

# Computergrafik.Online

## Drehbuch Shader

Hochschule Furtwangen University

Fakultät Digitale Medien

Betreut von:

Prof. Jirka Dell'Oro-Friedl

Version: 1.0

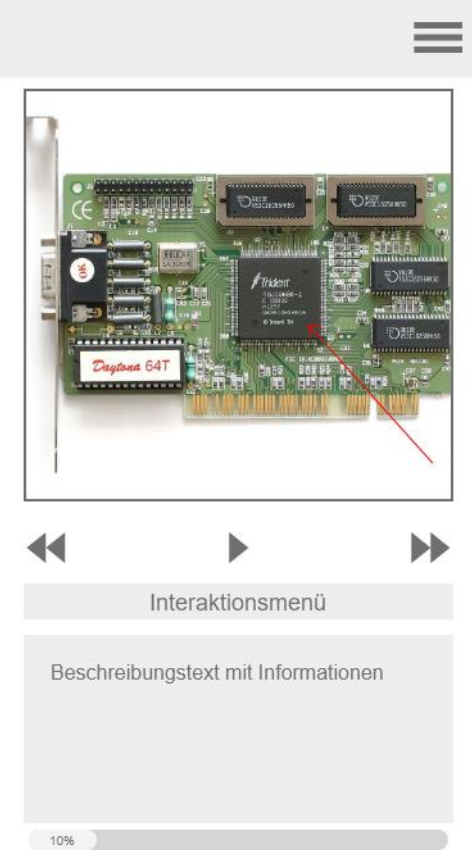
Letzte Änderung: 26.06.2018

Autor: Steven Romanek


# Inhalt

1. Einführung.....	3
2. Vertex Shader .....	5
3. Vertex Shader - Interaktion.....	6
4. Geometry-Shader .....	7
5. Pixel- / Fragment-Shader .....	9
6. Flat-Shading.....	10
7. Flat-Shading - Interaktion.....	12
8. Gouraud-Shading .....	13
9. Phong-Shading .....	14
10. Vergleich zwischen Flat-, Gouraud - und Phong-Shading - Interaktion .....	15
11. Toon-/Cel-Shading.....	16
12. Code Beispiel - Interaktion .....	18

# 1. Einführung

Screen	Sprechertexte	Stichwörter / Notizen	Regieanweisungen
	<p><b>110101</b>            Shader kommen in Computerspielen, in der Postproduktion von Videoinhalten und bei Computer Generated Imagery, kurz CGI, zum Einsatz. Sie waren ursprünglich, wie der Name schon sagt, für das Schattieren bzw. Erzeugen von verschiedenen Stufen von Licht, Dunkelheit und Farben in der Computergrafik zuständig. Heutzutage übernehmen Shader aber auch Aufgaben, die nichts mit dem ursprünglichen Schattieren zu tun haben.</p> <p><b>110102</b>            Die Verarbeitung von Shadern findet in den sogenannten Shadereinheiten statt. Diese befinden sich in Grafikchips von Grafikkarten, welche ein wichtiger Bestandteil in Computern, Spielekonsolen, Smartphones und anderen vergleichbaren Geräten sind. In Grafikkarten gibt es eine sogenannte Grafikpipeline, welche die Reihenfolge der auszuführenden Shader festlegt.</p> <p>Damit man Shader nutzen kann, ist noch eine programmierbare Schnittstelle nötig, wie z.B. DirectX oder OpenGL. Diese Programmierschnittstellen, auch application</p>	<p><b>110101</b>            Einsatz in Computerspielen, Postproduktion und bei CGI.</p> <p><b>110102</b>            - Werden in Shadereinheiten verarbeitet            - Grafikkarten sind Bestandteil von Computern, Smartphones und anderen Geräten            - Schnittstellen DirectX und OpenGL            - glTF -&gt; JPEG des 3D</p>	<p><b>110101</b>            Es werden nach und nach die Beispiele der Einsatzzwecke eingeblendet.</p> <p><b>110102</b>            Als erstes wird eine normale Grafikkarte eingeblendet und mit einem Pfeil gezeigt, wo sich darauf der Grafikchip befindet.</p> <p>Das Bild der Grafikkarte wird verkleinert und darunter erscheinen drei Geräte (PC, Smartphone, Spielekonsole). Daraufhin werden symbolisch drei Grafikkarten in diese Geräte geschoben, um</p>

	<p>programming interface genannt, sind bei der Entwicklung von 2D- und 3D-Computergrafikanwendungen üblich. Das GL Transmission Format, kurz glTF, genießt auch eine immer größere Verbreitung, da es unter anderem API neutral arbeitet. glTF basiert auf JSON und wird auch als JPEG des 3D bezeichnet.</p>		<p>hervorzuheben, dass diese Bestandteil solcher Geräte sind.</p>
--	---	--	---




