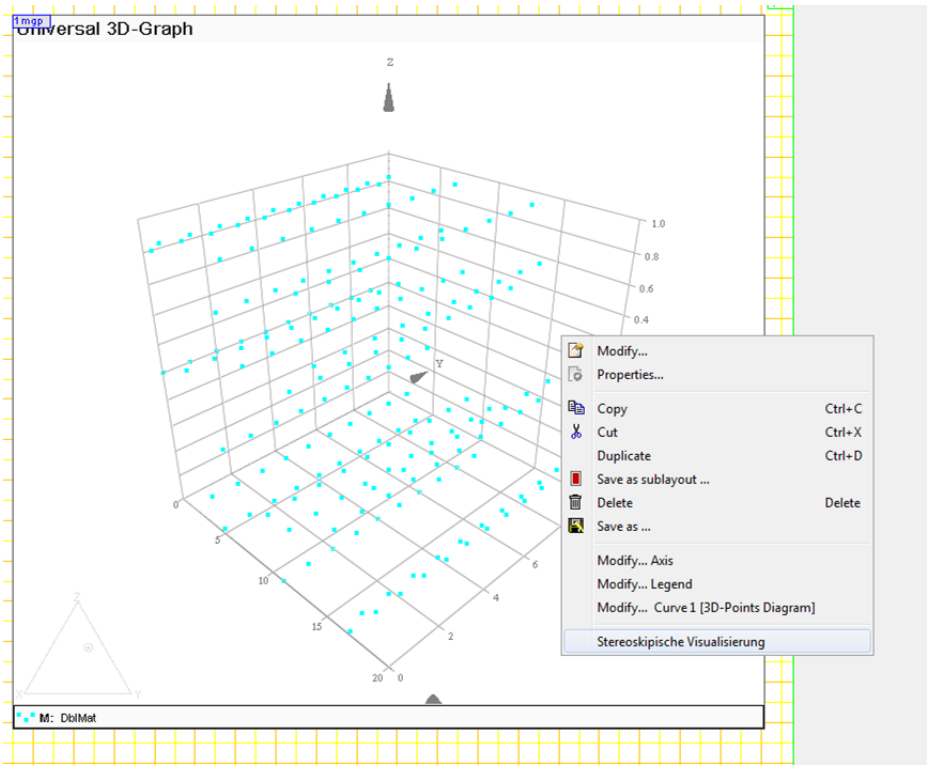


Abbildung 4. 1: Stereoskopische Visualisierung in jBEAM (eigene Darstellung)

Nach der Auswahl des Befehls „stereoskopische Visualisierung“ wird ein Fenster automatisch geöffnet und im Anfang das 3D Diagramm automatisch nach rechts gedreht. Wenn die linke Maustaste auf dem Fenster zu klicken ist, wird das 3D

Diagramm nicht weiter gedreht. Das Fenster wird 2 Teile über den folgenden Code eingeteilt, linken Teil und rechten Teil.

```
if (!is3DdragEvent(e)) {
    rotate = !rotate;
    if(e.getPoint().x < (this.getBounds().x + this.getBounds().width / 2))
        leftRotate = true;
    else
        leftRotate = false;
}
```

Z.B. Der linke Teil ist zu klicken, um das 3D Diagramm nach links zu drehen.

Man kann auch über Ctrl und linke Maustaste den Blickwinkel verändern.

In der Zukunft können mehrere Fähigkeiten des Moduls entwickelt werden:

- Das 3D Diagramm kann auch in die andere Richtung drehen.
- Die Form des Diagramms wird in dem Modul „stereoskopische Visualisierung“ frei verändert.
- Die Stereoskopische Visualisierung kann ohne Brille realisiert werden.

Bei der Realisierung gab es einige Probleme. Im folgenden Kapitel werden diese analysiert und Lösungen konzeptiert.