Navigation: << < > >>

Interaktionsmenü

Beschreibungstext mit Informationen

10%




Navigation: << < > >>

Interaktionsmenü

Beschreibungstext mit Informationen

10%

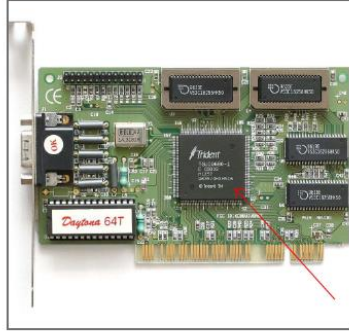


Navigation: << < > >>

Interaktionsmenü

Beschreibungstext mit Informationen

10%




Navigation: << < > >>

Interaktionsmenü

Beschreibungstext mit Informationen

10%



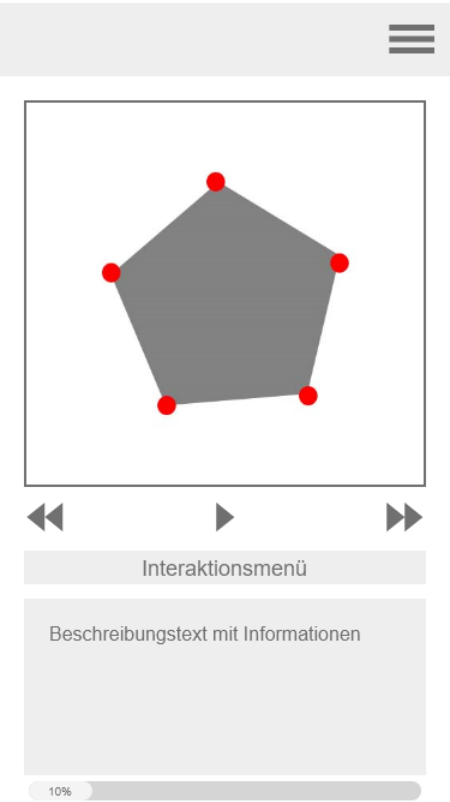
Navigation: << < > >>

Interaktionsmenü

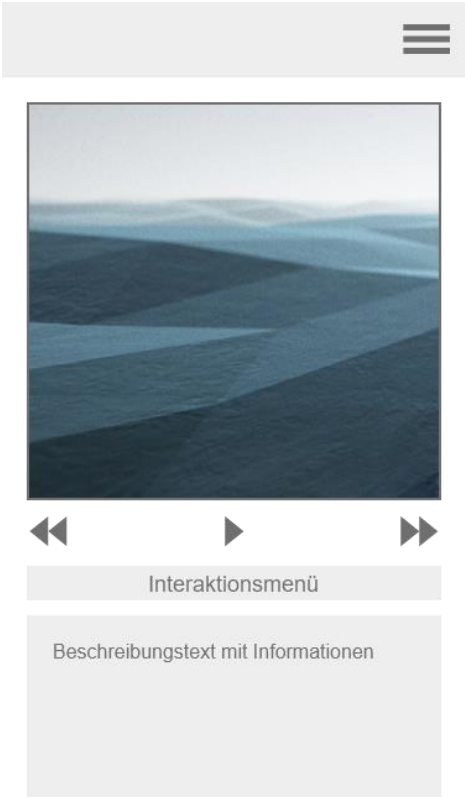
Beschreibungstext mit Informationen

10%

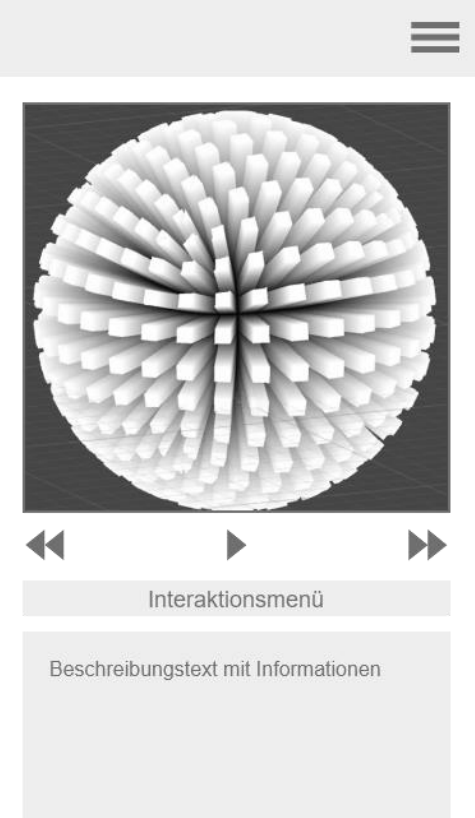
## 2. Vertex Shader

Screen	Sprechertexte	Stichwörter / Notizen	Regieanweisungen
	<p><b>110201</b> Vertex-Shader sind Programme, welche im Verlauf der Grafikpipeline in den Shadereinheiten einer Grafikkarte ausgeführt werden. Diese verarbeiten die sogenannten Vertices, bei denen es sich um Eckpunkte eines 3D-Modells handelt.</p> <p>Mit Vertex-Shadern lässt sich die Geometrie und somit die Form von Objekten beeinflussen, was sich wiederum auch auf die Beleuchtung niederschlagen kann. Dafür werden die Koordinaten der einzelnen Vertices im dreidimensionalen Raum für die zweidimensionale Darstellung transformiert. Da der Shader aber pro Vertex aufgerufen wird, kann er keine neuen Punkte zum 3D-Modell hinzufügen.</p>	<p><b>110201</b> - verarbeiten Vertices - Einfluss auf Geometrie und somit auch Beleuchtung</p>	<p><b>110201</b> Die Vertices werden nach und nach auf dem Objekt markiert.</p>

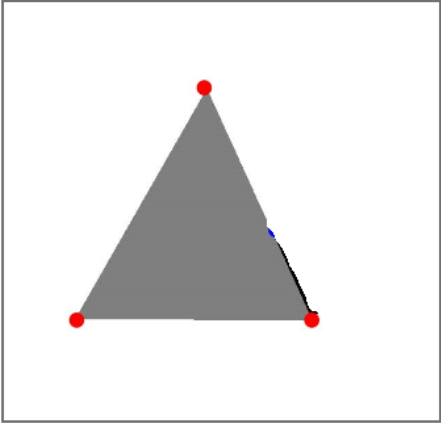
### 3. Vertex Shader - Interaktion

Screen	Sprechertexte	Stichwörter / Notizen	Regieanweisungen