4.1 Problemanalyse und Lösungskonzeption

- GLCanvas und GLJpanel

GLCanvas ist eine komplexe Komponente, GLJpanel ist eine einfache Komponente. GLCanvas ist eine AWT-Komponente, welche OpenGL Renderingunterstützung anbietet. GLCanvas ist die Hauptimplementation des GLDrawable-Interfaces. Für die Realisierung der Stereoskopie soll es GLCanvas verwenden. Die Unterschiede werden im Kapitel 3.8.1 und 3.8.2 erläutert.

```
canvas = new GLCanvas(capa);
canvas.addGLEventListener(this);
```

- dynamische Veränderung

Animator ruft die display()-Methode einer oder mehrer GLAutoDrawable Insatzen in einer Schleife auf. Kreiert einen Hintergrund-Thread um diese Aufgabe zu erfüllen. FPSAnimator ist eine Unterklasse des Animator. Hier kann man eine Frames-Per-Second-Rate angeben und dadurch bestimmen wie oft die display()-Methode aufgerufen wird. Dabei wird das Erfüllen der Rate nicht garantiert aber angestrebt.

```
final FPSAnimator animator = new FPSAnimator(canvas, 60, true);

animator.start();
```

- Canvas-Initialisierung

```
private void initCanvas() {
    GLCapabilities capa = new GLCapabilities();
    capa.setStereo(true);
    canvas = new GLCanvas(capa);
    canvas.addGLEventListener(this);
    final FPSAnimator animator = new FPSAnimator(canvas, 60, true);

    if (graph.isEnabled())
        registerMouseListeners();
    canvas.reshape(canvas.getX(), canvas.getY(), canvas.getWidth(), canvas.getHeight());
    setLayout(new BorderLayout());
    add(canvas, BorderLayout.CENTER);
    rotate = true;
    animator.start();
```

In der Methode initCanvas enthält die wichtige Aufruf und Schalter für Stereo buffering. Er muss true sein.

```
capa.setStereo(true);
```

Die Aufruf kann man direkt in der Main Methode definieren.

- Zeichenvorgang

Die Methode display() wird bei jedem Zeichenvorgang aufgerufen. In der Methode

soll man die Farben, die Bewegung bzw. Rotation und Zwei Augen für Stereo buffering der Grafik definieren usw..

Die Rotation wird eine feste Geschwindigkeit und Rotationsrichtung im Anfang definiert.

```
if (rotate) {
    if (leftRotate)
        phi += 0.007f;
    else
        phi -= 0.007f;
}

calcCamPos(theta, phi);
```

Die Methode calcCamPos definiert die Position der Kameras in den drei Achsen.

```
private void calcCamPos(float theta, float phi) {
    camX = (float) (radius * Math.sin(theta) * Math.cos(phi));
    camY = (float) (radius * Math.cos(theta));
    camZ = (float) (radius * Math.sin(theta) * Math.sin(phi));
    repaint();
}
```

Sie benutzt zwei Variablen theta und phi.

```
theta                       = (float) (2 * Math.PI / 3);
phi                         = (float) (1.75 * Math.PI);
```

Quad Buffer für zwei Augen:

```
double delx = camX - (float) (radius * Math.sin(theta) * Math.cos(phi - 0.02));
double delz = camY - (float) (radius * Math.cos(theta));
double dely = camZ - (float) (radius * Math.sin(theta) * Math.sin(phi - 0.02));
_gl.glDrawBuffer(GL.GL_BACK_LEFT);
_gl.glClear(GL.GL_COLOR_BUFFER_BIT | GL.GL_DEPTH_BUFFER_BIT);
_gl.glPushMatrix();
_gl.glTranslated(delx, dely, delz);
drawElements(_gl);
_gl.glPopMatrix();
```

```
double delx = camX - (float) (radius * Math.sin(theta) * Math.cos(phi + 0.02));
double delz = camY - (float) (radius * Math.cos(theta));
double dely = camZ - (float) (radius * Math.sin(theta) * Math.sin(phi + 0.02));
_gl.glDrawBuffer(GL.GL_BACK_RIGHT);
_gl.glClear(GL.GL_COLOR_BUFFER_BIT | GL.GL_DEPTH_BUFFER_BIT);
_gl.glPushMatrix();
_gl.glTranslated(delx, dely, delz);
drawElements(_gl);
_gl.glPopMatrix();
```