 <p>The screenshot shows a 3D rendered landscape with rolling dunes in shades of blue and white. The interface elements include a top navigation bar with a hamburger menu icon, a central image area, navigation arrows (back, forward, and a double forward arrow), an 'Interaktionsmenü' button, and a larger box for 'Beschreibungstext mit Informationen'.</p>	-	-	<b>110301</b> Der User kann beispielhaft an einem Eckpunkt/Vertex ziehen oder einen Regler betätigen, sodass sich die Geometrie der Wellen verändert.

## 4. Geometry-Shader

Screen	Sprechertexte	Stichwörter / Notizen	Regieanweisungen
	<p><b>110401</b> Der Geometry-Shader wird in der Grafikpipeline nach dem Vertex Shader aufgerufen, um neue primitive Geometrien aus bereits vorhandenen Punkten, Linien und Dreiecken zu erzeugen und diese erneut in die Grafikpipeline einzufügen.</p> <p>Beispiele für die Anwendung des Geometry-Shader sind die Erzeugung von Schattenvolumen oder die Erzeugung von Fell- oder Haargeometrie. Wenn z.B. an einem Dreieck gearbeitet wird, erhält der Geometry-Shader drei Vertices vom Vertex-Shader als Input. Nach der Verarbeitung werden die fertigen Fragmente an den Pixel-Shader weitergegeben.</p> <p><b>110402</b> Anhand einer Kugel kann man auch sehen wie der Geometry-Shader genutzt wird um Haare zu erzeugen.</p>	<p><b>110401</b> - Erzeugung neuer Geometrien</p>	<p><b>110401</b> Mithilfe der Vertices wird beispielhaft ein Schatten erzeugt.</p> <p><b>110402</b> Danach wird als Beispiel eine Kugel gezeigt bei der durch Anwendung des Geometry-Shaders Haare erzeugt werden.</p>

☰



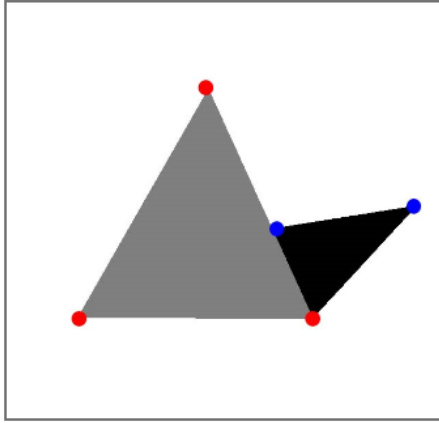
◀◀ ▶ ▶▶

Interaktionsmenü

Beschreibungstext mit Informationen

10%

☰



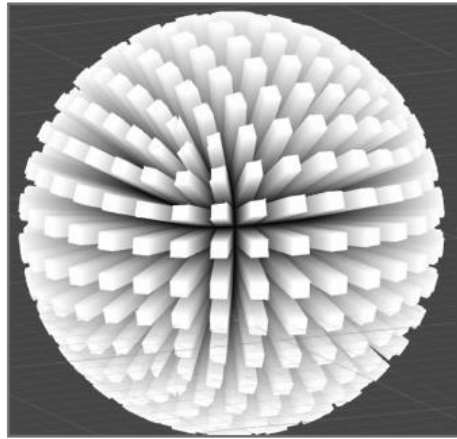
◀◀ ▶ ▶▶

Interaktionsmenü

Beschreibungstext mit Informationen

10%

☰



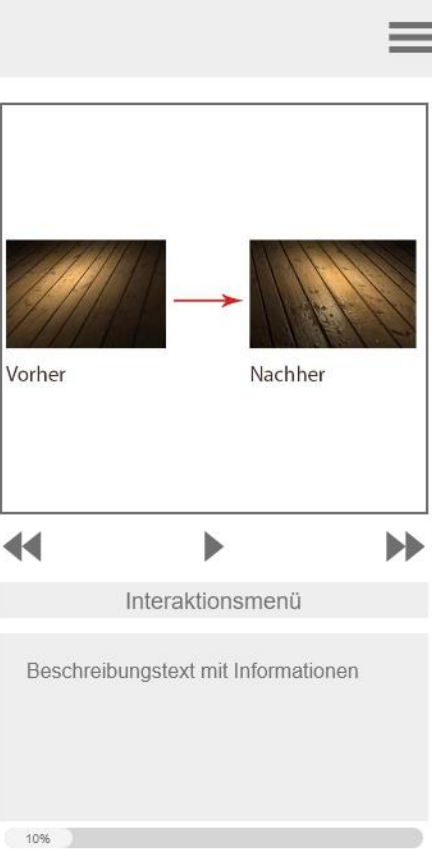
◀◀ ▶ ▶▶

Interaktionsmenü

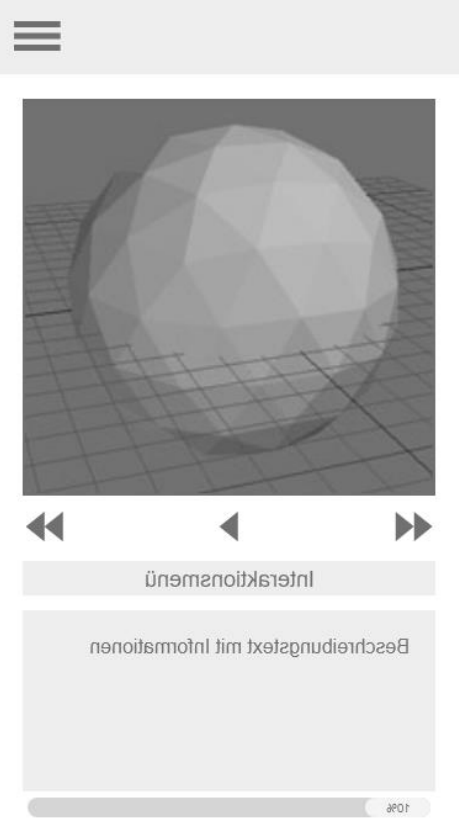
Beschreibungstext mit Informationen

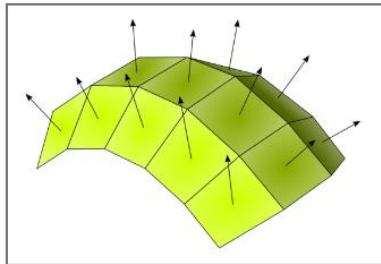
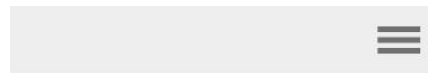


## 5. Pixel- / Fragment-Shader

Screen	Sprechertexte	Stichwörter / Notizen	Regieanweisungen
 <p>The screenshot shows a mobile application interface. At the top is a grey header with a hamburger menu icon. Below it is a large white area containing two side-by-side images of a wooden floor, labeled 'Vorher' (Before) and 'Nachher' (After), with a red arrow pointing from the first to the second. Below the images are three navigation arrows (left, center, right). Under the arrows is a grey button labeled 'Interaktionsmenü'. Below that is a large grey rectangular area labeled 'Beschreibungstext mit Informationen'. At the bottom is a progress bar showing 10% completion.</p>	<p><b>110501</b> Pixel-Shader, auch Fragment-Shader genannt, sind Programme welche ebenfalls in den Shadereinheiten eine Grafikkarte ausgeführt werden. Als Fragment werden in der Regel einzelne Pixel bezeichnet. In der Grafikpipeline folgt der Pixel-Shader auf den Geometry-Shader.</p> <p>Pixel-Shader werden genutzt, um eine realistische Darstellung von Oberflächen- und Materialeigenschaften von Fragmenten zu erreichen oder die Texturdarstellung zu verändern. Ein Pixel kann dabei aus mehreren Fragmenten bestehen, was zum Beispiel bei Transparenz der Fall ist. Zur Anwendung kommen Pixel-Shader unter anderem beim Phong Shading, welches in einem separaten Kapitel behandelt wird.</p>	<p><b>110501</b> Darstellung realistischer Oberflächen- und Materialeigenschaften.</p>	<p><b>110501</b> Die Anwendung des Pixel-Shaders wird beispielhaft an einem Bild gezeigt, sodass man sieht was sich durch den Pixel-Shader verändert.</p>