Die Quad Buffer ist sehr wichtig Technologie für Stereoskopische Visualisierung. Jedes Auge wird Stereo buffering definiert. Die Bild wird durch Methode drawElements(_gl) aufgerufen. Der Abtand ist 0.02.

Die Grafik werden nach kinks und rechts mit dem Abstand 0.02 verschoben. Wie der Aber die Z Ache wird nicht verschoben. glDrawBuffer schreibt die Puffer. glClear löscht das Puffer und hält den zuvor definierten Konfigurationswert.

drawElements(_gl) ruft die Methode drawElements auf, die Methode zeichnet das Diagramm, bzw. Punkte und Achsen.

- Maus Event

Durch die Methode mousePressed und mouseDragged kann man die Grafik kontrollieren, bsp. Grafik vergrößern oder verkleinen, die Ansicht der Grafik wechseln mit Ctrl.

Für die Realisierung der Maus Event soll man die Methode Animator verwenden. Davon verändert die Grafik dynamisch.

- Stereoskopie Effekt

Bis jetzt wird der Modul „Universal 3D-Graphic (OpenGL)“ der Software jBEAM mit der stereoskopischen Visualisierung realisiert. Aber dies gibt keinen guten stereoskopischen Effekt. Bei Nvidia gibt es ein paar 3D Fotos zum Test der stereoskopische Visualisierung .

Für die 3D Fotos wird die D3D Technologie verwendet. Die Fotos enthalten viele Elemente, jedes Element hat einen unterschiedlichen Abstand zur Betrachterebene. Die „Tiefenwirkung“ ist besser, was bei OpenGL ist leider nicht der Fall ist. Die Ursache ist nicht ganz klar. Aber es gibt einige Unterschiede zwischen D3D und OpenGL.

Der Modul „Universal 3D-Graphic (OpenGL)“ der Software jBEAM zeichnet ein Diagramm, das Diagramm enthält die Werte und Achsen. Die zwei Augen werden nur

einmal definiert, bzw. nur ein Element wird für den Abstand für das Stereo buffering definiert. Die weitere Unterschiede zwischen D3D und OpenGL werden im Kapitel 4.2 erläutert.

4.2 D3D und OpenGL

Direct3D(D3D) ist ein 3D API und kommt aus Microsoft, D3D wird in der Bereichen Computer-Spiel und Multimedia verwendet.

OpenGL ist auch ein API, aber OpenGL wird in der Bereichen 3D Grafik verwendet.

- Obwohl D3D und OpenGL 3D API sind, OpenGL ruft direkt die Treiber auf, ob OpenGL besser als D3D ist?
- Man muss Shader immer zwei schreiben, sehr kompliziert.
- Es gibt viele Unterschiede aud der Basis von OpenGL und D3D.
- Wenn die Grafik-Engine die OpenGL und D3D kongruieren, d.h. Die Fähigkeit ist nur von die Schnittmenge von OpenGL und D3D, die ist sehr schlecht.

In der folgende Arbeit werden die Vergleich und Diskussion zwischen D3D und OpenGL erläutert.

- Die Architektur von OpenGL und D3D

In der Abbildung 4.2 sind die Architektur von D3D und OpenGL dargestellt.