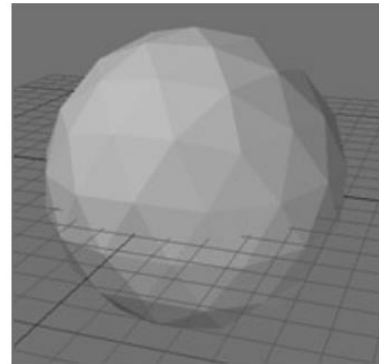
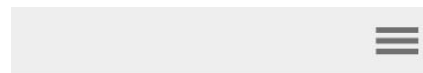
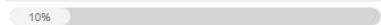
## 6. Flat-Shading

Screen	Sprechertexte	Stichwörter / Notizen	Regieanweisungen
	<p><b>110601</b> Flat-Shading, manchmal auch Constant-Shading genannt, ist ein sehr einfaches Schattierungsverfahren. Das sieht man daran, dass pro Polygon nur eine Farbe möglich ist. Wenn man z.B. ein Dreieck als Polygon nimmt, wird der Farbwert aus dem Flächen-Normalenvektor, welcher sich in der Mitte des Polygons befindet, berechnet. Danach werden alle Pixel des Polygons auf diese Farbe gesetzt. Als Ergebnis erhält man dann, besonders bei gekrümmten Oberflächen, eine Facettenartige Darstellung. Das ist auch der größte Nachteil des Flat-Shading, weil es dadurch zum sogenannten Mach-Band-Effekt kommt, wodurch die entstandenen Kanten besonders stark vom menschlichen Auge wahrgenommen werden.</p> <p><b>110602</b> Um die facettenartige Darstellung zu vermindern, kann man die Anzahl der Polygone erhöhen, wodurch sich aber der Rechenaufwand stark erhöht. Aufgrund der genannten Nachteile kommt das Flat-Shading meistens bei Objekten mit ebenen Flächen wie z.B. Quader, Würfel oder Pyramiden zum Einsatz.</p>	<p><b>110601</b></p> <ul style="list-style-type: none"> <li>- Einfaches Schattierungsverfahren</li> <li>- Eine Farbe pro Polygon</li> <li>- Facettenartige Darstellung</li> </ul>	<p><b>110601</b> Anhand eines Polygons wird gezeigt wie die Fläche, mithilfe des Flächen-Normalenvektors, gefärbt wird. Danach sieht man eine Kugel die aus Polygonen besteht.</p> <p><b>110602</b> Die Anzahl der Polygone wird erhöht, um zu zeigen wie sich der Mach-Band-Effekt verringert.</p>