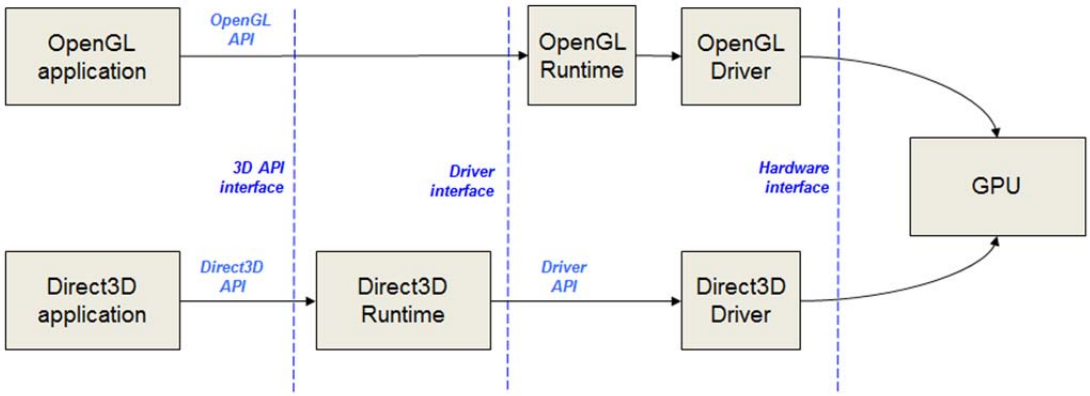


Abbildung 4. 2: Die Architektur von D3D und OpenGL

Die Runtime von D3D ist Bestandteil des OS, Runtime von OpenGL wird durch einen eigenen Treiber realisiert. Aber das ist noch nicht die Hauptursache.

- Shader

„shading language“ von OpenGL ist GLSL, D3D ist HLSL. Die Zwei Sprachen sind nur ähnlich. Cg(Computer graphics) von Nvidia ist wie HLSL.

- Fähigkeiten

Die aktuelle version von D3D ist D3D11. Die Fähigkeiten von D3D9 ist sehr ähnlich mit OpenGL 2.0. In der folgende Tabelle 15 zeigt die verschiedenen Fähikgkeiten von D3D und OpenGL.

D3D 10	OpenGL
Geomrtry shader	GL_ARB_geometry_shader4 oder OpenGL 3
Stream output	GL_EXT_transform_feedback oder OpenGL 3
State Objekt	keine
Constant buffer	GL_ARB_uniform_buffer_object oder OpenGL 3
Texture array	GL_EXT_texture_array +GL_ARB_texture_compression_rgtc +GL_ARB_texture_rg +GL_ARB_texture_rgb10_a2ui +GL_EXT_texture_integer oder OpenGL 3
Texture und sampler	GL_ARB_sampler_objects oder OpenGL 3
shader	GL_ARB_shader_bit_encoding oder OpenGL 3
Multisampled alpha-to-coverage	GL_NV_multisample_coverage 或 OpenGL 3
D3D 10.1	OpenGL

Read multisample depth/stencil Textur	GL_ARB_texture_multisample oder OpenGL 3
Cubemap array	GL_ARB_texture_cube_map_array oder OpenGL 4
gather4	GL_ARB_texture_gather oder OpenGL 4
D3D 11	OpenGL
Compute Shader	GL_ARB_cl_event + OpenCL
Dynamic Shader Linkage	GL_ARB_gpu_shader5 oder OpenGL 4
Multithreading	keine
Tessellation	GL_ARB_tessellation_shader oder OpenGL 4

Tabelle 14: Die verschiedenen Fähikgkeiten von D3D und OpenGL

OpenGL ist nur ein Gragik-Bibliothek, aber D3D enthält Grafik, Audio, Input und Netzwerk usw.. Aber D3D ist junger als OpenGL.

Jetzt wählen die viele Herstellers der Computer-Spiel D3D aus, aber wenn sie auf der änderer Betriebssystemen entwickeln möchten, sie AutoCAD und verteilt System entwickeln möchten, müssen OpenGL auswählen.

Aufgrund die schnelle Geschwindigkeit der D3D wählen die viele Herstellers D3D aus.

Microsoft entwicklt D3D, aber blockiert OpenGL, das Betriebssystem Windows unterstützt D3D besser als OpenGL.

Über das Computer-Spiel CS, man kann vergleichen, d.h. das Spiel mit OpenGL oder D3D spielen, will man verschiebene Bildqualität finden. In dem Spiel CS gibt es ganz kompliziert Umfang, z.B. viele Elemente, die dynamische Verschiebung, das Wetter usw.. Die werden den guten Stereoskopie Effekt erzeugen. Die 3D Fotos von Nvidia ist auch so, viele Elemente stehen im Foto, jedes Element hat verschiedene Abstand für Stereo buffering. Davon kann man sehr guten Stereoskopie Effekt sehen.

In jBEAM gibts es nicht so guten Stereoskopie Effekt, die Quad Buffer von OpenGL werden richtig definiert, weil OpenGL hat nur die Lösungsweg, um Stereoskopie

realisieren. Aber bei jBEAM verwendet OpenGL1.1 und das Diagramm als die Ganzheit angezeigt, so das ist vielleicht die Ursache wegen verschiedener Effekt.

Bei dem Stereoskopie Effekt ist es natürlich unterschied. In der Tabelle 15 kann man auch verstehen, die Fähigkeiten der D3D ist mehr als OpenGL, jetzt meiste Grafikkarten unterstützen schon bis D3D11. Es ist klar, D3D11 ist viele besser als D3D9 und D3D10, bei OpenGL ist auch so und gibt es viele Versionen, die Hohe-Version ist natürlich besser als Niedrig-Version, weil es mehre neue Fähigkeit von Hohe-Version gibt.

Die Spiele-Entwickler sind auf gute Verkaufszahlen angewiesen. Bei einem Spiel sind Grafik und Entertainment wichtig. D.h. neue Grafik-Technologie ist beliebt.

OpenGL entwickelt sich langsamer als D3D.

Aber unter OpenGL kann der Grafikkarten-Hersteller Extensions hinzufügen. D.h. OpenGL kann auch schnell die gleichen Fähigkeiten wie D3D entwickeln.

Da die Extensions von verschiedenen Herstellern stammen, gibt es oft Probleme mit der Kompatibilität, Patenten usw.

Im Gegensatz zu CAD sind Spiele Consumer Market. Der Spiele-Entwickler sucht die beste Lösung für alle PCs (gute Kompatibilität). Spiele brauchen unbedingt die 3D-Hardwarebeschleunigung. D3D bietet alle Driver für das Interface.

Bei OpenGL braucht man einen OpenGL ICD Driver, aber die Grafikkarten bringen ständig neuere Fähigkeiten mit, d.h. der OpenGL ICD Driver ist nie aktuell. Zwar kann man noch GLSetup benutzen, was aber veraltet ist.

Bei CAD braucht man nicht die neuesten Fähigkeiten, nur die Grundfähigkeiten von OpenGL.

Aber im Spiel braucht man viele neue Fähigkeiten bsp. Effect Framework, Bump Mapping, HDR, Texture Atlas usw..

Aber OpenGL kann auf allen Betriebssystemen laufen.

Die obere Diskussionen zwischen D3D und OpenGL sind auch die Hinweise, erst mal Update, danach noch mal vergleichen. Obwohl die Arbeit lange dauert.

5 Zusammenfassung und Ausblick

Die Bachlorarbeit betrachtet die stereoskopischen Visualisierung mit Java und JOGL. Es werden Grundkenntnisse der Stereoskopie, die 3D Hardware und ihre Realisierungsweisen behandelt und D3D und OpenGL verglichen. Für die Realisierung der Stereoskopie gibt es die Auswahl:

- Farbfilterbrille (Anaglyphenbrille) für beliebige Bildschirme
- Polfilterbrille mit 3D Bildschirm
- Shutterbrille mit 3D Bildschirm (120Hz)
- Stereoskopie ohne Brille

Die Stereoskopie kann in Anwendungen integriert werden, um eine professionelle Darstellung durch Stereoskopie zu realisieren.