Interaktionsmenü

Beschreibungstext mit Informationen

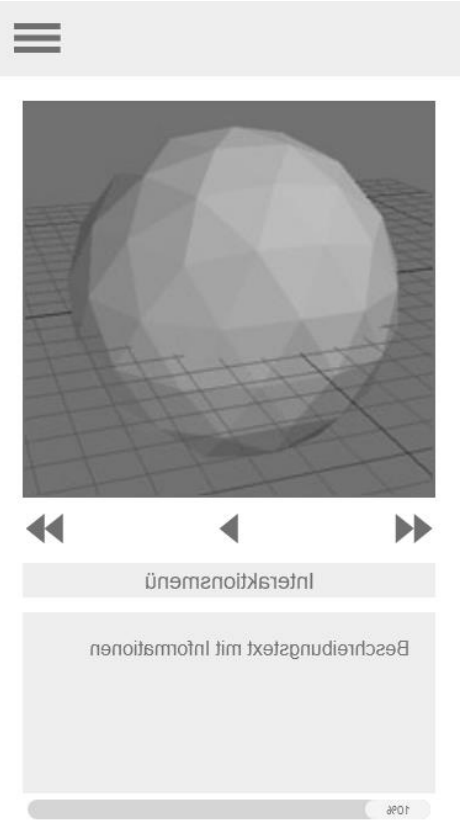


Interaktionsmenü

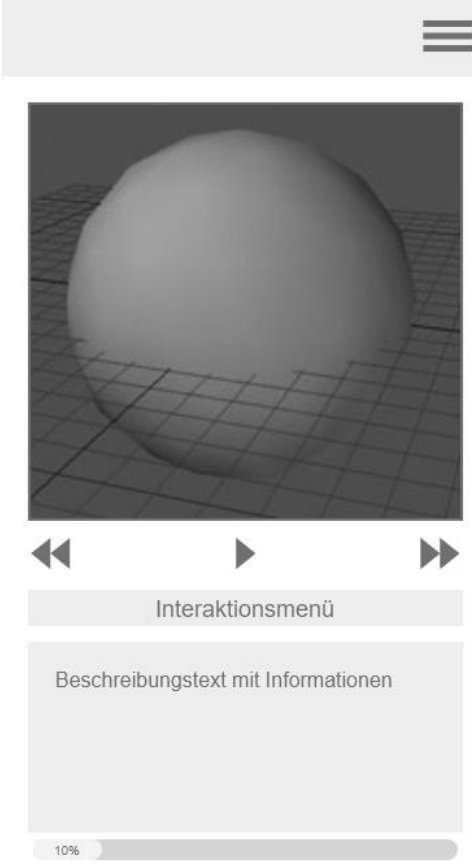
Beschreibungstext mit Informationen



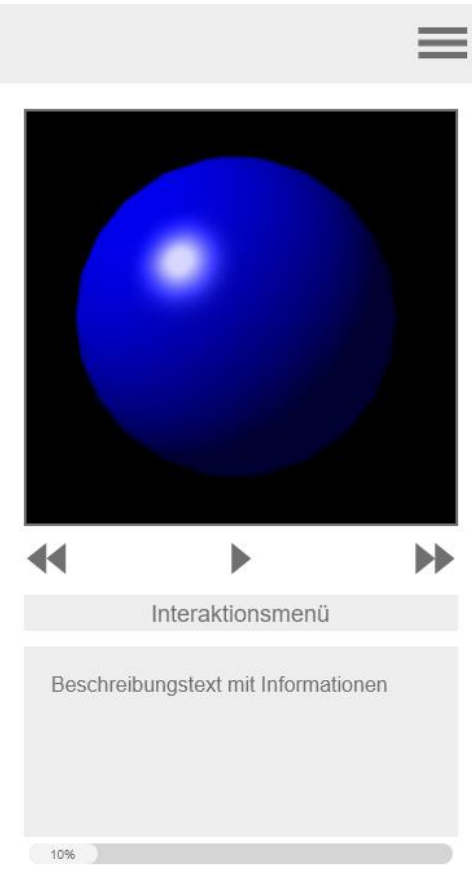
## 7. Flat-Shading - Interaktion

Screen	Sprechertexte	Stichwörter / Notizen	Regieanweisungen
	-	-	<b>110701</b> Die Anzahl der Polygone kann verändert werden, sodass man sieht welchen Einfluss es auf das Endergebnis hat.