6 Quellenverzeichinis

Internet

[1]http://jbeam.de/index.php?id=96&tx_ttnews[pointer]=1&cHash=b22adc40157c71a3543d33b27eca876e

[2]http://www.reald.com/

[3]http://www.nvidia.cn/object/product-geforce-3d-vision-wired-glasses-cn.html

[4]http://www.nvidia.com/object/product-geforce-gt-550m-us.html

[5]http://www.nvidia.com/page/workstation.html

[6]www.amd.com

[7]http://product.pcpop.com/000118355/Index.html

[8]http://www.pconline.com.cn/projector/special/1001/2030445_2.html

[9]http://tech.163.com/mobile/11/0519/06/74D9CE8U00112K88.html

[10]http://news.jinti.com/shangpin/791166.htm

[11]http://ndevil.com/gadmei-p83-pmp-tablet-kann-3d-ohne-brille-fur-179-dollar-2821203/

[12]http://en.wikipedia.org/wiki/1080i

[13]http://www.amd.com/us/products/technologies/amd-hd3d/Pages/supported-hardware.aspx

[14]http://www.pcpop.com/doc/0/652/652946.shtml

[15]http://mobile.zol.com.cn/237/2375460.html

[16]http://www.chinanews.com/cul/2011/04-30/3009659.shtml

[17]http://www.cgtiger.com/ch/yingyuan.asp

[18]https://www.staff.hs-mittweida.de/~stockman/intranet/grafik/02_Open_GL_JOGL.pdf

[19]https://www.staff.hs-mittweida.de/~stockman/intranet/grafik/04_JOGL_3D.pdf

[20]http://blog.csdn.net/Wadejr/article/details/4504112

[21]https://www.staff.hs-mittweida.de/~stockman/intranet/grafik/

[22]http://www.amazon.com/Learning-Java-Bindings-OpenGLL-JOGL/dp/14208036

2X

Vorlesungsskripte

Herr Stockmann, Daniel: Grafiksysteme, Hochschule Mittweida, 2010

Bücher

Jim X. Chen; Chunyang Chen: Foundations of 3D Graphics Programming, Using JOGL and Java3D, Second Edition

7 Anhang

7.1 Klassenhierarchie

- class java.lang.**Object**
  - class com.sun.opengl.util.**Animator**
    - class com.sun.opengl.util.**FPSAnimator**
  - class javax.media.opengl.**AWTGraphicsConfiguration** (implements javax.media.opengl.AbstractGraphicsConfiguration)
  - class javax.media.opengl.**AWTGraphicsDevice** (implements javax.media.opengl.AbstractGraphicsDevice)
  - class com.sun.opengl.util.**BufferUtil**
  - class com.sun.opengl.cg.**CGannotation**
  - class com.sun.opengl.cg.**CGcontext**
  - class com.sun.opengl.cg.**CGeffect**
  - class com.sun.opengl.cg.**CgGL**
  - class com.sun.opengl.cg.**CGparameter**
  - class com.sun.opengl.cg.**CGpass**
  - class com.sun.opengl.cg.**CGprogram**
  - class com.sun.opengl.cg.**CGstate**
  - class com.sun.opengl.cg.**CGstateassignment**
  - class com.sun.opengl.cg.**CGtechnique**
  - class java.awt.**Component** (implements java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable)
    - class java.awt.**Canvas** (implements javax.accessibility.Accessible)
    - class javax.media.opengl.**GLCanvas** (implements javax.media.opengl.GLAutoDrawable)
    - class java.awt.Container
      - class javax.swing.JComponent (implements java.io.Serializable)

- class javax.swing.JPanel (implements javax.accessibility.Accessible)
    - class javax.media.opengl.GLJPanel (implements javax.media.opengl.GLAutoDrawable)
- class java.awt.Panel (implements javax.accessibility.Accessible)
    - class java.applet.Applet
        - class com.sun.opengl.util.JOGLAppletLauncher