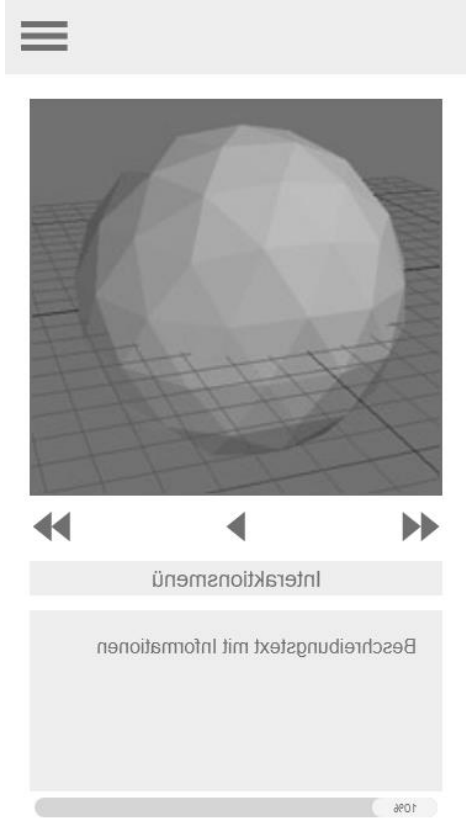
## 8. Gouraud-Shading

Screen	Sprechertexte	Stichwörter / Notizen	Regieanweisungen
	<p><b>110801</b> Beim Gouraud-Shading handelt es sich um ein Verfahren, welches Polygonflächen füllt. Das Verfahren wurde nach seinem Entwickler Henri Gouraud benannt, der es erstmals 1971 vorstellte.</p> <p>Das Besondere am Gouraud-Shading ist, dass im Gegensatz zum Flat-Shading, Farbverläufe dargestellt werden können. Dafür werden die Normalenvektoren an den Eckpunkten berechnet. Diese erhält man durch den Mittelwert der Normalen aller angrenzenden Polygone. Danach werden durch Interpolation die Werte aller Pixel im Polygon berechnet.</p> <p><b>110802</b> Durch dieses Verfahren erscheinen die Kanten der Polygone weich, wodurch Objekte besser rund oder gekrümmt dargestellt werden können. Ein Nachteil ist die bei manchen Objekten fehlerhafte Darstellung von Glanzlichtern und das Vorkommen von Sprüngen im Farbverlauf.</p>	<p><b>110801</b> - 1971 vorgestellt - Darstellung von Farbverläufen - Berechnung mit Hilfe von Normalenvektoren</p> <p><b>110802</b> Für gekrümmte Objekte geeignet</p>	<p><b>110801</b> Anhand eines Polygons wird gezeigt wie die Fläche gefärbt wird. Dafür werden die Normalenvektoren beispielhaft an einem Polygon dargestellt.</p> <p><b>1108012</b> Später sieht man eine Kugel die aus Polygonen besteht.</p>

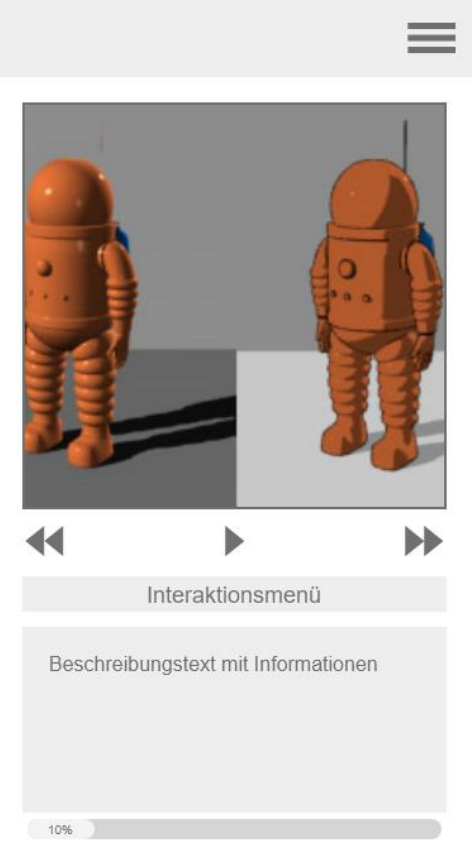
## 9. Phong-Shading

Screen	Sprechertexte	Stichwörter / Notizen	Regieanweisungen
	<p><b>110901</b> Phong-Shading ist ein Verfahren, um Polygon-Flächen mit Farbschattierungen zu versehen. Benannt wurde es nach seinem Entwickler Bui Tờng Phong, der es erstmals 1975 vorstellte.</p> <p>Bei diesem Verfahren werden zu Beginn, wie beim Gouraud-Shading auch, die Normalen an den Eckpunkten eines Polygons berechnet. Daraufhin wird beim Einfärben eines Pixels eine neue Normale zwischen den Eckpunktnormalen interpoliert, mit der das Beleuchtungsmodel ausgewertet wird. Dadurch erhält man für jeden Pixel die entsprechende Beleuchtung und somit auch die korrekte Farbe.</p> <p>Das Phong Shading liefert ein realistischeres Ergebnis als das Gouraud Shading, ist aber auch deutlich rechenintensiver.</p>	<p><b>110901</b></p> <ul style="list-style-type: none"> <li>- 1975 vorgestellt</li> <li>- Berechnung mit Normalen</li> <li>- Beleuchtung für jeden einzelnen Pixel</li> </ul>	<p><b>110901</b> Anhand eines Polygons wird gezeigt wie die Fläche mithilfe der Normalen gefärbt wird. Danach sieht man eine Kugel die aus Polygonen besteht.</p>

## 10. Vergleich zwischen Flat-, Gouraud - und Phong-Shading - Interaktion

Screen	Sprechertexte	Stichwörter / Notizen	Regieanweisungen
	-	-	<b>111001</b> Man kann die verschiedenen Shader an einem Objekt, in dem Fall einer Kugel, anwenden. Durch diese Interaktion kann man gut erkennen, welche Unterschiede es zwischen den Shadern gibt und welchen Einfluss sie auf das Endergebnis haben.