- class com.sun.opengl.util.texture.spi.DDSImage
- class com.sun.opengl.util.texture.spi.DDSImage.ImageInfo
- class javax.media.opengl.DebugGL (implements javax.media.opengl.GL)
- class javax.media.opengl.DefaultGLCapabilitiesChooser (implements javax.media.opengl.GLCapabilitiesChooser)
- class com.sun.opengl.util.FileUtil
- class com.sun.opengl.util.Gamma
- class javax.media.opengl.GLCapabilities (implements java.lang.Cloneable)
- class javax.media.opengl.GLContext
- class javax.media.opengl.GLDrawableFactory
- class javax.media.opengl.glu.GLU
- class com.sun.opengl.util.GLUT
- class javax.media.opengl.glu.GLUtessellatorCallbackAdapter (implements javax.media.opengl.glu.GLUtessellatorCallback)
- class com.sun.opengl.util.ImageUtil
- class com.sun.opengl.util.j2d.Overlay
- class com.sun.opengl.util.Screenshot
- class com.sun.opengl.util.texture.spi.SGIImage
- class com.sun.opengl.util.StreamUtil
- class com.sun.opengl.util.j2d.TextRenderer
- class com.sun.opengl.util.j2d.TextRenderer.DefaultRenderDelegate (implements com.sun.opengl.util.j2d.TextRenderer.RenderDelegate)

- class com.sun.opengl.util.texture.Texture
- class com.sun.opengl.util.texture.TextureCoords
- class com.sun.opengl.util.texture.TextureData
- class com.sun.opengl.util.texture.TextureIO
- class com.sun.opengl.util.j2d.TextureRenderer
- class com.sun.opengl.util.texture.spi.TGAImage
- class com.sun.opengl.util.texture.spi.TGAImage.Header
- class com.sun.opengl.util.TGAWriter
- class javax.media.opengl.Threading
- class java.lang.Throwable (implements java.io.Serializable)
    - class java.lang.Exception
        - class java.lang.RuntimeException
            - class com.sun.opengl.cg.CgException
            - class javax.media.opengl.GLException
- class com.sun.opengl.util.TileRenderer
- class javax.media.opengl.TraceGL (implements javax.media.opengl.GL)

7.2 Interfacehierarchie

- interface javax.media.opengl.AbstractGraphicsConfiguration
- interface javax.media.opengl.AbstractGraphicsDevice
- interface javax.media.opengl.ComponentEvents
    - interface javax.media.opengl.GLAutoDrawable (also extends javax.media.opengl.GLDrawable)
        - interface javax.media.opengl.GLPbuffer
- interface java.util.EventListener
    - interface javax.media.opengl.GLEventListener
- interface javax.media.opengl.GL

- interface javax.media.opengl.GLCapabilitiesChooser
- interface javax.media.opengl.GLDrawable
    - interface javax.media.opengl.GLAutoDrawable (also extends javax.media.opengl.ComponentEvents)
        - interface javax.media.opengl.GLPbuffer
- interface javax.media.opengl.glu.GLUnurbs
- interface javax.media.opengl.glu.GLUquadric
- interface javax.media.opengl.glu.GLUtessellator
- interface javax.media.opengl.glu.GLUtessellatorCallback
- interface com.sun.opengl.util.j2d.TextRenderer.RenderDelegate
- interface com.sun.opengl.util.texture.TextureData.Flusher
- interface com.sun.opengl.util.texture.spi.TextureProvider
- interface com.sun.opengl.util.texture.spi.TextureWriter

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, die vorliegende Bachelorarbeit mit dem Thema „Stereoskopische Visualisierung technischer Daten mit Java und JOGL“ selbständig und nur unter Zuhilfenahme der im Literaturverzeichnis angegebenen Quellen erstellt zu haben. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Chemnitz, den 07.06.2011

_____

Fang, Yi