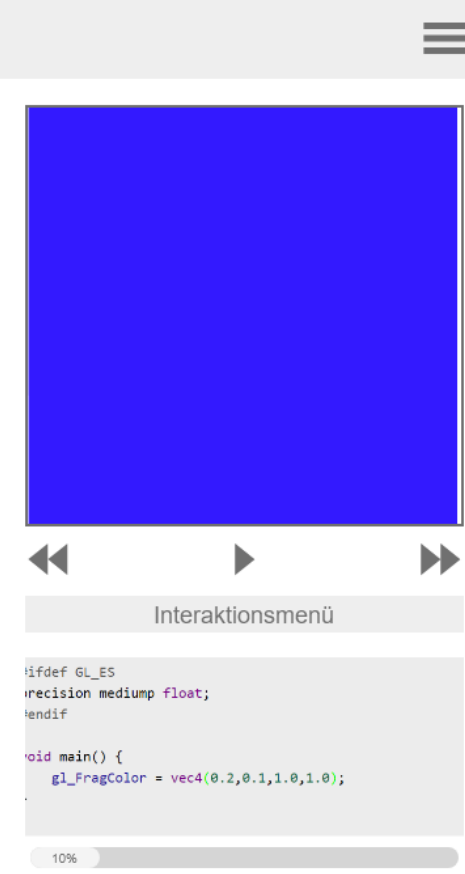
## 11. Toon-/Cel-Shading

Screen	Sprechertexte	Stichwörter / Notizen	Regieanweisungen
	<p><b>111101</b> Beim Toon-Shading, auch Cel Shading genannt, handelt es sich um eine Technik zum nicht-fotorealistischen Rendern von 3D-Computergrafiken. Als Ergebnis erhält man bei diesem Verfahren eine Optik, die der von gezeichneten Comics oder Zeichentrickfilmen entspricht.</p> <p>Um diesen Effekt zu erhalten wird auf weiche Verläufe verzichtet und es kommen nur drei oder vier Helligkeitsstufen zum Einsatz, in der Regel weiß, hellgrau und dunkelgrau. Außerdem verzichtet man meistens auf eine Textur und verwendet nur einzelne Farbtöne. Der benötigte Grauwert wird über den Winkel zwischen dem Normalenvektor des Polygons und dem Vektor vom Polygon zur Lichtquelle berechnet.</p> <p>Um die schwarzen Linien zu erhalten, welche die Kontur des Objekts darstellen, invertiert man Polygone die aufgrund der Perspektive nicht sichtbar wären. Dafür wird das Backface Culling, welches nicht sichtbare Polygone aufgrund der Performanceverbesserung entfernt, rückgängig gemacht. Dies wird teilweise mehrmals mit</p>	<p><b>111101</b> - Comic-Optik - 3-4 Helligkeitsstufen - einzelne Farbtöne - Intervenieren von Polygonen um Konturen zu erhalten</p>	<p><b>111101</b> Zuerst wird gezeigt wie ein Toon-Shader in der Praxis aussieht. Dann wird Schritt für Schritt gezeigt, wie der Toon-Shader auf ein normales Objekt angewendet wird. Als erstes werden die Farben reduziert und dann die Konturen erstellt.</p>



	leichten Variationen durchgeführt, um eine bessere Kontur zu erhalten.		
--	--	--	--

## 12. Code Beispiel - Interaktion

Screen	Sprechertexte	Stichwörter / Notizen	Regieanweisungen
 <p>The screenshot shows a web interface for a shader example. At the top right is a hamburger menu icon. Below it is a large blue square representing the rendered output. Under the square are three navigation arrows: a double left arrow, a single right arrow, and a double right arrow. Below the arrows is a label 'Interaktionsmenü'. Underneath is a code editor showing GLSL code. At the bottom is a slider control set to 10%.</p> <pre> #ifdef GL_ES precision mediump float; #endif  void main() {     gl_FragColor = vec4(0.2,0.1,1.0,1.0); } </pre>	<p><b>111201</b></p> <p>In diesem Kapitel bekommt man einen einfachen Einblick in den Code eines simplen Shader-Programmes. Dieser Code ändert die Farbe des angezeigten Quadrats. Der Code eines Shaders erinnert stark an die Programmiersprache C, weshalb es viele Parallelen zu dieser und auch anderen Programmiersprachen gibt. Dieses Beispiel wurde in der OpenGL Shading Language, kurz GLSL, geschrieben. Mit precision mediump float; wird die Genauigkeit der Berechnungen mithilfe des Fließkomma-Datentypen auf Mittel gestellt. Alternativ kann man die Genauigkeit auch auf lowp oder highp stellen, was Einfluss auf die Performance und die Qualität hat. Daraufhin folgt eine main-Funktion, welche am Ende einen Farbcode zurückliefert. In dieser Funktion sehen wir die Variable gl_FragColor mit dem Datentyp vec4. Die Zahlenwerte in dieser Variablen stehen für die Farbkanäle Rot, Grün, Blau und Alpha. Der Farbkanal Alpha gibt dabei die Transparenz an. Je nachdem welche Zahlenwerte man angibt, erhält man am Ende eine andere Farbe.</p>	<p>-</p>	<p><b>111201</b></p> <p>Man kann den Code selber bearbeiten und erhält dann eine Farbe als Ergebnis.</p>